Peer to Peer (P2P) applications for searching and exchanging information over the internet are highly popular. For a long time schema based approaches are being utilized to store and retrieve data. Resource Description Framework (RDF) based P2P networks have a number of advantages compared to a simpler P2P networks such as Napster, Gnutella or with approaches based on distributed indices such as Content addressable network (CAN). RDF based P2P networks allow complex and extendable descriptions of resources, instead of fixed and limited ones and they provide complex query facilities against these metadata instead of simple keyword searches. This project is an attempt to overcome some of the challenges to build a super peer based P2P network and produce an effective routing strategy among the super peers.
## TABLE OF CONTENTS

Abstract ............................................................................................................................ ii

List of Figures ................................................................................................................ iv

1. Background and Rationale ......................................................................................... 1
   1.1 What is a peer to peer network ........................................................................ 1
   1.2 The concept of data integration ........................................................................ 3

2. Narrative ..................................................................................................................... 7
   2.1 Building Blocks and Challenges ......................................................................... 7
       2.1.1 Schema languages .................................................................................. 7
       2.1.2 Query languages ................................................................................... 8
       2.1.3 Network Topologies and Clustering ......................................................... 8
       2.1.4 Information integration .......................................................................... 9

3. System Design ........................................................................................................... 11
   3.1 System Requirements ....................................................................................... 11
   3.2 Code Design .................................................................................................... 11
   3.3 Database Design ............................................................................................. 12
   3.4 How the system works? ................................................................................... 13
   3.5 Proposed System Algorithm ......................................................................... 17

4. Testing and Evaluation ............................................................................................... 27

5. Future Work ............................................................................................................. 34

6. Conclusion ............................................................................................................... 35

7. Acknowledgements ................................................................................................. 36

8. Bibliography and References .................................................................................. 37
LIST OF FIGURES

Figure 1.1 Peer to Peer Model.................................................................2
Figure 1.2 Client Server Model...............................................................2
Figure 1.3 A Super Peer Based p2p Network..........................................6
Figure 1.4 Schema Capabilities Distribution..........................................6
Figure 2.1 Example HyperCuP Super Peer Topology..............................8
Figure 3.1 Super Peers-Peers relationships inside the database...............12
Figure 3.2 Example Super peer network ..............................................14
Figure 3.3 Example Super Peer/Peer Routing Index..............................15
Figure 3.4 Super Peer/Super Peer Routing Index.................................16
Figure 3.5 Image of the UI for initializing the network.......................22
Figure 3.6 Sample image from UI showing P2P network......................23
Figure 3.7 Image from the UI to enter Peer data.................................24
Figure 3.8 Image from the UI shown before data search......................25
Figure 3.9 Image from the UI displaying search results.......................26
Figure 4.1 Peers being added to the network.....................................28
Figure 4.2 Template from GUI with Data being entered....................29
Figure 4.3 Template from GUI with peer Metadata being published........29
Figure 4.4 SP/SP RI details from GUI template.................................30
Figure 4.5 Initial network structure before a continuous search..........31
Figure 4.6 Network Structure modified as a result of continuous search....31
1. BACKGROUND AND RATIONALE

We all learn things from other people who are willing to share their knowledge with us in the same way we share our knowledge with others. In this sense human society can be viewed as a network of ‘Knowledge sharing peers’, the power of which can be appreciated by the fact that most of the things we know actually come from our participation in this network. In the same way that humans possess knowledge about the world, so do organizations like universities, companies etc. Such organizations store information as database information. Greater possibilities would arise if this knowledge could be shared in the same way as humans. However, creating a peer to peer (P2P) system where peers can share semantic rich information (databases) is very difficult.

1.1 WHAT IS A PEER TO PEER NETWORK?

A peer to peer system is a network of machines where all participant machines act as service providers to the entire network and at the same time take advantage of the collective service provided by the other participant machines [P2P 2008]. A pure peer to peer computer network does not have a notion of clients or servers, but only equal peer nodes that simultaneously function as both client and servers to the other nodes of the network. Figure 1.1 shows a sample P2P network.
The P2P model of network arrangement differs from the client-server model in which the communication is usually to and from a central server. In a client server model, the server is the driving force of the network and all the communication among the clients happens through the server. Figure 1.2 shows an example client server model.
The dynamic nature of a P2P network, the lack of centralized administration and the fact that its power increases as more peers join the network makes the P2P networking a very attractive model and has been used successfully by popular file sharing systems e.g. Napster, Gnutella, CAN etc [Morris 2001].

But sharing files is easy, since there is no consideration of the semantics of the data (the meaning of the information in the files) and a file is identified by a small fixed set of attributes. Hence, locating a file can be done alone by a simple string matching. This makes the communication feasible between the peers. So what would happen if we tried to move from sharing just plain files to sharing richer and more structured data like database information?

Typically different organizational databases will use a different schema (structure) for the same type of information. The problem in sharing this information is how to integrate the different schemas so that the participants can communicate with each other. The problem becomes even more difficult when dealing with a peer to peer system due to the above mentioned features. Integration of data in a P2P system is referred to as Peer to Peer Data Integration (P2PDI).

1.2 THE CONCEPT OF DATA INTEGRATION.

It is a modern world axiom that ‘information is power’. Organizations keep large databases for their own use, but it is often the case that information in one organization can be useful to another and vice versa and therefore if the two organizations could share their information, it would mean that both their information power would increase and benefit both. However, the databases of the two organizations are typically heterogeneous
(Consisting of elements that are not of the same kind or nature) which does not allow the data to be unified.

Generalizing the above, it is not surprising that large enterprises, business organizations, e-government systems and in short any kind of internetworking community needs an integrated virtualized access to distributed information resources. There is a need for a way to share the information possessed by each member of the community for the benefit of the entire group but still allowing each member to have a local authority to its own resources. This is the problem that integration attempts to tackle.

First we should give a more precise definition of what data integration refers to. Lenz defines data integration as “The problem of combining data residing at different sources and providing the user with a unified view of these data” [Lenz 2004]. A majority of approaches in data integration attempt to somehow form a mapping between schemas (over all structure of the database), such that either the data that is defined in a schema can be materialized in the mapped schema or a query expressed to a schema can be translated to a query defined to the mapped schema.

One of the most important aspects in the design of data integration systems is the specification of the correspondence between the data at sources (local schema) and those in the global schema. [Lenz 2004] . There is four basic approaches:

- **Local as View (LAV) [wiki 2008]**: In a LAV approach, each data source is expressed as a view over the global schema. This means that the sources are defined in terms of the global schema.
• Global as view (GAV) [wiki 2008]: In the GAV approach, there is an association between each element of the global schema and a view over the sources. This means that the information held in the global schema is defined in terms of the sources.

• Global local as view (GLAV) [Haas 1997]: Combines the expressive power of both GAV and LAV. It can be considered as a variation of LAV in the sense that the global schema is independent of the sources.

• Both as View (BAV) [Levy 1996]: BAV is an approach to data integration, where schemas are mapped to each other using a sequence of schema transformations. An important feature of BAV is that the transformations are bi-directional and it’s therefore easy to go from the local to global schemas and vice versa.

• Peer to peer (P2P) [wiki 2008]: Unlike the other techniques, in P2P systems, data integration is not based on a global schema. Instead there is a network of peer nodes which can define mappings from themselves to other nodes thus forming a logical association between peers. Figure 1.3 shows a sample super peer based P2P network.

In Figure 1.3, the nodes colored green are called super peers or super nodes. Super peers contain the schema information about its normal peers. Each super peer is connected to its neighboring super peer and the communication between the peers happen via the super peers.
P2P networks have evolved from simple systems like Napster and Gnutella to more complex systems based on distributed indices (e.g. distributed hash tables) such as CAN and CHORD. RDF and RDF schema are used to annotate resources on the web thus providing a means by which the computer systems can comprehend and exchange data. In contrast to the traditional database management systems, RDF metadata has the ability to use distributed annotations for a resource which makes RDF suitable for constructing distributed repositories. Moreover RDF schemas are flexible and extendible such that schemas can evolve over time and RDF allows the easy extension of schemas with additional properties.

Figure 1.4 describes the schema capabilities distribution for various distributed and P2P projects. A considerable research has been done to refine topologies and query routing functionalities in schema based and peer to peer models like Edutella and Piazza. In the view below, schema based P2P systems are the point where the research on P2P
applications and their query routing capabilities meets with the database research. Figure 1.4 depicts a few P2P models with their schema capabilities distribution.

Fig 1.4: Schema capabilities distribution [Siberski 2003]
2. NARRATIVE

The primary goal of this project is to come up with an optimal routing and clustering strategy for super peer based P2P networks. This will allow us to dynamically add and remove either super peers or peers from the system without affecting the system on a whole. This will also allow a user using this system to do a search on multiple databases efficiently bring up the precise search results and save time.

2.1 Building Blocks and Challenges:

To provide a schema based P2P infrastructure, we need at least the following building blocks:

1. A schema language to define and use the various schemas which specify the kind of data available in the P2P network.
2. A query language to retrieve the data stored in the P2P network.
3. A network topology combined with an appropriate query routing algorithm to allow efficient queries
4. Facilities to integrate heterogeneous information stored in the P2P network.

2.1.1 Schema languages

One important aspect of the overall design of the semantic web is the exchange of data among computer systems without the need of explicit consumer-producer relationships.
2.1.2 Query languages

Any schema based or RDF based P2P networks need complex query capabilities rather than simple key or keyword based queries. It is therefore, required that a query language supports the definition of semantics of several schema languages.

2.1.3 Network Topologies and Clustering

Query routing in P2P networks benefits from suitable clustering algorithms, where the peers are arranged in the P2P network based on their peer characteristics. A sample arrangement of databases (peers) is showed in Figure 2.1.

![HyperCuP Super-Peer Topology](image)

Fig 2.1: Example HyperCuP Super-Peer Topology [Nejdl 2003]

Figure 2.1 shows a sample arrangement of the peers in a HyperCuP topology [Nejdl 2003]. The one database (peers) in the bigger circles is called as super peer and the remaining in smaller ones are called as ordinary peers. Super peer based networks can provide better scalability when compared to broadcast networks and provide support to
schema based networks with different metadata schemas and ontology’s which are crucial for the semantic web.

Query routing in Peer to Peer networks benefits from suitable clustering algorithms, where the peers are arranged in a P2P network based on their peer characteristics. A few among the clustering strategies are:

- **Rule based clustering:** The main idea behind rule based clustering is to group and register peers in subject specific clusters via cluster specific rules. Every cluster provides its own rules, expressing which peers are allowed to join the cluster and which peers are denied to enter the cluster.

- **Query based clustering:** This method employs a dynamic way to take the characteristics of queries into account while deciding the clustering parameters. Each peer, super-peer and query is characterized by a set of items, which can be used to cluster similar peers.

2.1.4 Information integration

The final step is to integrate the data from different peers and that too needs to be done from peers having different schemas. There are two ways to achieve information integration

- **Distributed query processing:** In distributed query processing the P2P network is queried for information satisfying some given conditions, and this query is routed to the peers which can answer it. When a possibly large number of results from distinct data sources are returned to the client peer, further query processing takes place centrally at the client.
• **Mediation:** Mediation is a service that is responsible for locating peers and exchanging their data between two peers that search for each other.

Depending on the dynamic metadata routing indices, the super peer networks can support sophisticated routing and distributed strategies. Routing indices contain the schema identifier as well as the peers supporting this schema.
3 SYSTEM DESIGN

This database integration project aims at proposing a search algorithm for searching data in a peer to peer network. This involves first producing a network of peers and then publishing the peer metadata onto super peers so as to optimize the search in the network. The proposed algorithm supports wider range of schema design and thus not limiting to a specific schema structures.

3.1 System Requirements

The proposed algorithm is implemented in a GUI which is designed in order to retrieve data from various peers. The system is built using Visual C# for programming the GUI and MS Access 2003 as the database at the backend. The system utilizes simple tree topology to simulate the system design. The usage of simple tree network topology allows us to simplify the implementation of the proposed algorithm on a single machine and thus simulating the real world scenario where a number of peers needs to be in a network. This also simplifies implementation of the system by allowing dynamic addition or removal of peers to the network and updating the peer metadata at the super peers.

3.2 Code Design

The code is designed using Microsoft Visual C# language. The code could be executed using the Microsoft Visual C# 2005 express edition. The code is divided in two parts, firstly to describe the data into the database (forming a network) and then publishing it on the network for the search.
3.3 Database Design

The database is designed to store relations among super peers and peers and to store the routing indices in a one to many fashion. All network nodes details are implemented in a database using MS Access engine. The peers store information about the parent super peer where it publishes the peer metadata in a near RDF fashion. Similarly the super peer stores information about adjacent super peers where it publishes the metadata information. Figure 3.1 below shows the layout of peer relationships inside the database.

![Diagram showing peer relationships in the database]

Fig 3.1: super peer-peers relationships inside the database.

All objects inside the database are uniquely identified by an identifier key (ID). Super peers keeps information about the parent super peer using a super peer id key (SPID)
variable whereas a peer stores information about its super peer using the same variable (SPID). The super peer and peer metadata objects store information about the neighboring super peers and peers using the fields like peer id (PID) and metadata id (MDID) variables.

3.4 How does the System Works?

At the beginning, the network has only one main super peer which contains no super peer/super peer routing indices (SP/SP-RI’s), which is also considered as Tree Root or root super peer. Any other super peers should be connected as leaves of this node or as leaves of root leaves without limitation in depth in the network. The following assumption simplifies the system design:

- Peers are only connected to super peers, which have no super peers as leaf node. Therefore, a peer node at the branch cannot have a super peer node at the same branch.

While the network grows more than two leaves, nodes at the second level are considered as its subnet gateway, as any search queries pass through it, the query will be directed to destination peer using SP/SP RI in leaf nodes. As soon as it reaches destination super peer’s complete path will be sent back to the querying super peer. The querying super peer will analyze that path of the query result and if it discovers that the query passed out its network, it will start to create a counter to hold the number of times the particular super peer has made a query to this remote super peer. When gateway counter reaches a predefined threshold, source super peer is automatically moved to the
closest destination super peer. The gateway counter is reset automatically if it not used for a longer time.

The search algorithm proposed in this project uses the simple tree network topology for the network structure. The use of simple tree topology simplifies the network complexity and helps optimize search in the network.

A root node initializes the network being the root super peer. Peers connect to the super peers in a tree node fashion providing the content and the content metadata to the super peers. The introduction of super peers reduces the workload of peers significantly by distributing queries to appropriate subsets of all possible peers only. The network of peers and super peers performs the search based on the peer metadata published at the super peer and the routing indexes among peers and super peers.

The initial indices needed at the super peers are called super-peer/peer routing indices (SP/P RI-s). These initial indices consist of the information about the metadata usage at each peer. Figure 3.2 shows a sample super peer based P2P network containing super peers SP1 through SP4 and peers P0 through P4.
At each super peer (SP) there contains a routing index which contains the information about neighboring super peer and local peers as well. Figure 3.3 shows a sample Super peer/peer (SP/P) routing index example. The database design permits peers to have support for different schemas. The routing index contains the schema identifier as well as the peers supporting this schema.

Fig 3.2 An Example of super peer network [Nejdl 2003]

Fig 3.3: Example Super Peer/Peer Routing Index [Nejdl 2003]
In a similar fashion like Super Peer/Peer (SP/P) routing indices, the super peers will also contain the Super Peer/Super Peer (SP/SP). The information about the neighboring Super peers is contained in the SP/SP routing indices. Figure 3.4 shows Super peer/Super peer routing index used in this project. To avoid broadcasting queries to all super peers, Super Peer/Super Peer routing indexes are utilized (SP/SP RI-s). The SP/SP RI-s contains similar information like SP/P RI-s but refers to neighboring super peers based on the SP/SP indices as shown below in Figure 3.4.

Before the system operation each peer has to register itself with a super peer using the so called advertisements, which contain the metadata schema used at the peer. Upon registration of the peer metadata at the super peer, the peer provides the super peer with the metadata information for publishing an advertisement. At each super peer, elements used in a query are matched against the SP/P routing indexes in order to determine local peers which are able to answer the query. If the resulting query data is contained in the peer, the peer information and data are returned or else an empty result set is returned.

Update of SP/SP indices is based on the registration of peer metadata. When a new peer registers with a super peer, it publishes the necessary schema information to the super peer. The super peer thus matches this information against the entries in its SP/P
routing index. Every time peer elements have to be updated the SP/P routing index needs to be updated for more accurate publishing of metadata over the network. The network is made dynamic and optimized in nature in different ways:

1. Peers could be added or removed from the network dynamically by updating the SP/P RI-s. This gives the flexibility to add new sets of peers without destroying the whole network and vice versa. Also, updating the SP/P RI while adding or removing a peer does not affect the query search time.

2. The query search among the peers is optimized by making the network self learning. Any query which searches for a remote peer for a large number of times is logically moved under the same super peer where the search is being performed. This significantly reduces the time and cost to search for a query result.

3.5 Proposed System algorithm

Super peer network has main functions needed to first build the network, which are (CreateSuperPeer, Create Peer, DestroySuperPeer, Destroy Peer, Add Metadata for Peer, MoveSuperPeer, and Search), which are as following:

- CreateSuperPeer:
  1- Create a root super peer to initialize the network.
  2- Set ParentSPID to desired Parent SP ID.
  3- Add SuperPeer object to the database.

- DestroySuperPeer:
1- If SP has child SPs exit procedure.

2- Check if this SP has child Ps, if it has child go to step 3, if not go to step 4.

3- Delete all children Ps with all related Metadata and SP/SP RI. In parent SPs (which are done automatically by executing Destroy Peer procedure).

4- Delete SP from DB tables, DB will care about deleting any related fields in other tables.

- Create Peer:
  1- Create a peer.
  2- Set SPID to desired SuperPeer ID.
  3- Add Peer object to the database.

- Destroy Peer:
  1- Get Peer Metadata list and pass it to PublishRemoveMetaData function.
  2- Delete Peer from DB, and its related data in other tables.
  3- Delete SuperPeerMetaData fields related to this Peer (SP/P RI).

- PeerAddMetaData:
  1- Add schema element to elements list.
  2- Add schema type to every element will be added to the database.
  3- Add Metadata list to Peer Metadata table in database.
  4- Use PublishAddMetaData function in updating SP/SP RIs of parent nodes (SPs) in network.
- **PublishAddMetaData:**
  1- Check if SP/SP RIs should be updated or SP/P RIs. If SP/P RIs so Meta Data list will be added to SuperPeerMetaData table in DB, and go to step 6.
  2- Update Parent SP/SP RIs, looping on Metadata list.
  3- If Metadata element and value pair are found in SP_SP_RI table then proceed, else loop to next element and go back to step 2.
  4- Search for SPIID in SPIIDs field, if found its counter should be incremented, else this SPIID should be added to SPIIDs with counter set to 1.
  5- Repeat from step 2 for all elements. After processing all elements and updating current SP with Metadata list, then proceed to next step.
  6- Get Parent Super Peer ID, if it’s not the equal to 0 (means current SP is the root, so there is no parent nodes) then steps from 2 will be repeated.

- **PublishRemoveMetaData:**
  1- Check if SP/SP RIs should be updated or SP/P RIs. If SP/P RIs so Meta Data list will be deleted from SuperPeerMetaData table in DB, and go to step 6.
  2- Update Parent SP/SP RIs, looping on Metadata list.
  3- If Metadata element and value pair are found in SP_SP_RI table then proceed, else loop to next element and go back to step 2.
4- Search for SPID in SPIDs field, its counter should be decremented, when count reaches 0 this SPID will be deleted from SPIDs field.

5- Repeat from step 2 for all elements. After processing all elements and updating current SP with Metadata list, then proceed to next step.

6- Get Parent Super Peer ID, if it’s not the equal to 0 (means current SP is the root, so there is no parent nodes) then steps from 2 will be repeated.

- MoveSuperPeer:
  1- Update All SP/SP RIs for parent SPs.
  2- Modify Super Peer ID in database.

- Search:
  1- Get Parent SP object for currently searching peer.
  2- Check if current SP has child peers. Unless go to step 5.
  3- Check SP/P RI for Metadata list, if found Peer ID should be added to result list.
  4- If current SP is not root, so parent SP of current SP should be checked by executing step 2 recursively. Exit with current results list.
  5- Check SP/SP RIs for current SP. And collect all SP IDs that have those Metadata list elements and there relevant values (List is
anded) under it in other SPs. Then check every possible SP ID recursively downward till reaching bottom nodes. Then repeat from step 2 through all SPs.

The most important part in this algorithm is its structure flexibility, as the network cares about search performance, moving super peers which has peers (those at the bottom of tree) close to their frequently searched super peers can enhance system performance dramatically. That is why every super peer object holds a counter and timer array list for other accessed super peers in prior search operations, this functionality of how network utilizes is described as follows:

- **Update Network**
  1- Get Super Peer object from list in SuperPeersNetwork object, and check if it has counters list. If not go to step 4.
  2- If it has counters list, then all counters should be checked as if Hit Threshold has passed the maximum limit (which is configurable as system parameter), then that searched SP has to be moved close to searching SP.
  3- If time period (configurable) has elapsed then timers should be reset.
  4- Check for next Super Peer in list and go to step 1. If no more Super Peers then Update Network should end. It’s should be called periodically. So network will keep updated and efficient in search functionality
A graphical representation of the project can be viewed by executing the code in Visual C#. For added flexibility a sample database in MSAccess is attached. After executing the project in Visual C# the GUI appears as shown in Figure 3.5 below:

![Image of the UI for initializing the network](image)

**Fig 3.5: Image of the UI for initializing the network**

At this point the network is not initialized and needs to be initialized. The initialize entry under the network on the toolbar of UI will initialize the network hence producing the Root super peer as shown in Figure 3.5. The root super peer will contain super peers and peers below it. After we initiate the network with a root super peer, we can create a super peer(s) by clicking on Create Super peer on the right of the window. This will create a super peer below the root super peer. Now that the network is initialized and a super peer(s) been created, peer(s) could be joined to the network. This could be done by the clicking on the Create Peer control on the right of the screen. A
super peer could have any number of peers below it. The Figure 3.6 illustrates this behaviour:

![Figure 3.6: Sample image from UI showing P2P network](image)

Now that the network being created data could be added manually by clicking on the peers or using the ‘MetaData Entry’ tab under the ‘Network’ link on the toolbar. The Using the ‘MetaData Entry’ under the ‘Network’ link on the toolbar allows us to enter data from existing datasource rather than entering it manually. The system design allows us to use a wider range of schema object to be added to the search, rather limiting the system to use a specific set of schema. The system behaviour to add MetaData from an external data source has been illustrated in Figure 3.7 below:
Once the peer has its metadata published to the super peer, search on the whole network could be performed using the ‘Searching Data’ control on the bottom right side of the screen. To perform a search the system has to be supplied with search values. This behaviour is illustrated in Figure 3.8 below:
Fig 3.8: Image from the UI shown before data search

The query search depending on the search criteria will output all the information about the peer and the available metadata on the GUI as shown below in Figure 3.9.
Fig 3.9: Image from the UI displaying search results

The query when search against the element ‘dc: ename’ for a value of ‘Kevin’ displays all information available in the data source associated with the record. This facilitates to perform search against a wider range of data sources for a specific element.
4. TESTING AND EVALUATION

Testing after development is an important part of any system. A system until tested could not be declared functional. This project aims at an algorithm which optimizes the query routing and hence reduces the search time when compared with ordinary search techniques. Like any project there were a few test procedures performed on this system as well, which are discussed as follows:

- **Query routing between peers and super peers**
  While searching over a network, the query first searches the super peer metadata thus looking for the result set and only if the super peer metadata matched the query result, the query searches the peer for the resultset. This functionality has been tested using multiple schemas in the peers over the network.

<table>
<thead>
<tr>
<th>GRANULARITY</th>
<th>PROPERTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHEMA</td>
<td>dc;scott</td>
</tr>
<tr>
<td>PROPERTY</td>
<td>dc:ename;dc:epno;scott:telno</td>
</tr>
<tr>
<td>PROPERTY VALUE</td>
<td>Dc:ename:=’george’</td>
</tr>
<tr>
<td></td>
<td>Scott:telno:=’1234567890’</td>
</tr>
</tbody>
</table>

Table 4.1: Sample contents of a query at a granular level

- **Dynamic addition or removal of peers**
  Peers could be added or removed from the network at the time of creation or while the network is already built. The dynamic addition of peers allows us the flexibility to add more number of nodes to expand the search across multiple peers not just during the formation of network but even when the network is under
The requisite of adding a peer is to update the metadata over the network so as the peer could be included for any search. This functionality has been tested in the code and is been illustrated in Figure 4.1 below:

![Figure 4.1: Peers being added to the network](image)

- **Adding Peer MetaData to SuperPeers**

Adding a data to the peer could be done via inserting the records manually or by attaching a database to the network. The resulting peer updates its peer metadata automatically as shown in the Figures 4.2, and 4.3 below:
Figure 4.2: Template from GUI with Data being entered

Figure 4.3: Template from GUI with peer metadata being published
Figure 4.4: SP/SP RI details from GUI template

- **Evaluating the scalability of the algorithm.**

  The proposed algorithm could be scaled to a larger number of nodes but at an expense of greater query response time. Figure 4.5 demonstrates this behavior:

  **Algorithm Evaluation**

  ![Algorithm Evaluation](image)

  Figure 4.5: Evaluation of algorithm scalability
The query response time is directly proportional to the number of nodes as they increase in number.

- **Check network update and move super peers when searched continuously by neighboring peers.**

  When the data inside a peer is searched continuously by a query from another peer, the peer is logically moved to the branch of the querying peer. This behavior is controlled using a counter inside the code to keep track of the number of searches made to the peer. Figure 4.5, below shows the tree containing SP2 and SP3. When a continuous search is made from a node inside SP3, SP4 which is a child node of SP2 is logically moved to the tree branch under SP3 and hence reducing the search time among the nodes. This phenomenon is displayed in Figures 4.5 and 4.6 below:

![Figure 4.5 Initial network structures before a continuous search](image)
Figure 4.6: Network structure modified as a result of continuous search
5. FUTURE WORK

The proposed algorithm could be enhanced by using various ontology’s and could be deployed at a larger scale using wrappers and various other protocols available, e.g. JXTA from Sun Microsystems. Different topologies could also be used to boost the performance depending on the type of data and location of the peers. The tree-root topology used in this system design can be altered with a more sophisticated topology such as HyperCube, which could lead to a much more complex and more efficient network.
6. CONCLUSION

P2P networks have numerous advantages for sharing information over a network. This project proposes and demonstrates a way to optimize the search among peers. The proposed algorithm is an attempt to address a few problems with traditional P2P network and their search functionalities. Moreover, the dynamic nature of the system adds more flexibility to add or remove peers without affecting the system as a whole.
7. ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to Dr. Longzhuang Li, Associate Professor of Computer Science, Texas A&M University – Corpus Christi, for his expertise in area of data integration and information retrieval, unending support, much needed motivation, and for providing guidance for completion of my project successfully.

My Sincere thanks to Dr. Dulal Kar, Associate Professor of Computer Science, Texas A&M University – Corpus Christi for his unending support, and for providing guidance which helped me towards completion of my project.

My sincere thanks to Dr. David Thomas, Associate Professor of Computer Science, Texas A&M University – Corpus Christi, for his keen interest, unending support and warm wishes that provided me the much-needed motivation in completing my project.

And finally my heartfelt gratitude to all the faculty, and staff of the Department of Computing and Mathematical Sciences, my colleagues and friends for their motivational support.
8. BIBLIOGRAPHY AND REFERENCES


