SEMANTIC QUERY ROUTING IN AGENT-BASED PEER-TO-PEER SYSTEMS USING LIGHTWEIGHT COORDINATION CALCULUS

Zeeshan Pervez

Master of Science
(Computer Science)
School of Informatics
University of Edinburgh
2005

Abstract

Agent based peer-to-peer system is an effective way of solving complex problems and sharing data. Messages can efficiently be routed to any particular agent in the network. However determining the destination agent is the actual challenging task. Efficient query routing mechanisms are required to solve this problem. Query routing is a distributed search process in which each agent considers the capabilities of other agents in order to limit the scope of the query and its processing cost. In this project we are applying Lightweight Coordination Calculus (LCC), which is used to write communication protocols for agents, in an agent based peer-to-peer system. Three different types of routing mechanisms namely overt, covert and hybrid are designed in this project using LCC. An expertise based agent selection model has been used in which every agent advertises its expertise to other agents in the network. An agent can select appropriate agents to which to forward a query on the basis of semantic similarity between the subject of the query and agent's expertise. A bibliographic references scenario has been chosen to evaluate the protocols. In this scenario every agent represents a researcher, who wants to share bibliographic description of publications with other researchers on the network. In our simulation based experiments we show that expertise based agent selection improves the recall of the system.

ACKNOWLEDGEMENTS

I am filled with the praise and glory to Almighty Allah, the most merciful and benevolent, who created the universe, with ideas of beauty, symmetry and harmony, with regularity and without any chaos, and gave me the abilities to discover what He thought.

Bless MUHAMMAD (P.B.U.H) the seal of the prophets and his pure and pious progeny.

I would like to thank my supervisor and long-time supporter during my time at University of Edinburgh, Dr. Dave Robertson for all of his guidance, patience, and instruction in the ways of the research world. From early on when I just had our first encounters with multiagent systems and LCC, he guided me to develop my own ideas and he challenged me to make them work. At the same time he helped me to achieve the scientific rigor required for research. This project would not have been possible to complete without his help. There is no part of this work that has not been the object of his penetrating mind, and there has been no question posed to him to which he has failed to apply his full intellect.

I would like to thank Ronny Siebes, Li Guo, Chris Walton, Adam Barker and Jane Hilston for all the guidance; they have provided me throughout this project duration.

I would like to thank my parents and my sister for their lifelong guidance and steadfast support throughout my life.

I would also like to thank Aitezaz Ali and Peter Muir for helping me setting up Postgres Database, Danish Najam for giving me the idea of using Jane Hilston's simulator and helping me in formatting this report; Chris Ellis for helping in Java Swing, Shakir for giving me valuable advice on completing dissertation in time; Asher, Saqib, Muddassar, Fouad, Awais and Omair for their emotional support, emails and phone calls throughout the year.

Lastly, I would like to thank my best friend in Edinburgh, Salman Elahi for helping me get through the difficult times, for all the emotional support and care he provided and for

all the discussions, walks, dinners, laughs we had together. Life at Edinburgh would not have been the same without him.

Declaration

I declare that the thesis was composed by me, that the work contained herein is my own except where explicitly stated otherwise in the text, and this work has not been submitted for any other degree or professional qualification except as specified.

(Zeeshan Pervez)

Dedicated to My Ammi & Abbu

Thanks for being the greatest parents of this world!

Table of Contents

Table of Figures

List of Tables

Chapter 1

1.Introduction

The exponential growth of Internet has immensely changed the availability of electronically available information. Currently there are 600 billion static pages in the WWW and the growth rate is 10,000,000 pages per day [30]. However this success has made it very hard to find and use the information required by different users. Since it is not possible for any user to examine all the web pages, search engines are there to do this work.

Search engines use spiders, which roam around the Web, extract important textual information from web resources and build a huge index correlating keywords to web pages. When user enters a query, these search engines actually search in their index database, located on centralized server instead of searching the Web. It can be possible that some of the indexed web pages have been changed or even removed. So, to tackle this problem, the index database needs to be updated continually to maintain high quality search results [31].

Searching a specific piece of information in billions of web pages is a compute intensive task for centralized search engines. For this, they need to have complex and expensive hardware as well as efficient strategies to reduce response time. Dedicated and powerful processing machines are also required both to feed web addresses to spider programs for crawling and to process the data returned by these programs. On the software side they need highly customized compression and hashing algorithms to reduce storage cost and ensure efficiency and speed [31].

Some of the disadvantages of these centralized search engines are (i) they are expensive to build and maintain (ii) they poorly handle web pages whose content changes rapidly (iii) they even do not know about large amount of those web pages which cannot be reached by spider programs and (iv) their referenced information can quickly go out of date. All of these problems are also increasing as Internet continues to grow, making it impossible for any centralized search engine to regularly visit and index all web pages

[31].

Peer-to-peer (P2P) systems are distributed systems in which all the peers are equal and there is no centralized control. In P2P, peers act as both clients and servers, form an application-level network, and route messages e.g. requests to locate a resource [4]. P2P paradigm is best suited for information retrieval. In this case instead of a centralized index database, indices are maintained at each node. These indices are small, updated regularly and point to current information.

Scalability is an important issue to be considered in these systems. P2P systems which broadcast queries to all the peers in the network are not scalable. In order to perform smart search, we need intelligent query routing mechanism to be able to route queries to relevant subset of pairs. Query routing protocols are of the highest importance in P2P systems [4]. Unfortunately current techniques tend to be inefficient as they generate too much load on the system by broadcasting queries to all the peers in the system.

The notion of agent has been introduced with a two-fold purpose: i) a new paradigm for the modeling and implementation of complex software systems and ii) a way to include in a computational abstraction some behaviors which are usually associated with living beings, such as the capability to take autonomous actions in a specific context/environment in response to other agents activities and/or based on design objectives [5, 6].

Agents and P2P systems are complementary concepts as agents can drive the coordination between peers in P2P systems by residing in each peer [8]. Agents can initiate the task on the behalf of the peers e.g. an agent can receive a request from the user and it searches in its own local information repository/knowledge base present at the peer at which it is residing, or if it cannot find the answer, it can forward the query to one of the agents that it believes can answer the query. Multiagent Systems (MAS) can be thought of as network of equal peers. Agents in MAS give the same kind of abstraction as peers in P2P systems. In the next generation P2P systems, one might wish to have typical properties associated with agency (e.g. autonomy, reactivity, pro-activeness etc.) [32].

Lightweight Coordination Calculus (LCC) is a coordination language between agents.

“The most basic behaviors are sending a message and receiving it, where sending a message may be conditional on satisfying a constraint, and receiving the message may imply constraints on the agent accepting it” [9].

Query routing is a process of identifying the peer which is expected to contain the answer of the query and passing the query to that peer. Effective query routing not only minimizes the query response time and the overall processing cost, but also eliminates the unnecessary communication overhead that are inherent in the global networks and also over the individual information sources. Currently, a query routing mechanism does not exist in LCC. This limits its use in the communication where information is distributed across various sources.

This project is aimed at the development of query routing mechanism in an agent based P2P system, which will make use of LCC protocols. In this system, one agent would be present at each peer and act on the behalf of that peer. Every agent performs its tasks autonomously without any centralized control. Users give query and select their desired settings for query routing. LCC protocols for different routing mechanisms have been written and implemented. These routing mechanisms are compared with each other. Finally this system is compared with intelligent agents system [33].

This research is very closely related to SWAP project [40]. We have used the idea of semantic expertise advertisement and agent's expertise calculation from their work after necessary modification. However we have developed LCC protocols, query routing algorithms and query relaxation algorithms of our own.

This thesis is structured as follows:

In chapter 2, background knowledge for this project has been described. Agents have been introduced; their types and environments are discussed. Multiagent systems and coordination languages of multiagent systems are described in detail.

Chapter 3 describes the related work in the area of query routing and distributed dialogue protocols.

Chapter 4 covers the system overview, a complete example of system working, types of

messages used in this project, query routing mechanisms and agents roles.

Chapter 5 is a description of system architecture, components of the system, query relaxation, search algorithm and duplication filtering algorithm.

User interface is explained in chapter 6 with diagrams.

Chapter 7 describes the tests which were performed to evaluate the protocols.

Chapter 8 covers the comparison between simple agents with intelligent protocols and intelligent agents.

Chapter 9 summarizes our conclusion.

Chapter 10 identifies topics of further research in this area.

Chapter 2

2.Background

In this chapter, concepts of agents, multi-agent systems and agent communication are explained with examples.

2.1.Agent

An agent is a software program which is autonomous and interacts with its environment with the help of its sensors and actuators. Generally some sort of reasoning is performed to decide about the behavior of the agent. Input to agent is given through its sensors and the agents give output through its actuators. Agents produce appropriate output on the basis of given input [10]. An agent autonomously acts on the behalf of its user, has a set of objectives and takes actions in order to accomplish these objectives [14].

2.2.Agent Characteristics

Some important agent characteristics are as follows [13]:

Autonomy: Agents have control over their internal state and actions and they can perform their tasks without being intervened by humans.

Social ability: Agents can cooperate with humans or other agents to perform their tasks.

Reactivity: Agents can perceive their environment, and respond to changes that occur in it.

Pro-activeness: Agents not only act in response to their environment but also take the initiative to achieve their goals.

2.3.Agent Classifications

Agents can be classified into five groups based upon their functionalities [10]:

2.3.1.Simple Reflex Agents

This is the simplest type of agent. These agents do not take percept history into account for deciding about their actions. In other words these agents do not have any memory. According to current situation of environment, they find a rule in their perception and then perform actions which are associated with those rules. These agents work only if the environment is fully observable [10].

2.3.2.Model Based Reflex Agents

This type of agent has internal state, which acts as a memory and keeps track of previous events in their life. These agents find a rule which best matches their current situation, not only by using perception but also their stored internal state and then perform the action associated with that rule. The internal state of the agents helps them make a better selection of the rule to apply [10].

2.3.3.Goal Based Agents

This type of agent has explicit goals and performs actions to achieve its goals. In deciding about actions, these agents also perform planning in which they evaluate different possibilities and then chose the one which is closer to their goals. In general, they are more flexible than simple reflex agents and are capable of achieving their goals. If the world is changing, even then these agents have the same goals and they try to reach their goals [10].

2.3.4.Utility Based Agents

This type of agent has a utility function that maps agent states to a real number that describes the degree of how well the agent is performing. The decision about actions of agents depends upon this utility function. If the agent have conflicting requirements i.e. only some of them can be achieved (e.g. safety vs. robustness), utility function provides the appropriate trade off between them [10].

2.3.5.Learning Agents

This is the most sophisticated type of agent. This type of agent evaluates its performance and changes its behavior according to that. It also explores new behaviors and tries to find the best possible behavior for itself so that it can perform better in every possible

situation [10].

In this project model based reflex agents were used because if an agent receives some query again, which it has already answered, then it will use its knowledge base to find out which is the most appropriate agent to answer the query based upon the information which is available regarding its known agents and their expertise.

2.4.Agent Environments

Russel and Norvig [10] classify agent environments into six dimensions. Those six dimensions are:

2.4.1.Observable

If at every time, sensors of the agent can access the complete environment, the environment is said to be fully observable. Otherwise the environment is only partially observable. In case of fully observable environment, agents do not need to maintain internal state of the world. Reasons for partially observable environment can be noise, inaccurate sensors or missing state data from sensors [10].

2.4.2.Deterministic

An agent environment is said to be deterministic, if there is no uncertainty in it and new state of the environment can be completely determined by the current state and the actions performed by the agent. In this type of environment, agent does not need to worry about unexpected behavior of the environment, in result of its actions. If the agent is not sure about the state of the world its actions will result in, then the environment is said to be stochastic [10].

2.4.3.Episodic

In an episodic environment, the choice an agent makes is only based upon its current perception and has no dependency on its previous perceptions. If previous perceptions are also considered in taking a decision, then the environment is called sequential. An example of sequential environment is games, where each move by player is a part of his/her strategy to win the game [10].

2.4.4.Dynamic

If the environment can change in between the time an agent receives input at its sensor and when it performs the action, then the environment is dynamic. In static environments an agent can take as much time as it needs [10].

2.4.5.Discrete

In a discrete environment, there are fixed and finite number of possibilities for percepts and actions. If this is not the case then the environment is said to be continuous. In other words discrete environments fall into categories while continuous environments do not have categories but they have ranges for perception [10].

2.4.6.Agents

If there is only one agent in the system, then it is a single agent environment, otherwise if there are more than one agents involved, the environment is said to be multi agent environment [10].

After defining and explaining all types of agents' environments, we can classify the environment in which our case is in: multiple agents coordinating with each other to answer user queries.

The environment of this project is multiagent environment as it involves more than one agent. It is static because the number of agents and their knowledge bases are kept constant. The environment is partially observable because if the agents do not store the expertise description of other agents they would not be able to retrieve this information later from the system. The environment is stochastic because the agent which receives the query from a user does not know whether those agents to which it is passing the query will be able to answer the query or not. The environment is episodic as current query processing is being affected by previously asked queries and expertise of those agents which answered those queries.

This environment is in one of the hardest categories of agent environments as it is partially observable and stochastic.

2.5.Why to use Agent based approach

There are many reasons to use an agent based computing approach. Some of them are as follows [13]:

- If the components of the system change over time or not known in advance, an agent based approach allows developing such systems which are flexible, robust and they can adapt to the environment by using their negotiation ability and their social skills.

- In the case of domains where data and expertise are distributed, an agent based approach is more natural.

- An agent based approach is more useful in non deterministic cases where it is impossible to predict in advance that how the system would exactly behave. Agents can learn and change their behavior according to the requirements.

2.6.Multiagent Systems

Multiagent Systems (MAS) are defined as " the subfield of Artificial Intelligence (AI) that aims to provide both principles for construction of complex systems involving multiple agents and mechanisms for coordination of independent agents' behaviors" [11].

A MAS consists of multiple agents that can collectively solve problems which are beyond the capabilities of individual agents. In MAS agents coordinate to accomplish unsupervised actions.

MAS normally assumes a decentralized control. Message passing is used for communication purposes among agents. The key advantages of MAS are that each agent in MAS can be developed easily and independently as long as it is developed by following the specifications so that it can communicate with other agents in the system. MAS are also scalable as agents can easily be added into it as compared to changing the whole system every time.

MAS is an ideal solution for large and complex problems with more than one problem

solving entities and methods. In MAS, this type of problem is divided into number of small problems and each agent in the system is responsible for solving only part of the whole problem. Solving the complete problem is beyond the capabilities and knowledge of each problem solver.

Following are the characteristics of MAS [14]:

- There is no centralized control
- Each agent has incomplete information or capabilities to solve a problem.
- Computation is asynchronous.
- Data is decentralized

The interaction of agents in MAS is shown in Figure 1 [14]:

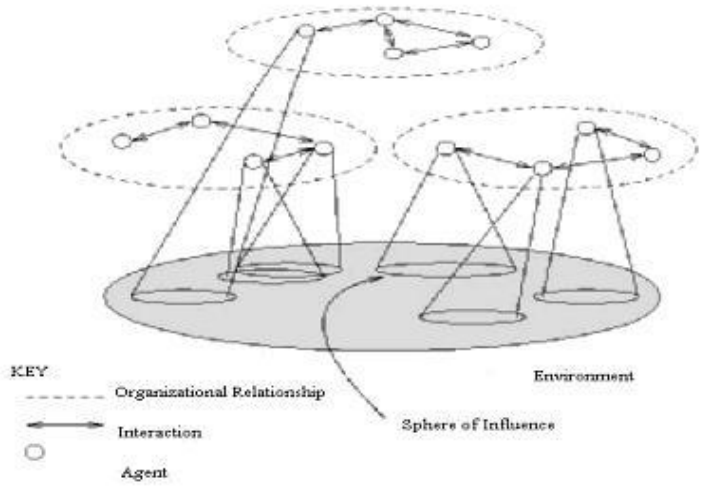


Figure 1 Multiagent System and its Interactions

Although agents are autonomous in MAS, yet MAS is viewed as a society. This society sets the principles for agents to communicate and coordinate with each other and achieve the common goal. An auction house can be seen as an example of MAS. The rules and regulations of auctions are defined and each agent in the system must follow them. In this case, there are two possible roles; an agent can be a bidder or an auctioneer. Auctioneer wants to sell the item for maximum possible price where as each bidder wants to acquire the item for lowest possible price.

2.7.Multiagent Coordination

As described earlier, an individual agent is not capable of solving a complex and large problem on its own. It needs to interact with other agents. In particular if agents are designed by different developers, then they must have some common conventions in order to understand each other. Collections of such conventions can be represented in an agent protocol. The most common method for communication between agents is sending and receiving messages. An agent protocol defines types and order of messages along with the conditions in which agents will send and receive the messages. Different protocols will be required for different scenarios e.g. Protocols describing two different types of interactions will be different [45].

An agent protocol does not describe how the agents interact. KQML/KIF [15] and FIPA-ACL [16] are two standard communication languages in MAS domain. In these languages, agents communicate with each other by passing messages. Internal implementation of agent and its decision making abilities are not defined in these languages. These issues are left up to the individual implementation of agents.

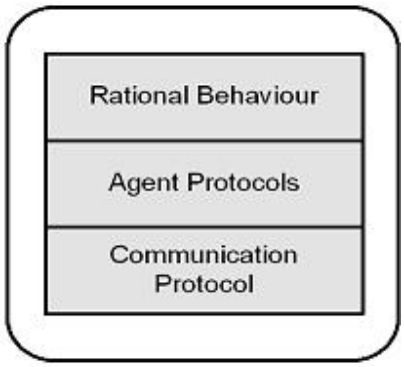


Figure 2 Agent Stack

Agent protocols come in between the top layer, which defines the rational behavior of an agent, and the communication protocol layer, which defines how the agents will communicate as shown in Figure 2 [42]. An agent protocol is just a predefined plan of how the agents will collaborate in a given situation.

2.7.1.Electronic Institutions

Electronic Institutions (EI) is a way of providing communication in MAS, between agents which have been implemented in different ways. EI have been designed by following the conventions of human organizations. Different institutions exist in human societies and in these institutions, responsible authorities set laws, act upon them and monitor others to do the same [17].

An EI is a formal framework for an open multiagent system. It requires agents to interact in a predefined manner and follow the commitments and obligations of their roles. Agents in an EI can be viewed as actors. Every actor within the institutions can adapt different roles [17].

Each role has dialogic actions associated with it. After adopting a role, an agent can perform those actions which are associated with that role. Agents interact with each others in group meetings which are called scenes in EI. A scene contains a well defined protocol which is known as a script. A script contains all the possible dialogues, which agents in that scene might have. All the agents that are present in that scene follow the same script [42]. “These scenes can be connected and compose a network of scenes which is called performative structure. Performative structures capture the relationship between scenes and it also contains description that how different roles change as agents move between scenes” [43].

This is illustrated in a simple example of auction scene. In this scene, there are four agents. Three of them are in the role of bidder, while the forth have adopted the role of auctioneer. The agents who have adopted the role of bidder can perform the dialogic action “bid” where as the agent who is in the role of auctioneer can perform two dialogic actions, “Invite bids” and “Sold”. In agent protocols, it is defined that how two groups of agents with different roles can communicate with each other [42].

Agents also require common knowledge to interact with each other. This knowledge is represented in a dialogic framework. The structure of dialogic framework allows heterogeneous agents to exchange knowledge. Illocutions are defined by the institutions through common language (ontology). Agents interact with each other by sending and receiving messages. By observing human conventions, EI have been able to deal with

the complexities of open multiagent systems like trust, heterogeneity of agents and social change etc [42].

2.7.2.Problems with Electronic Institutions

Although EI produces a framework for agent coordination in MAS and solves many related issues but it also has some significant problems in it. The following are some of the problems in EI, due to which it has not become a true standard of the agent community [17, 42, 43]:

- One of the main problems is that all the interactions are coordinated through a central agent known as administrative agent. The administrative agent is responsible for making sure that agents adhere to institution's regulations. The administrative agent is the bottleneck in the system. The correct functionality of the whole system is dependent on it. If it does not perform correctly or crashes then all the agents in the system will be unable to behave correctly. In the presence of an administrative agent as central body, agents are no longer autonomous; hence they undermine one of the key properties of agency.

- Another problem is the scenes in which the agent will be taking part must be determined before the agents are deployed. Due to this, if a new agent wishes to participate in an institution, it needs to know all the internal details of the institution and follow the design of the institution. The current approach is that path of the agent needs to be pre-determined. This approach is fine with simple and predictable interactions. The problem comes in large and complicated cases when it is not possible to predetermine the path of an agent through institution.

- Thirdly, it is assumed that the topology of the institution will not change. If the institution definition changes, then the individual agents will need to be re-synthesized and corrected. Thus a small change in the definition of an institution may end up complicated reworking of all the individual agents which is not desirable and make it difficult to deploy.

2.8.Distributed Dialogue Protocols

Distributed dialogue [18] is an interaction among group of agents. It consists of dialogue sequences between agents. It is built on the basis of EI, however it provides an enhanced mechanism for ensuring that the agents are truly autonomous and there is no central agent. In MAS, agents coordinate by sending and receiving messages to other agents. This message behavior can be described in a notation which is very similar to Calculus of Communicating Systems (CCS) [19]. CCS is a process calculus which is used to formally describe concurrent and communicating processes [18].

Consider an example of two agents. First agent a (r1, a1) wants to send a message m1 to second agent a (r2, a2). Where r1 and r2 are the roles of agents and a1 and a2 are the unique identifiers of the agents. After receiving message m1, agent a (r2, a2) will send reply message m2 to agent a (r1, a1). Assuming that each agent operates sequentially, set of possible dialogues between two agents can be represented as follows [42]:

For a (r1, a1): m1=> a (r2, a2), m2 <= a (r2, a2)

For a (r2, a2): m1<=a (r1, a1), m2 => a (r1, a1)

In the above protocol => denotes message sending and <= denotes the message receiving.

Two languages have been developed on the basis of distributed dialogue protocols: the Light Weight Coordination Calculus (LCC) and the Multi Agent Protocols (MAP). However these are similar languages from the same research group. For the purpose of this project LCC was used.

2.8.1. Lightweight Coordination Calculus

LCC is an extension of EI and it defines the scenes in the framework in very innovative way. It does not require any centralized agent for coordination between agents. Due to this, agents have been able to preserve their autonomy property. LCC has been formally defined by using light weight formal methods. The abstract syntax of LCC is presented in figure 3 below [9]:

Framework := {Clause, …}

Clause := Agent :: Def

Agent := a (Type, Id)

Def := Agent | Message | Def then Def | Def or Def | Def par Def

Message := M => Agent | M=> Agent ← C | M <= Agent | C ← M <=Agent

C := Term | C ∧ C | C ∨ C

Type := Term

Id := Constant

M := Term

Figure 3 LCC syntax

An agent A is defined by a term A::D where D is the definition which defines the messages, an agent is allowed to send. Different operators can be used to construct D. A message is only sent if the constraint associated with it is satisfied and this type of constraint is known as a proaction constraint. Similarly a message is received only if the constraints implied by the message are accepted. This type of constraint is known as a reaction constraint. Complex behaviors can be specified using the connectives then, or and par which denotes sequence, choice and parallelization respectively. With the help of constraints on messages, an agent can interact according to social norm while maintaining as much as possible of their autonomy [44].

Agents also share certain amount of common knowledge. An example of this could be current reserve price of an item taking part in auction [42]. As an example in LCC, we take the scenario of interaction between two agents in which one of them is in the role of bookseller and other is in the role of inquirer. An inquirer agent asks the price of book X from bookseller agent. The bookseller will send a reply in a message containing price Y of book X. Below is the LCC version of this scenario:

a (inquirer(X), A):: ask (price(X))⇒ a (bookseller, B) then

offer (price(Y)) ⇐ a (bookseller, B)

a(bookseller, B) :: ask (price(X)) ⇐ a (inquirer(X), A) then

offer (price(Y)) ⇒ a (inquirer(X),A) ← knowsPrice (X)

As described earlier, in EI all the agents in a scene follow a global script which cannot be modified and all the agents participating in the scene must stick to it. LCC on the other hand takes a different approach. There is only one copy of script which is shared between all agents participating in a scene. This script is passed from one agent to another during the execution of protocol. Hence there is flexibility in LCC that script can be extended as the conversation of the agent progresses [45].

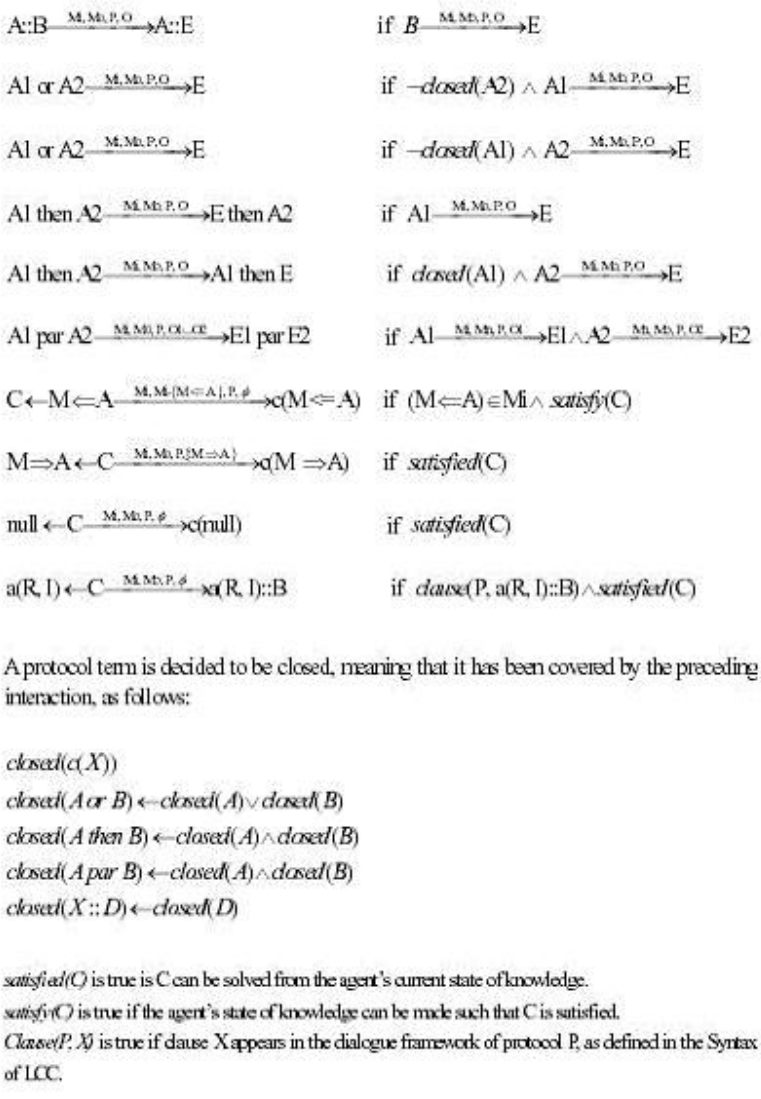
In LCC when an agent communicates with other agent (who is unaware of the first agent), the framework allows the first agent to inform second agent that how it is intended to communicate with it. Every agent can also determine the current state of dialogue. This is achieved by separately maintaining the instances of dialogue clauses used by each agent participating in the dialogue [44].

With every message received by the agent, a protocol is attached. The protocol is of the form as illustrated above. The agent updates the protocol it receives and finds the next move it is allowed to take. After taking its move, it will update the protocol describing the new state of the dialogue. In simple words each time an agent performs an action which it has been asked to do by the script; it will update the script and pass the protocol on. In the same way when the next agent receives the protocol, it will exactly know what it is required to do because the preceding agent has marked the script, retrieving the current state and knows where to pick up where the last agent left off [42].

"This is done by applying rewrite rules. As an example an agent receives a message with an attached protocol, P, and extracts from P the dialogue clause, C_i, determining his part of dialogue. Rewrite rules are applied to give an expansion of C_i in terms of protocol P, in response to set of received messages, M_i. They generate a new dialogue clause C_n, an output message set M_n (a subset of M_i). These are produced by exhaustively applying the rewrite rules:

$$< Ci \xrightarrow{Mi,\ Mi+1,\ P,\ Oi} Ci+1,\ \ldots\ldots,\ Cn\text{-}1 \xrightarrow{Mn\text{-}1,\ Mn,\ P,\ On} Cn>$$

Agent clause C_i will be replaced by C_n and protocol P_i will be replaced by the protocol P_n. Now the agent will send messages accompanied by new protocol P_n" [9]. Rewrite rules are listed in Figure 4 [9]:



A protocol term is decided to be closed, meaning that it has been covered by the preceding interaction, as follows:

closed(σ(*X*))
closed(*A or B*) ← *closed*(*A*) ∨ *closed*(*B*)
closed(*A then B*) ← *closed*(*A*) ∧ *closed*(*B*)
closed(*A par B*) ← *closed*(*A*) ∧ *closed*(*B*)
closed(*X* :: *D*) ← *closed*(*D*)

satisfied(*C*) is true is C can be solved from the agent's current state of knowledge.
satisfy(*C*) is true if the agent's state of knowledge can be made such that C is satisfied.
Clause(*P*, *X*) is true if clause X appears in the dialogue framework of protocol P, as defined in the Syntax of LCC.

Figure 4 LCC Rewrite Rules

LCC works even in those cases where agents do not possess protocols before hand and where the number of agents participating in the interaction has not been predefined. As

autonomy is key to agency, in LCC this property is retained in agents as there is no centralized agent present to synchronize the coordination of agents. The protocol is also not required to be distributed to all agents which are going to take part in the scene in advance. This means that change can easily be made without modifying the protocol and having to redefine each and every agent. In short with LCC, agent definitions are dynamic, simpler and without the need for a centralized controlling agent.

2.9.Machine Learning Algorithms

In machine learning paradigm, two types of algorithms are generally discussed:

- Lazy Learning Algorithms
- Eager Learning Algorithms

2.9.1.Lazy Learning

In these algorithms, training samples are simply stored for future use. In these algorithms, no generalization of these algorithms is performed to generate any type of rules. The characteristics of these algorithms are as follows [39]:

- They do not perform any processing on the input until they receive a request for that information.
- They combine stored training data to reply queries
- They do not store any intermediate results.

Typical type of lazy learning algorithms are case based learning, example based learning and instance based learning. The term “lazy” is borrowed from lazy evaluation in functional programming languages [39]. As an example of lazy learning in agent based P2P system, suppose a query is made of some agent. The first agent which receives this query from the user will forward this to other agents in the network and receive results from them. Now instead of inferring any information from this, the agent will simply store this data against the query. If this agent receives the same query in future, it will search its knowledge base to find out which agent replied last time and then forward the query to that agent.

2.9.2.Eager Learning

Eager learning algorithms induce abstract concepts from the training samples e.g. using decision trees, neural networks etc. When eager learners receive requests for information, they make use of abstract information. As an example, when an agent receives an answer for a query from different agents in the network, it will store the expertise of agent in its knowledge base depending upon the related topic of the query instead of just storing the agent id and query. Whenever it receives the same query or a similar query, it can forward the query to appropriate agent based upon the subject of the query (cf. section 5.2.2).

2.9.3.Comparison of Eager and Lazy Learning

The main difference between both algorithms is that lazy learners perform little processing on training data and eager learners induce abstract rules from it. In eager learning algorithms, learning time is high while the query response time is low and vice versa in lazy learning algorithms.

- Lazy learning is useful when number of training data is small and query arrival is not frequent. However if there is large set of data, finding the appropriate agent is time consuming and requires smart indexing techniques.

- As eager learning algorithms discard training data and store only compact summary in the form of rules so they require quite small size of knowledge base as compared to lazy learners.

- Lazy learners can generate detailed explanations of its decisions about any particular request for information instead of abstract explanations which may be required for some tasks.

2.10.Peer-to-Peer Computing

P2P computing paradigm is very popular for sharing computing resources/services such as data files or processing cycles. In these systems very large amount of peers pool

together their resources and depend upon each other for data and services. These systems are characterized by symmetric roles among the peers, where each peer in the network acts alike. Popular examples of P2P systems are Napster[1], Gnutella[2], Freenet[3] and Kaaza[4]. P2P computing can also be applied to other areas like Grid Computing (e.g. SETIHome[5]) and collaboration networks (e.g. Groove[6]).

Locating the peer which contains the resource/information required by the user is the key issue in P2P systems. There are two types of solutions for this. First is to forward the query to all the agents in the network. This technique is called "Flooding". Gnutella is based on this technique. Second technique is Distributed Hash Tables (DHT). DHT based schemes (e.g. Chord [41].) include peers into structured overlay networks and assign each data item to a specific peer. In this way each peer needs to know about O(log N) neighbors [32].

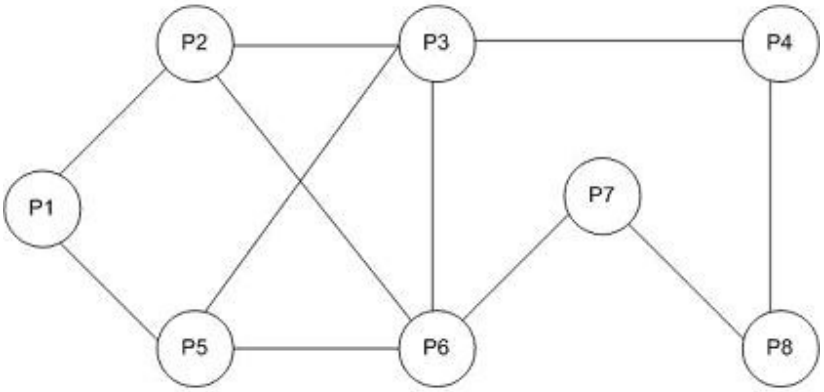


Figure 5 Typical P2P Network

None of these methods support semantic methods. They perform keyword based search. However, in this project we are using semantic similarity of agents' expertise (cf. section 5.2.1) and query subject (cf. section 5.2.2) in the process of identifying appropriate peers.

[1] http://www.napster.org
[2] http://www.gnutella.wego.com
[3] http://www.freenet.sourceforge.net
[4] http://www.kaaza.com
[5] http:// www.setiathome.ssl.berkley.edu
[6] http://www.groove.net

2.11.Hop Count

The number of agents/peers traversed by a message between its source and destination is known as a hop count. As an example Agent A is the querying agent, it forwards the query to Agent B, this makes hop count=1. Now Agent B sends the same message to Agent C, which will make the hop count=2.

Figure 6 Hop Count Example

Chapter 3

3. Related Work

We consider two areas of research related to our work. The first is distributed dialogue protocols and the second area deals with semantic query routing in P2P systems. In this chapter work on both of these research areas is described briefly.

As it has been discussed previously, that two languages emerged from distributed dialogue protocols namely LCC and MAP. LCC has been explained in depth in Chapter 2. In this chapter we will briefly look at MAP and its difference with LCC.

"The MAP language is a lightweight dialogue protocol language that allows designing a multiagent protocol in an easy and convenient way" [22]. In MAP every agent is defined by its role which contains different methods with input parameters. Every agent has a fixed role which cannot be changed for the duration of the whole protocol. The protocol describes who sends and receives messages along with their types. In MAP logical operations and procedures are also allowed. These procedures are used to check the constraints associated with the messages [22].

Although both LCC and MAP are built upon CCS, yet the execution of protocols in both languages are not the same. Agents' communication via MAP is reliable, buffered and non-blocking. In MAP a separate protocol is defined for each role in the scene. In addition to this, agents' decisions are also added to the syntax [42].

In MAP, all agents have their own copy of protocols before the execution starts. The agent just unpacks the protocol and follows the steps in the protocol. In MAP, there is no need to update the protocol as every agent has its own copy [22]. This is the fundamental difference between MAP and LCC, as in LCC there is only one global copy of the script and agents need to update it [42].

For query routing in P2P systems, the challenge is to find the peer that is expected to contain the answer of the query and passing query to that peer without involving

unrelated peers and keeping messages on the network as minimum as possible. This is very hot area of research and many research groups have come up with different techniques. Below is the brief analysis of different approaches of query routing and related work.

The Small-World Effect project [20] is one example how network and document distribution topologies can be used efficiently to establish a relation between two peers. In this project, contents of a peer are advertised to all of its neighbors. In this way whenever a peer joins or leave the network or it updates its contents, the network will be flooded with advertisement messages, which is a substantial overhead over the network.

Tempich et. al. [21] have used a lazy learning approach in their system REMINDIN' for query routing in P2P system. In this approach, instead of passing advertisement messages to all the peers in the network, peers observe the queries being asked on the network and their answers and store this information in their local repository. Next time if these peers receive similar queries, then they would exactly know which peers contain the answers of the query. So they will pass the query to exactly that peer instead of broadcasting the query or passing it to all neighbors. If a peer does not have information about the query and it also does not know which peer can answer this query then it will relax the query. They use the ACM Topic Hierarchy (cf. section 4.6) so if the peer is unable to decide on an appropriate peer then it will generalize the query subject using this hierarchy. They have claimed 80% recall after running 20,000 queries [21]. However this approach has two major shortcomings. First is it ends up replicating large subsets of foreign knowledge bases at every peer and secondly it requires to send a large number of queries to become efficient.

[12] presents schema-based P2P networks and the use of super-peer based topologies for these networks, in which peers are organized in hypercubes. [11] shows how this schema-based approach can be used to create Semantic Overlay Clusters in a scientific Peer-to-Peer network with a small set of metadata attributes that describe the documents in the network. In contrast, the approach in our system is completely decentralized in the sense that it does not rely on super-peers.

In [23] a query routing mechanism has been proposed based upon super peers. In this

system peers with the same topic are organized in a hypercube topology [26]. There is a two step routing mechanism involved in this setting. In the first step, queries are routed to the super peer back-bone and then it is distributed to those peers which are connected to super peers. The second step can be avoided if the super peer cache data from their connected peers. With this topology, each peer will be queried only once for a single query (which is still quite a big number of queries). They also have not provided any test on the performance of their algorithm. Another limitation of this approach is that it does not distinguish between more knowledgeable and less knowledgeable peers within the hypercube [26].

A different approach to finding appropriate peers in the network is social network analysis. ReferralWeb [24] make efficient use of this technique. ReferralWeb reveals the existing social network by data mining resources found on the web. These resources can be anything ranging from home pages, links to other resources, list of author and co-authors in a research paper, citation of a technical report to news archives and organizational charts [26].

Their experiments prove that this technique work even better in large networks. [25] Shows that precision of answers increases when these referrals are used. They have also shown the case where the peers with similar expertise are grouped close to each other according to their similarity function. They have claimed that number of messages per query passed on the network will decrease. However they have not proved this in their experimental results. They store the queries and expertise in vectors and calculate similarity based on taking the cosine of both vectors [26].

Bibster [27] is an application which is based on the approach of using semantics in P2P systems. This system is designed for exchanging bibliographic data among researchers. In this system each bibliographic entry is structured and classified into two ontologies SWRC [28] and ACM Topic Hierarchy [29]. In this system peers advertise their expertise to all of their known peers. If the received advertisement is related to one's known expertise then it will accept the advertisement, otherwise it will discard it. Different experiments have been performed in this project and they have proved that their technique works in different settings. However one shortcoming in this system is that querying peer does not store this information that which peer has replied about this

query, it just adds the replying peer to its known peers. If again it receives the same query, it needs to repeat the whole process instead of just passing the query to that peer who has replied last time.

Chapter 4

4.Project Overview

This project has several purposes:

- Enabling query routing in LCC (cf. section 2.8.1) by writing different query routing protocols in it.
- Development of an interpreter in Java that is able to understand the protocols.
- Testing the protocols in a simulation based environment to check if they are working correctly.
- Comparison of an MAS based upon agents with LCC protocols to an MAS consisting of agents which are intelligent themselves and do not need any protocols to guide them.

This project has been done as a part of joint research study carried out by two MSc students Zeeshan Pervez and Salman Elahi under supervision of Dr. Dave Robertson. We have tried to study that whether agents with little intelligence but with LCC protocols, can perform at comparable levels of query routing effectiveness as more intelligent agents. As query routing is of paramount importance in agent based P2P computing, so we have selected this feature and developed two query routing systems, one with intelligent protocols and other with intelligent agents. This report is related to intelligent protocols work but a brief summary of the intelligent agent system [33] is also included in this report for completeness (cf. chapter 8).

This project is developed as a simulation to test the effect of different input parameters (cf. section 4.11) in a controlled environment. There are 20 agents in the simulation and each agent knows only 4 other agents which means that every agent can initially communicate with only 4 other agents. These 4 agents are called known agents of that particular agent. These known agents are selected by a Java program which randomly selects 4 known agents for every agent.

4.1.Assumptions

The following are some assumptions on which this project has been based:

- The user of simulation is Computer Science researcher and not a naive user. He/She will neither give any invalid input nor misspell the words.

- The user is aware of ACM topic hierarchy (cf. section 4.6). He/She selects the related area of query by selecting related node from ACM topics tree in simulation. This selected area will be the closely related to the query. For no reason, he/she will select an area which is not related to the query.

- The user will write major part of title or important keywords of the bibliographic reference, he/she is looking for. As this system is not a search engine, so if the user pastes some lines of text, he/she might not be successful as full text search facility is not provided in this system.

- Network related issues like network topology and physical connection between peers are not discussed in this project. It has been assumed that communication is reliable. Messages sent over the network are always received. The message content does not change on the medium. Messages are only received by the intended receivers and no unauthorized agent has access to the message contents.

- Peers/Agents are also assumed to be reliable and they will not fail during the simulation.

- The Number of agents is fixed in the simulation and it cannot be changed. No peer can leave or join the network. The number of bibliographic references and agent's expertise are also assumed to be fixed and they do not change.

4.2.Working Example

Before moving into technical details about the project, one complete working example of the system is explained below in simple terms:

When the simulation starts, every agent in the network will send advertisement messages (cf. section 4.3.1) to all of its known agents (which are 4 in our case). The advertisement

message consists of description of expertise of the sender. From these messages, every agent knows the expertise of all of its known agents. This information is required when every an agent needs to forward the query to its known agents. Instead of forwarding the query randomly or broadcasting the network, the agent can select the most appropriate agents which are likely to answer the query.

The user selects an area from ACM expertise tree which is closely related to his/her query. On the basis of this selection of user, the first agent (the querying agent which will receive the query before anyone else) will be selected.

The user will select one of the settings from the setting combo box. Each setting includes different combination of input parameters (cf. section 4.11), message passing (cf. section 4.3) and peer selection algorithms (cf. section 5.3).

The user will enter the keywords on which he/she wants to perform a search and press the search button.

The simulation will select the first agent on the basis of closest area identified by the user. This first agent will receive keywords entered by the user. This agent will perform a local search into its knowledge base and retrieve all the results (if any), it finds relevant to the search.

Now depending upon the setting selected by the user, the querying agent will select the best agents (other than the naïve settings (cf. section 4.10), in which the query is forwarded to the randomly selected known agents) and forward the query to those agents. In the semantic settings (cf. section 4.10), these agents are selected based upon the expertise sent in the advertisement messages.

When the intermediate agents (cf. section 4.12) receive the query message, they will look into their knowledge base and try to find the relevant answers. If they find the relevant results, they will reply to the sender of query message (which will be querying agent itself in this case) with answer message containing the relevant results of the query. If these agents are unable to find any relevant result in the knowledge base, then they will not send any message to the querying agent. Similarly if the querying agent does not receive any message from the agent to whom it has forwarded the query, it will

assume that the agent does not know about that query. This process of forwarding the query to next agent continues till hop count (cf. section 2.11) is not reached to its limit, which is set to n=2 for the purpose of this project. This essentially means that querying agent will forward the query to its known agents that will make hop count=1 and these agents will forward the query to their known agents which will make hop count=2.

The above process is shown in Figure 7:

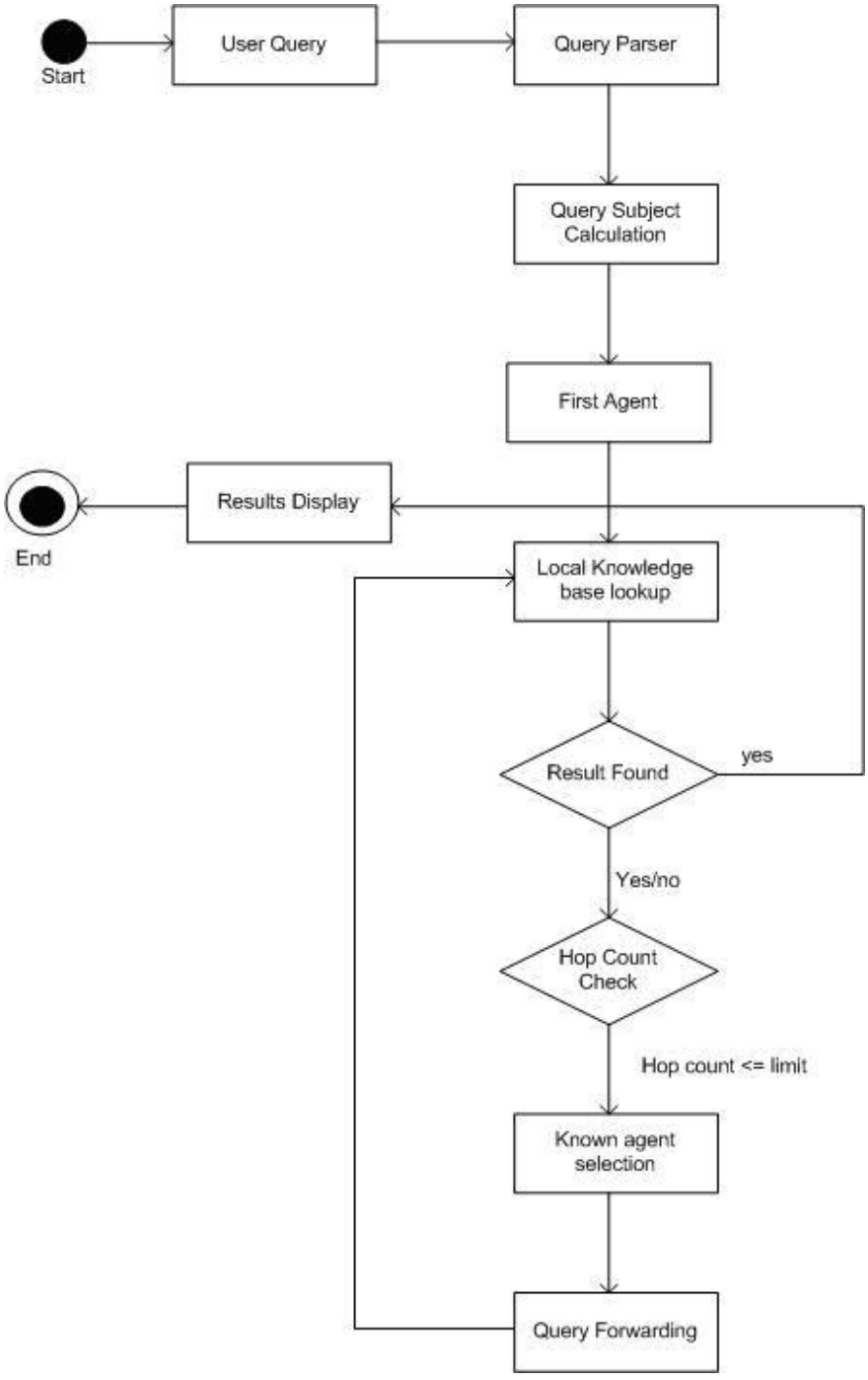


Figure 7 Process Diagram

In chapter 5, all the above system modules are explained in detail.

4.3.Types of Messages

In this simulator, the following four types of messages have been used:

4.3.1.Advertisement Message

The purpose of this message is to inform the known agents about the expertise of the sender agent. As the number of known agents is fixed to 4 in the simulation, so every agent will send advertisement message to all of its 4 known agents. At the same time, it will receive an advertisement message from all of its known agents.

4.3.2.Query Message

When an agent receives a query from the user, it forwards the query to its known agents based upon the setting selected by the user, so it will send a query message. When an agent receives a query message, it will try to answer the query by looking into its knowledge base. If the hop count (cf. section 2.11) of that query has not reached its maximum limit (which is n=2 in our case) then the agent will forward a query message to its known agents based upon the settings.

4.3.3.Answer Message

When an agent receives a query message, it will look into its knowledge base and try to find the relevant answers for the query. If it finds the answers then it will send an answer message to the sender of the query message, containing the answer of the query. In that case when an agent receives a query message but it does not find any relevant results in its knowledge base, then it will not send any message to the sender of query message. If the sender does not get any reply then it will assume that the receiver of the query message has no information about the query.

4.3.4.Re-route Message

This type of message is used in the overt routing mechanism (cf. section 4.4.1). In this case if intermediate agents (cf. section 4.12) do not have the answer of the query, then instead of sending messages directly to their known agents, they will send re-route message to the querying agent. The purpose of this message is to introduce querying agent and known agents of intermediate agents so that they can communicate directly without the involvement of any middle agent.

In this project as we do not deal with network related issues, so network communication

is assumed to be reliable. Messages neither are lost nor are they forged. So, there is no need of acknowledgment or negative acknowledgment messages. It is always assumed that messages sent are always received.

4.4.Routing Mechanism

As query can be sent from one agent to other agent in different possible ways, the following three mechanisms are used in this project:

4.4.1.Overt

In this mechanism, every agent will only communicate with the querying agent or the first agent which receives the query from the user. The querying agent will pass the query to its known agents. Now if those agents know about the query, they will send an answer message to the querying agent, otherwise they will send the re-route message to the querying agent. The querying agent will check that if the hop count of that query has not exceeded its maximum limit (which is n=2 in our case), then it will forward the query to the known agents of its known agents.

As an example, Agent A is the querying agent; it receives a query from the user. It will forward the query to agent B which is its known agent. Now agent B will look into its knowledge base. If agent B cannot find any answer, it will send a re-route message to agent A, which contains information about agent C, which is the known agents of agent B. After receiving re-route message agent A will check that if the hop count (cf. section 2.11) of this query has not reached to its maximum limit, then it will forward the query to agent C, otherwise it will only display results it has received till now. This process is shown in Figure 8.

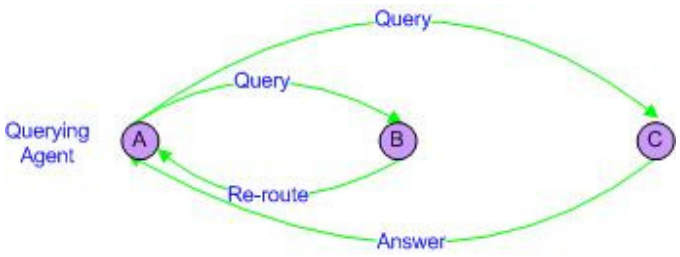


Figure 8 Overt Routing

The overt routing mechanism is expensive in terms of query and re-route messages, as every agent has to communicate with the querying agent. Number of messages can be reduced if intermediate agents start sending messages to their known agents themselves, without involving querying agent every time. At the same time, this technique is efficient in answer messages. As every agent is communicating with the querying agent which actually needs the answer of the query, so the answer message is received at the querying agent in just one message.

4.4.2.Covert

This is a more advanced form of routing mechanism as compared to the overt. Intermediate agents can communicate with each other along with the querying agent and they do not need to send re-route messages to the querying agent. They will directly send the query message to their known agents. In this case, an answer is always sent to the sender of query message, instead of the querying agent or any other agent. The receiver of the answer message will check that if it is not the originator of this query, then it will send the answer message to that agent which sent it the query message. In this way messages are kept on forwarding to the known agents until the hop count limit does not reach its maximum value (which is n=2 in our case). Every agent sends the query message to its known agent and an answer message to the sender of the query message.

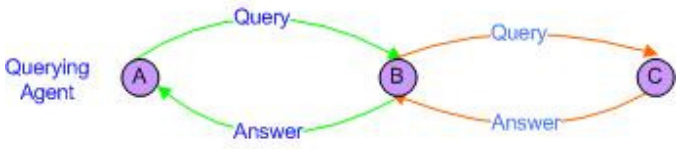


Figure 9 Covert Routing

As an example, agent A receives a query from the user. It looks into its local knowledge base and then forwards the query to agent B which is its known agent. The hop count for this query has reached to 1 till this point. Now whether agent B, finds results from its local knowledge base or not, will forward the query to agent C which is its known agent. Hop count has reached to 2 which is the maximum value set for this project, so if agent C finds the relevant results, it will send the answer message to agent B, otherwise it will

not do anything. This process is shown in Figure 9.

The covert mechanism is more parsimonious in terms of query messages, as intermediate agents send the messages directly to their known agents, instead of passing them to the querying agent every time. In this way the total number of messages sent on the network is reduced. At the same time they are not very good in terms of answer messages. As in overt, all agents receive messages from the querying agents, so they send answer messages directly to querying agent but in covert agents receive messages from their known agents, so they send answer messages back to the sender of query messages, In this way, answer messages follow the path of query message in the opposite direction and finally reach the querying agent.

4.4.3.Hybrid

This is the most advanced form of routing mechanism. It is basically a mixture of both approaches i.e. overt and covert. It is more efficient than both above mentioned techniques as it floods the network with the least number of messages. In the hybrid mechanism, query messages are sent following the covert mechanism and answer messages are sent following the overt mechanism. This means that query messages are sent directly to the known agents of intermediate agents and the answer messages are sent directly to the querying agent, instead of sending it to the immediate sender of the query. As this approach is taking the best of above both approaches, it involves the least number of messages on the network.

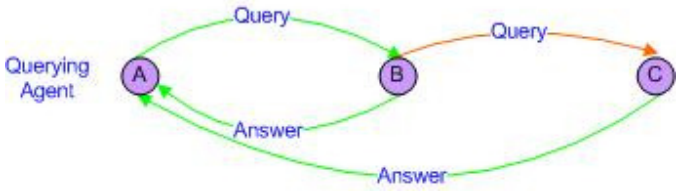


Figure 10 Hybrid Routing

As an example, agent A is the querying agent. It forwards the query to agent B. First agent B will look into its local knowledge base, if it finds relevant result, it will send the answer message to agent A. As the hop count has not reached its maximum value (which is n=2 in our case) so B will directly forward the query to agent C without involving

querying agent. If agent C finds the relevant results for the query, it will send the answer message back to the querying agent A, instead of passing it to their immediate senders of query message (i.e. agent B). This process is shown in Figure 10.

4.5.Simulator

For the simulation purposes, a network simulator developed by Dr. Jane Hilston and her PhD student Yussuf Abu Shaaban has been used. This simulator has been developed in Java. One of the reasons for using this simulator is we wanted to use Java instead of Prolog as an LCC interpreter in Prolog has already been developed. We wanted to study LCC in some object oriented programming language like Java.

This simulator has many features that exactly matches our requirements and best suited for this project. The simulator allows the user to generate a network by creating peers. The user can store the network and use the same network next time by loading the already stored network.

It uses an XML file to store the information about the network. In that XML file the number of peers in the network along with id, color, failed status, x-coordinate and y-coordinate of every peer is stored. Every peer in the network has a unique id and is shown by a blue circle on the simulator except for the first peer which has yellow color. This difference in color is made to distinguish the first peer from the others. To connect two peers, the user needs to left click on the first peer and then left click on the other peer. In order to demonstrate peer failure, the user press the centre button of the mouse on that peer and a red cross will appear on that peer to show that the peer has failed and can longer work. When a peer fails, its failed status is set to true in XML file, which is false otherwise.

The message passing behavior between the peers is demonstrated by drawing a yellow rectangle at the sender and then after a second or two that rectangle will disappear from the sender and appear at the receiver peer, which basically shows that the message has been sent to receiver from the sender. This rectangle contains three pieces of information namely type of message (e.g. query message or answer message), sender id and receiver id.

However for the purpose of this project, many changes were made to this simulator. Now in this project, the first agent is not selected by the user; it will be selected by the simulation based upon the ACM topic area selected by the user. This is done because we want to pass the query to that agent which is most likely to answer the query instead of just selecting a random agent by the user. As we are not dealing with agents' failure so this facility is no longer available to the user. We are assuming that only known agents can communicate so user is not required to draw the connections between agents. In our case when ever the user starts the simulation, a network of fixed agents (which is set to 20, other than the scale up experiments) is displayed in the simulator. The user cannot add or delete agents or make any changes in the network.

We have added many features in the network which were not initially present in the network:

A notice board has been added to the simulator. In that message board, all the messages which are sent across the network are printed. Initially the user has only one method of determining that which agent has sent the message and which agent is the receiver of the message and that was by looking at the message rectangles. When multiple agents simultaneously send and receive messages of different types, then it is very different for the user to understand what is happening in the system by just reading the text in the rectangles. If an agent receives two messages, then the first message rectangle will be overdrawn by a second message rectangle, so if the user misses that, he/she is unaware of the first message, but now, with the help of message board, the user can read all the messages even at the end of simulation and no message disappears until the user enters another query to search. In order to graphically demonstrate the results, a table has been added to the simulator. After receiving the answer message, the querying agent will retrieve answers from it and display them to the user in that table. In this table we present to user the id of bibliographic reference, its title, authors, year of publication and its relevance with the keywords entered by the user. The user can get the complete details of bibliographic reference by a mouse click on that particular record and a new window will appear which contains the complete reference in Bibtex format. An example of Bibtex entry is :

@article{1058005,
author = {Cirrus Shakeri and David C. Brown},

title = {Constructing design methodologies using multiagent systems},
isbn = {0890-0604},
pages = {115--134},
publisher = {Cambridge University Press},
address = {New York, NY, USA},
}

The ACM topic hierarchy is also given to user in a tree format. The user is required to expand the tree and find the most relevant area related to his/her query. This selection helps the simulator in selecting the most appropriate agent as the querying agent. This selection plays an important role because if an agent with a different expertise is selected, then the user might not be able to get relevant results.

4.6.ACM Topic Hierarchy

When a user enters a query, we need to find the topic of that query i.e. related area of that query so that most suitable agent can be selected as the querying agent. In order to find the relations between different topics, we are using ACM topic hierarchy. This topic hierarchy is called ACM Computing Classification System (CCS). This classification is useful to readers as it provides the quick content reference for relevant literature search. It also makes search easy in the ACM Digital Library and other ACM resources.

"CCS is a four level hierarchy of terms that has three coded levels and an uncoded level of subject descriptors" [36]. Subject descriptors are generally at the fourth level. This hierarchy contains both categories as well as subject descriptors. The tree classification is restricted to three letters and numbers coded levels. Categories are assigned a single alphabet at the first level while sub-categories are assigned alpha numeric codes [36]. As in example:

I Computing Methodologies (First-level node number and title)

I.2 Artificial Intelligence (Second-level node number and title)

I.2.11 Distributed Artificial Intelligence (Third level node number and title)

* *Multiagent Systems* (subject descriptors)

This tree consists of 11 first-level nodes. In addition to that there is a set of 16 separate concepts called “General Terms” that apply to all areas. Language theory is an example of these terms [36].

4.7.Scenario

In order to simulate and test our system, we have selected a bibliographic references scenario:

In the daily life of a computer scientist, one has to look for publications related to their research interests. Most common methods for performing these searches are with search engines like Google and Yahoo, via libraries or simply asking those people who are likely to know the desired information [26]. As publications are increasing day by day, this problem is also increasing. We present a P2P system in which on every peer, there is an agent which can communicate on the behalf of that peer. Every agent is basically representing a researcher. Each researcher has some bibliographic references which he/she wants to share with others on the network. These bibliographic references are classified against the ACM topic hierarchy.

As an example a researcher is expert on the topic of “knowledge management” and is looking for references of paper on the same topic. He/She can get bibliographic references using the search functionality of our system. Along with this, researchers can also determine which other researchers are working on the same area. In case this researcher has some bibliographic references which are not available with anyone else on the network, it can advertise those references to the researchers whose expertise are also in the same area.

4.8.Agents Roles

As it has been described in chapter 2 that every agent has a role in a scene and it performs those actions which are associated with its role. Now we look at different roles that agents can have in this project:

4.8.1.Requester

All the agents which are sending query message to other agents are in the role of

requester. This role is adopted in the situation when an agent does not have some information which is currently needed so it is asking for the information

4.8.2.Replier

In the role of replier, an agent receives a query and send answer message to the requester of that information. In the execution of one query, an agent can shift between its roles e.g. When an agent A receives a query from another agent B, A is in the role of replier. When A is forwarding query to other agents and receiving answers, it is in the role of requester. After receiving answer message, when it sends back the reply to agent B, which is the actual querying agent, A will be again in the replier role.

4.8.3.Advertiser

In this role, every agent sends advertisement messages to all of its known agents (which is 4 in our case).This advertisement message describes the expertise of the sender from ACM CCS (cf. section 4.6).

4.8.4.Receiver

In this role, every agent receives advertisement messages from all of its known agents (which are 4 in our case). These advertisement messages contain the expertise description of the sender.

4.9.LCC Protocols for Routing

As we have discussed that three routing mechanisms have been used in this project, we now look at the LCC version of these protocols:

4.9.1.Overt Protocol

Agent A in role of Requester asks about query X from agent B in role of Replier. If agent B knows about the answer Y of query X, then it will send an answer message to agent A. Otherwise agent B will send a re-route message to agent A, introducing agent C (which is a known agent of agent B but agent A does not know about that) to agent A, so that agent A can directly ask from agent C (as every agent only communicates with the

querying agent in overt protocols).

a(Requester(B), A) :: Enquire(X) ⇒ a(Replier,B) ← needs (X) then

answer (X,Y) <= a (Replier, B)
OR
re-route (C) <= a (Replier, B) then a(Requester(C), A)

a(Replier, B) :: Enquire(X) <= a (Requester (B), A) then

answer (X,Y) ⇒ a(Requester(B), A) ← has_answer(X,Y)
OR
re-route (C) ⇒ a(Requester(B), A) ← not (has_answer(X,Y)) and peer (C)

4.9.2.Covert Protocol

Agent A in role of Requester asks about query X from agent B in role of Replier. If agent B knows about the answer Y of query X, then it will send an answer message to agent A. Otherwise agent B will become requester itself and ask from one of its known agents and this process continues. When any agent has got the answer for the query, it will send it to its immediate sender of query message and agents keep on sending answer messages back to their senders till the first agent (agent A in this example) receives the answer.

a(Requester(B), A) :: Enquire(X) ⇒ a(Replier(A),B) ← needs (X) then
answer (X,Y) <= a (Replier(A), B)
OR
no_answer(X) <= a (Replier(A), B)

a(Replier (A), B) :: Enquire(X) <= a (Requester (B), A) then
answer (X,Y) ⇒ a(Requester(B), A) ← has_answer(X,Y)
OR
a(Requester(C), B) ← not (has_answer(X,Y)) and peer (C)
OR
no_answer(X) ⇒ a(Requester(B), A) ← not (has_answer(X,Y)) and not(peer (C))

4.9.3.Hybrid Protocol

Agent A in role of Requester asks about query X from agent B in role of Replier. If agent B knows about the answer Y of query X, then it will send an answer message to agent A. Otherwise agent B will become requester itself and ask from one of its known agents and this process continues. When any agent has got the answer for the query, it will send it to first agent (I in this example).

a(Requester(I,B), A) :: Enquire(X) ⇒ a(Replier(I,A),B) ⟵ needs (X) then

answer (X,Y) <= a (Replier(I,A), B)

OR

no_answer(X) <= a (Replier(I,A), B)

a(Replier (I,A), B) :: Enquire(X) <= a (Requester (I,B), A) then

answer (X,Y) ⇒ a(Requester(I,B), I) ⟵ has_answer(X,Y)

OR

a(Requester(I,C), B) ⟵ not (has_answer(X,Y)) and peer (C)

OR

no_answer(X) ⇒ a(Requester(I,B), A) ⟵ not (has_answer(X,Y)) and not(peer (C))

4.9.4.Advertisement Protocol

Agent A is in the role of Advertiser and it wants to send message M (which contains its expertise description) to all of its known agents whose ids' are in list S. Agent A will keep on sending message M until all the known agents receive M. Agent B in the role of receiver will receive message M from all of its known agents whom ids' are in list T. Select is a predicate which pulls out first element from the list.

a(Advertiser(M,S),A)::= (M ⇒ S1⟵S=[S1|Rest] then

a(Advertiser(Rest),A)

OR

null⟵S=null

a(Receiver(M,T), B)::= select (S1,T,Rest) ⟵ M <= S1 then

a(Receiver(M,Rest), B)

OR

null ⟵ T=null

4.10.Settings

In this project, we are comparing different types of agent selection mechanisms. If the querying agent and other intermediate agents (cf. section 4.12) randomly select other agents to forward query, this is known as the naïve setting, while if they select agents on the basis of semantic similarity (cf. section 5.3) between agent's expertise and query subject (cf. section 5.2.2), this is known as the semantic setting.

4.11.Input Parameters

In this section we define the input parameters which will provide a basis for our algorithm evaluation. These parameters are mainly from information retrieval community and mainly discussed in [37].

4.11.1.Number of Agents

The number of agents in an agent based P2P system is regarded as the size of the network [26]. Other than scale up experiments, we have used 20 agents in our simulation.

4.11.2. Number of Bibliographic References

"The ability of a P2P system can also be expressed in terms of its shared resources" [26]. In our case, as we are not making any restrictions on the number of results retrieved so this parameter does not make any difference to our results. We have set this parameter as a constant.

4.11.3.Data Distribution

Almost every data distribution has some pattern and is not completely random [26]. We have used two distributions in this project, first is topic distribution (cf. section 5.7.1) and second is proceeding distribution (cf. section 5.7.2). We study how our algorithm performs in two different types of data distributions.

4.11.4.Network Topology

The performance of a P2P system is strongly dependent on the network

topology and can vary with change of network topology. Examples of different possible network topologies can be a star topology, a ring based topology, a super-peer based topology or simply a random topology. In our simulation we are assuming a random topology. Every agent in the simulation knows 4 other agents in the network. No further assumptions have been made about the network topology.

4.11.5.Advertisement

Semantic topology (knowledge of agents about the expertise of other agents) in the network is built on the basis of advertisement in the network. There are many variables related to advertisements e.g. whom to send advertisements and which received advertisement messages to accept.

4.11.6.Agent Selection Algorithm

This algorithm selects agents to whom query will be forwarded. In a naïve setting (cf. section 4.10), query is forwarded to two randomly selected agents, while in semantic setting (cf. section 4.10), query is forwarded on the basis of agents' expertise.

4.11.7.Maximum Number of Hop Counts

This parameter determines that how many times a query is allowed to be sent to next agent. From this parameter, we can determine that how much the network will be flooded by a single query [26]. We have set this parameter (n=2) in most of our experiments but we have also performed experiments by increasing and decreasing this number.

4.12.Intermediate Agents

Agents which receive query from one of their known agents and forward it to other known agents are known as intermediate agents.

4.13.Relevance Calculation

When an agent looks into its knowledge base to find results for a query, it also determines that how much these results are relevant to the query. Different users may

require different result relevance criteria; however we have set it to be 50 % in our project which means that at least half of the keywords entered by the user must be in the result.

As an example of relevance calculation, the querying agent receives the query of "business process modeling and multi-agent system". The querying agent will discard the word "and" from the query. Each word in the remaining phrase has 20% weight. If any particular result has two words from these five words, then it has 40% relevance with the query. If another result has all of these 5 keywords, it is 100% relevant with the query.

However we also performed sensitivity analysis (cf. chapter 7) and test our algorithm at different relevance criteria including 75% and 100% relevance. We have seen that performance of naïve algorithm (cf. section 4.10) reduces significantly as we increase the size of network (cf. chapter 7).

Chapter 5

5.System Design

Although in our implementation of the system, we have used simulation on a single machine in which we have demonstrated all the agents. However our protocols for query routing can work in real applications. Similarly our agents are by no means dependent on the simulation. They are complete entities themselves. Although there is no network communication taking place in reality in the simulation but all the required components are present for the complete functionality of the system.

5.1.System Components

We will now briefly describe the components of our system:

5.1.1.Local Knowledge base

Information gathered from local knowledge sources or acquired from other sources are stored here. All the queries are answered from this module. It provides a basis for peer ranking and selection

5.1.2.Advertiser

The responsibilities of an advertiser are to advertise the currently available knowledge of the agent to its known agents in the network. This is realized by sending advertisement messages about the expertise of an agent over the network. In our system this description consists of those topics in which that agent is expert.

5.1.3.Query Replier

It is coordinating component which controls the process of replying queries. As our system is implemented in Java, this component consists of threads, which support handling multiple queries at a time. However the simulator, we are using have only support for dealing with a single query at a time, so in our current system we are not dealing with multiple queries at a time.

5.1.4.User Interface

The user interface of our system allows the user to search for particular keywords, visually look interaction between agents through message passing, read details of messages from message board and look at the results in the result table. The user Interface has been described in detail in chapter 6.

5.2.System Modules

We will now look at all the modules of the project in the order of sequence of events in detail:

5.2.1.Expertise Advertisement

Our agent based P2P system consists of "A" agents. Every a ϵ A has a local knowledge base that contains the knowledge which it wants to share with others over the network. All of these agents use the same ontology "O" (a shared conceptualization of their domain). This ontology is used in describing expertise of the agent and the subject of the queries. An expertise e ϵ E is the semantic description of the knowledge base of the agent which is based on common ontology O. The expertise of an agent can be determined from its knowledge base. Advertisements $A^{/} \subseteq$ A X E are used to distribute the expertise of an agent in the network. An advertisement $a^{/}$ is used to relate an agent a with an expertise e. Every agent decides itself whom to send advertisement based on the network topology [26].

The advertisement messages are sent at the start of each simulation, which basically is the network bootstrap time in which every agent is telling other agents about its expertise. In result of this advertisement "known" relationship is established between two agents. Knows $\subseteq$ A X A where knows (a1, a2) means that a1 and a2 knows about the expertise of each other. Hence they both are known agents of each other [26]. In this project we have always used this assumption that if an agent a1 knows agent a2, then agent a2 also knows agent a1.

The last thing about the advertisement messages is that agents can also discard the advertisements which are unrelated their own expertise. However in our current system

agents accept all advertisements.

5.2.2. Query Subject

A subject s ϵ S is an abstraction of given expression expressed in terms of ontology [26]. Queries are given by the user to find the relevant results in the whole network. In order to select the appropriate querying agent and appropriate known agents of querying agent we need to find the subject of query. This subject will be matched with the expertise of the agents. We have used 20 ACM topics in our initial setting because we have 20 agents in it and we are demonstrating that every agent is expert of only one topic.

All of these topics have keywords associated with them. When a user enters the query, we match the keywords entered by the user with the topic keywords. The expertise which has maximum similarity will be selected as the subject of the query. This query subject is third level node in ACM CCS. At the time of determining the query subject, we also determine its first and second level topic from ACM CCS. This information will be used in query relaxation module.

5.2.3.Querying Agent Selection

Querying agent is the first agent which receives the query entered by the user. The selection of querying agent is based upon the subject of the query. The subject of the query is matched to the expertise of all the agents. As described earlier, the expertise of an agent can be determined from its knowledge base. The agent, whose expertise is most closely related with the topic of the query, will be selected as the querying agent. This querying agent will start the interaction in the network. It will gather all the relevant results returned by other agents in the network.

Once the querying agent is selected and the query is passed to that agent, then depending upon the setting selected by the user, an appropriate LCC protocol is received by the querying agent. Querying agent will always be in the role of requester other than at network bootstrap time when advertisement messages will be sent by it in the role of advertiser and received by it in the role of replier. With the help of the LCC interpreter, all the agents will be able to find their task from the protocol and act according to that. The task of the interpreter is just to translate the LCC protocol into a form which is

understandable by the agents. The interpreter capabilities are limited to the extent of LCC which is being used in this project. This interpreter is currently embedded into the simulator and cannot be used separately.

Below is the querying agent selection algorithm:

```
Q:=Query topic
E:= Expertise of all agents
A:=selected Agent
MaxSimilarity:=eo // first agent is assumed to have maximum similarity
for every e ε E do
        if(calculateSimilarity (Q, e) > MaxSimilarity then
                MaxSimilarity=calculateSimilarity (Q, e);
                A= getAgentOfExpertise(e);
end;
```

5.2.4.Local Knowledge Base Lookup

Every agent keeps the bibliographic references in its local knowledge base which it wants to share with other agents in the network. Whenever an agent receives a query, it looks into its local knowledge base. It is assumed that all the agents in the system have this much intelligence. In our LCC protocols, we have not added this clause which asks the agent to look into its local knowledge base after receiving its query message.

5.2.5.Message Forwarding Mechanisms

Every agent including the querying agent after looking into its local knowledge base will check the hop count (cf. section 2.11) of the query. There are only two well known methods of determining when to stop forwarding the query to the next agent in the network:

Introduce a limit on number of answers retrieved, when a certain number of answers are retrieved, query should be stopped forwarding to next agents. This method is not a good choice in our case as in most of our settings an agent is only expert of one topic from ACM topic hierarchy. Due to this, if that particular agent is not selected then the query has to traverse the whole network. Alternatively if that agent is selected as the querying agent then there will be no routing at all, both of these situations are not desirable for our

project.

Set a limit on the number of agents traversed. This approach is more reasonable as neither we want the query to traverse the whole network nor not traverse at all. So setting a limit on the number of agents traversed is a good approach to ensure that this query will traverse this many number of agents for sure.

After looking into its local knowledge base, every agent will check the hop count (cf. section 2.11) of the message. In our all of the settings in this project, we have set the hop count limit n=2. In every message we are storing the following information:

- Message Types
- Query Originator
- Message Sender
- Message Receiver
- Hop Count
- Visited Agents

There can be four possible types of messages i.e. query messages, answer messages, re-route messages and advertisement messages (cf. section 4.3). Specifying the message type in the message is necessary as, after receiving the message, an agent performs different actions depending upon the type of message. Query originator information is stored in the message because in case of overt mechanism, every agent is only communicating with the querying agent which is originator of the query, while in covert mechanism every agent will keep on sending the answer message back till the query originator receives it and in the hybrid mechanism every agent sends the answer message only to the query originator. So in all of our mechanisms we need information about the query originator.

Message sender and message receiver id's are also stored in the message to determine which agent has sent the message to whom. This information will also be required for the network communication purposes. As in our current implementation, we are not using any network communication but the system has been designed in the way that it can be deployed in real applications by adding network support.

Visited agents ids are also stored in the message so that if the same agent is selected again as a candidate for query, query should not be passed to it as it has already answered what it knows. By keeping this information stored in the message, we can save a lot of messages on the network. Finally results and expertise of answering agent (the agent which has originated the answer message) are also stored in the answer messages.

As the querying agent is the originator of the query and the hop count of that query is 0 till now, so it will forward the query to its known agents. In order to save the network from redundant messages and not involving those agents which are not related to the subject of the query, the querying agent needs to select agents from the list of its known agents based upon their expertise and then forwards the query to those agents.

5.3. Similarity Function

Querying agent uses the subject of the query given by the user and matches that subject against the expertise of its known agents using a similarity function. The similarity function SF: S X E can have values between 0 and 1. The values closer to 1 mean agent expertise and query subject are closely related, while values close to 0 denote less similarity between query subject and agent expertise. If the value is 0, the topics are not similar at all and if the value is 1, query is exactly related to agent's expertise. Querying agent will forward the query based upon their SF value with the query subject [26].

The idea of using a similarity function is based on the notion that topics which are close according to their positions in the tree are more similar than those which have a larger distance. As an example, agents whose expertise topic is “Distributed Artificial Intelligence/Multiagent systems” is more likely to answer the query related to “Intelligent agents” as compared to that agent whose expertise is related to “Programming Techniques/Logic Programming”. [34] have studied similarity functions for hierarchically structured semantic networks like ACM topic hierarchy which we are using as common ontology in our project. According to them, the following similarity function gives the best results:

$$S(t_1, t_2) = \begin{cases} e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} & \text{if } t_1 \neq t_2, \\ 1 & \text{otherwise} \end{cases}$$

l is the length of the shortest path between topics t1 and t2 and h is the level in the tree of the direct common subsumer of t1 and t2. α and β both are positive numbers and they are scaling the contribution of shortest path of length l an depth h respectively. The optimal values of α and β are 0.2 and 0.6 respectively based upon their benchmark data [10]. They may have different values in some data sets but we are taking these both values as constants in all of our experiments. Depth of direct common subsumer has been included in the calculation because topics which exist at the upper layer of these hierarchical nets are more general and semantically less similar than the topics at the lower levels [35]. Rada [35] have proven that the minimum number of edges separating two topics is a metric for measuring the distance between two topics in the tree.

This function lies in the category of edge counting-based or dictionary thesaurus -based methods. This method is useful for those applications which have highly constrained taxonomies e.g. Medical semantic net [35].

As an example we calculate the semantic similarity between two topics "functional programming" and "object oriented programming". In order to reduce space, only relevant part of ACM topic hierarchy is shown below in the figure 11.

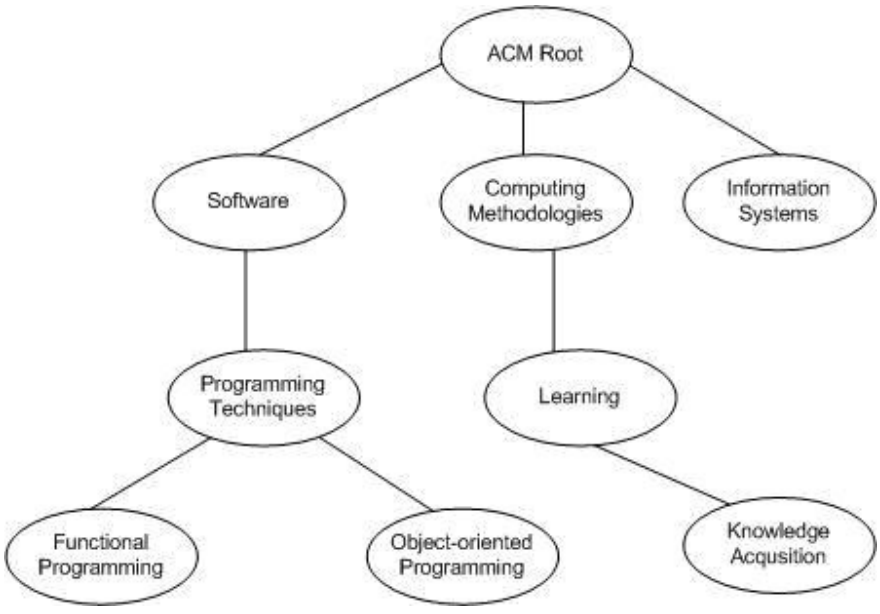


Figure 11 ACM Topic Hierarchy

The shortest distance between these two topics is 2 and their direct common subsumer is

the node containing topic "Programming Techniques" which is immediate parent of these both topics. "Programming Techniques" node is at second level expertise of ACM topic so the direct common subsumer value will be 3 (as we take root at level 1), so we have the following similarity value:

sim("Functional Programming", "Object-oriented Programming") = 0.63

Applying the above formula we get the similarity value between both topics is 0.63 which is above our threshold of 0.5. So we can forward the query related to "functional programming" to the agent whose expertise is related to "object-oriented programming" and vice versa. Now we calculate the similarity of "functional Programming" and "Knowledge acquisition", which comes out to be 0.3 which is less than our threshold, so we do not pass queries related to "functional Programming" to the agent whose expertise is in the field of "knowledge acquisition".

sim("Functional Programming", "knowledge acquisition") = 0.3

We now discuss the pseudo code of the agent selection algorithm for query forwarding:

This algorithm assigns a rank to all the candidate agents. The value of this rank is same as provided by the similarity function. From these ranks, one agent can select the best "n" known agents from the list of all known agents to forward the query.

```
L:= List of all known agents
A:= Advertisements available about the expertise of known agents.
rankedAgent:= φ
Threshold:= 0.5
for all adver εA do
        agent:= getAgentOfExpertise (adver)
        rank:=getSimilarityFunction(Expertise(advr))
        if(rank>threshold)
                rankedAgent:=rankedAgent ∪(agent,rank)
                return rankedAgent
return;
```

From this list of ranked agents, we will take the two highest rank agents. In this algorithm we have also set a threshold of at least 50 % similarity to make sure that the query is forwarded to those agents which contain the answer of the query. In this way

querying agent will select those agents which are more likely to contain the answer of the query.

When the intermediate agents (cf. section 4.12) receive the query, they will repeat the above process again. In short, they will look into their knowledge base, if they get relevant results they will send the answer message to the querying agent followed by checking hop count value. If the hop count is still less than the maximum value then they will also send the query to those agents which have similar expertise as that of query topic and this process continues till the hop count of the query reaches its maximum value.

The querying agent will keep on returning the results to the user by displaying it in the results table as soon as it gets answers from different agents in the network.

5.4.Query Relaxation

We have discussed above that when an agent wants to forward the query to other agents, their expertise similarity with the query subject must be above a certain threshold which we have set to 0.5. Now if the agent is not successful in finding any such known agent which has at least 50% similarity with query topic, then it will relax the query. In query relaxation we exploit the consideration that if an agent has answered similar queries before, it is more likely to answer query as compared to those agents which have answered queries in totally different domain.

For this the agent will look at the second node level topic of the query which is more generalized than the previous topic. Now the agent will match the more generalized query with the agent's expertise. Note that each agent has also three levels of expertise so query topic and agent's expertise can be matched easily.

Now after matching generalized query topic with agent's expertise, the agent will pass the original query to those agents whose similarity function has returned values above the threshold. In our experiments, we generally find suitable agents after relaxing the query. If the agent is still unsuccessful in finding such a suitable agent then it will forward the query to those known agents which have answered most of the queries in the past.

5.4.1.Duplicate Filtering

As our system is based upon distributed system without any centralized control and the results come from different agents in the network, it is possible to get duplicate results. If the query subject is related to expertise of more than one agent then the chance of getting duplicate results is fairly high. Querying agent is actually responsible for collecting results from different agents and displaying them in a table. Duplication filtering is the responsibility of querying agent.

For this task, the querying agent will match the titles of new records with the already existing records. If the title of both the records is the same, then the new result will be considered as the duplicate result.

In this system, duplicate results are considered as one result, so we do not show the same results again and again; rather present the user only one record. However if there are conflicting property values e.g. Two records are exactly the same other than the publisher detail. One of them have publisher "ACM Press" and the other has "Association for Computing Machinery Press". We know that both of these values are the same but the querying agent does not have this intelligence to infer this. In this case the querying agent will take that value which has more detail than other. In the above example, the querying agent will select "Association for Computing Machinery Press" as publisher. It selects this value by counting words in both values and selects the one which have maximum words.

In the case when there are two records for a particular bibliographic reference and both have some missing property values, then the querying agent will take union of both records and give the user one complete record e.g. The querying agent receives two records for title "A peer-to-peer network for open archives" but one record does not have publication year information and second record does not have the conference name information. The querying agent will merge both records and present a complete record to the user.

5.5.Search Mechanism

Though our project is very closely related to information retrieval, our focus is on query

routing mechanisms, instead of efficient search algorithm implementation. We have used a simple keyword based search mechanism which matches the query keywords with the keywords of bibliographic references. As the target user is a researcher, who has information about the title of the bibliographic reference, he/she is looking for, so there are fair chances that he/she will include majority of important keywords from the title in the search criteria and can easily find the results he/she is looking for.

The researchers can also add those words in the search criteria which do not have any significance in getting results e.g. prepositions like “for”, “of”,” by” and conjunctions like “or”,” and” etc. So the querying agent will eliminate all this kind of words from the query. As we have limited amount of keywords from ACM hierarchy we have distributed these keywords into two categories. The first category is of most important keywords and the second category is of the less important keywords. Examples of first category are “distributed”, “multi-agents” etc. while the examples of second category are “systems”, “programming” etc. First category of words has more weight as compared to words of second category.

When the agents are performing search in their local knowledge bases, they will not separately chose keywords of the second category but they will pick a first category word from the query which is adjacent to second category words and then search for that e.g. If the user is searching for the keywords “Security in Distributed Systems”, the query agent will eliminate the keyword “in” from the query and the remaining phrase will be searched in the local knowledge base and will be passed to other agents. In the remaining phrase “security” and “distributed” are first category keywords where as “systems” is second category keyword. So the word “system” will not be searched alone but the compound word “distributed systems” will be searched.

This categorization helps us in eliminating irrelevant results because second category keywords can come in many different expertise queries e.g. word “system” can come in queries like “multiagent systems”, “database systems” and in “distributed system”. All of these come under different ACM expertise.

Though there is no one definition for relevant results. It depends upon the needs and type of user that how much relevant result is required by him/her. Some users require

very strictly close results and some wants to look at loosely related broad range of results. However we have set a medium level relevance criterion which is 50%, so every agent will return only those results to the querying agent which has at least half of those keywords which are entered by the user.

5.6. Data Set

To obtain large amount of bibliographic references (1200 bibliographic references), we have used the ACM portal[1]. We have selected 4 main categories in ACM CCS and some of their sub categories and in each selected sub category, we have selected some subcategories randomly. For each of third level subcategory we have taken twenty references from ACM portal.

To decide about whether a particular publication comes under a given classification or not we have left this up to ACM portal search. We have selected on each of our topic manually and selected the results. Though there are many sophisticated classification mechanisms but as high precision of classification is not required in this project, so this method worked well.

5.7.Data Distribution

In the simulation and the evaluation of this system, two different data distribution schemes have been used as described in the following:

5.7.1.Topic Distribution

In this distribution, bibliographic references are distributed according to their topics. There is only one dedicated agent for each of the expertise. However few of the references related to that topic are spread to two more agents. One agent will be from the known agents of that agent and other agent from outside the known agents list.

The reason for this is we want to study that whether the results which are present with other two agents are included in the result or not. Another reason is that we do not want

1 www.acm.org

all results to be coming from one agent only as there will be no point of routing the query in this case.

5.7.2.Random Distribution

In this distribution, references are distributed randomly across the network. In this setting more than agent is expert on one topic in the ACM CCS (cf. section 4.6). This distribution is basically simulating that each agent contains bibliographic references of one conference proceedings. As there are multi topic conferences in which proceedings belong to more than one particular topic, so in this case agents can be expert of more than one topic. However conferences generally cover a coherent set of topics, so there will still be a correlation between agent's expertise and distribution.

Chapter 6

6.User Interface

In this chapter user interface of the simulation has been described. All the steps required by the user are explained in step by step manner both in textual as well as graphical form:

The simulation starting interface is shown in Figure 12. On the left side, ACM topic hierarchy (cf. section 4.6) is displayed in a tree format. The user will specify an area which is closely related to his/her query from this table. In the centre of the simulation, all the agents are displayed as blue nodes with their ids. On the right side of the simulation, there is a notice board, on which all the messages will be displayed. There is a result table at the bottom section of the simulator to show the query results to the user.

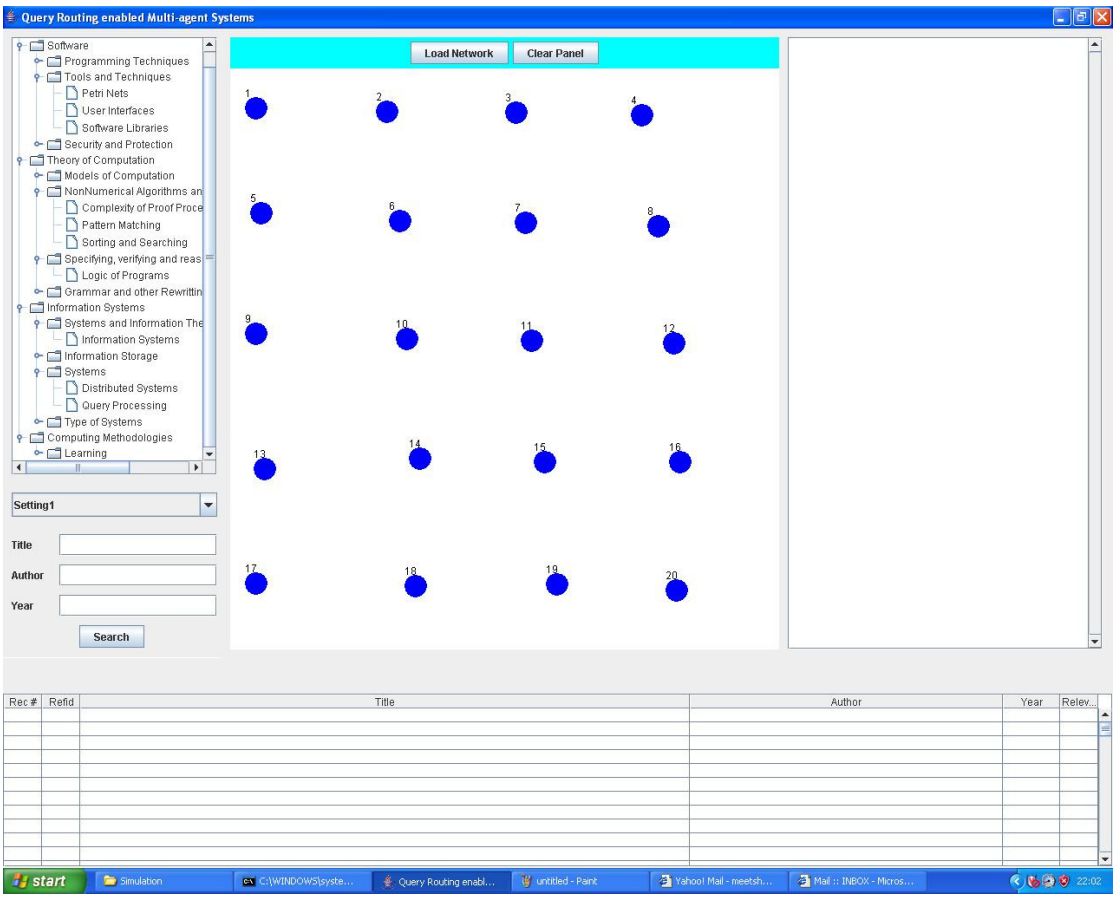


Figure 12 Simulation Starting Interface

The user will select a node from ACM topic tree, which he/she thinks, is closely related to the query from the available options as shown in Figure 13. This topic tree has been populated with ACM topics for computer science.

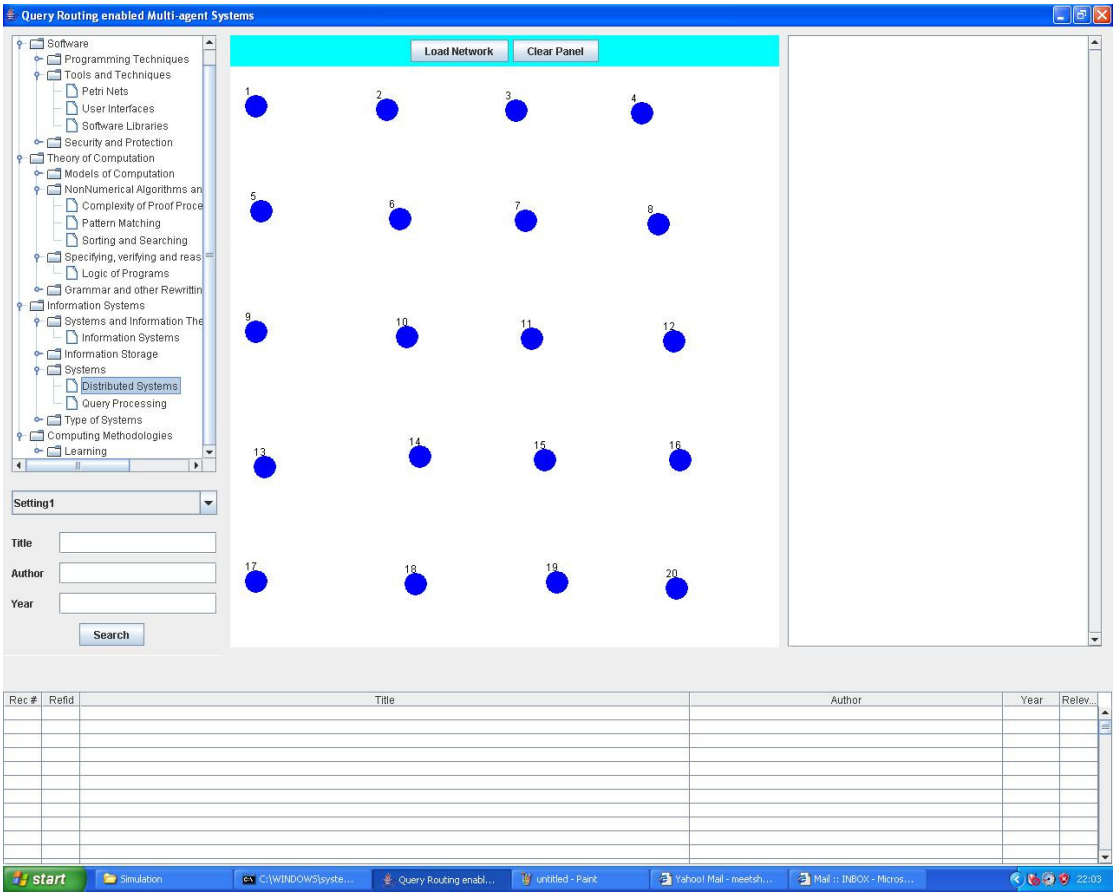


Figure 13 Topic Selected from ACM Topic Hierarchy

The user will select a setting (cf. section 4.10) from the settings drop down list as shown in Figure 14. Currently, there are six settings in the simulation. Following is a brief summary of all the settings in Table 1.

Setting No.	Agent Selection Algorithm	Routing Mechanism
1	Naïve	Overt
2	Naïve	Covert
3	Naïve	Hybrid
4	Semantic similarity based	Overt
5	Semantic similarity based	Covert
6	Semantic similarity based	Hybrid

Table 1 Simulation Settings

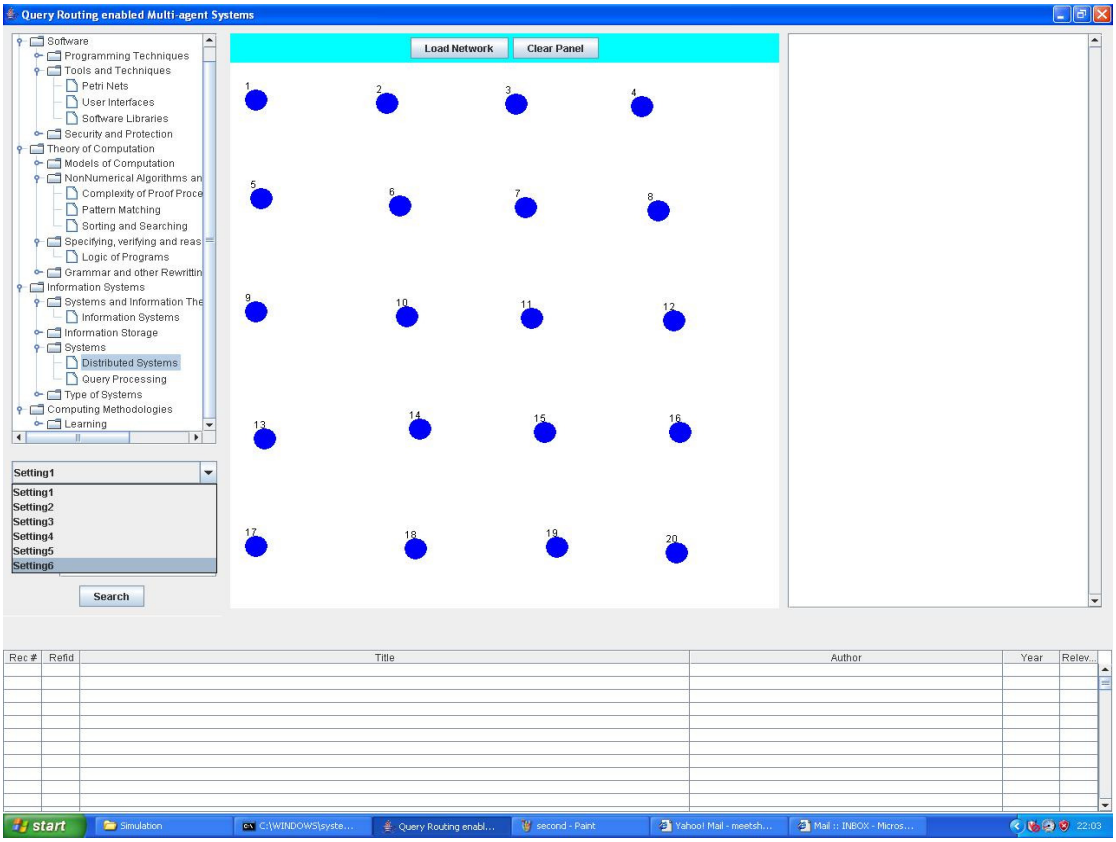


Figure 14 User selecting setting from setting drop down list

The user will enter the keywords of bibliographic reference; he/she wants to search. We assume that the user is aware of title of major keywords of the title; he/she is going to search. After entering these keywords, he/she will press the search button as shown in figure 15.

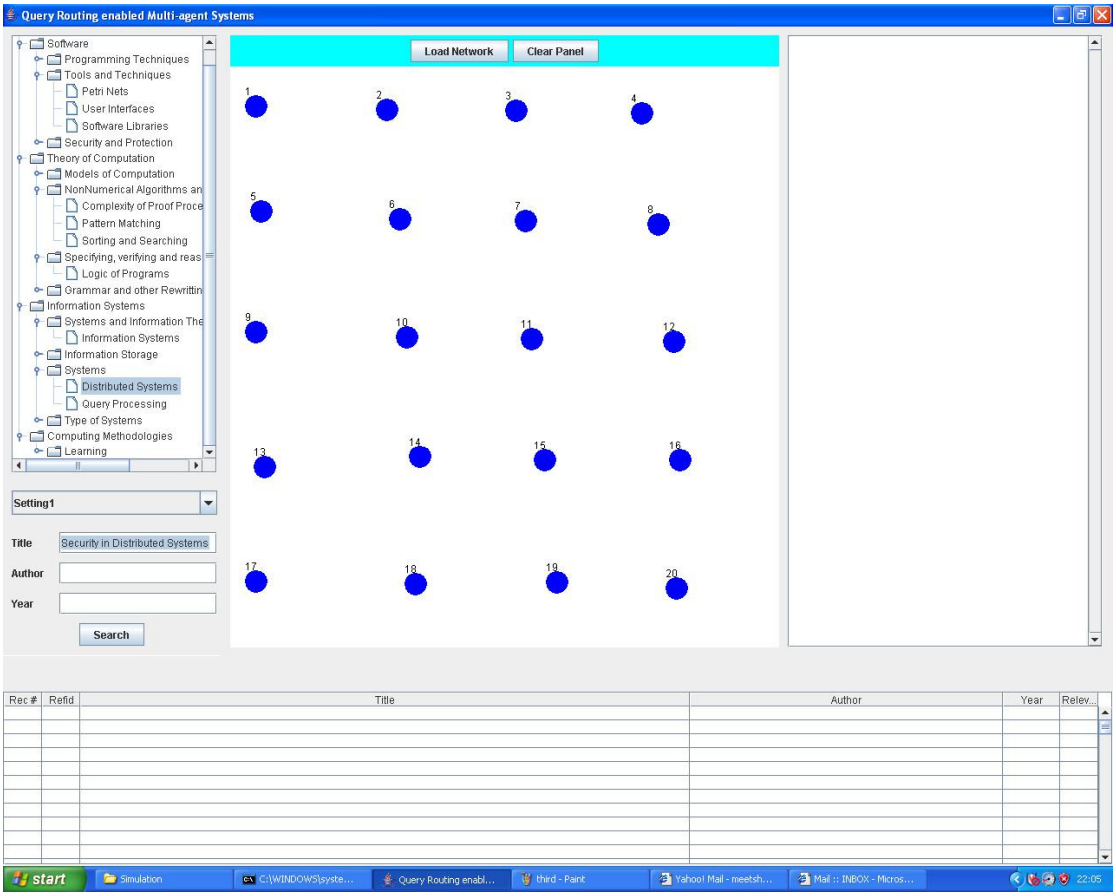


Figure 15 Search Keywords are entered

The first agent will be selected. It will perform local knowledge base lookup, forward the query to appropriate known agent as per query selected by the user, collect results from different agents in the network and display the results in results table as shown in Figure 16. All the agent communication will be displayed in notice board for user information.

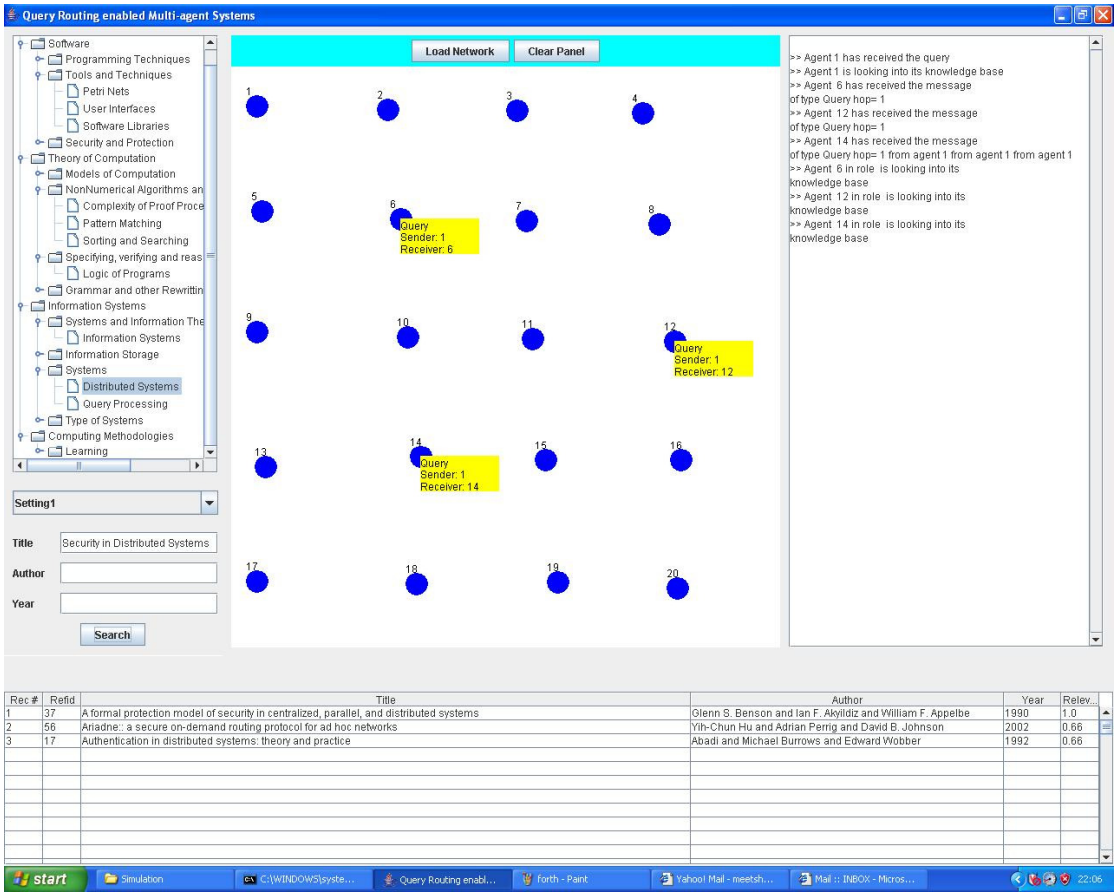


Figure 16 Results Displayed

The user will click on the desired record in the results table and the complete bibliographic reference for that record is displayed to the user in Bibtex format in a new window as shown in Figure 17.

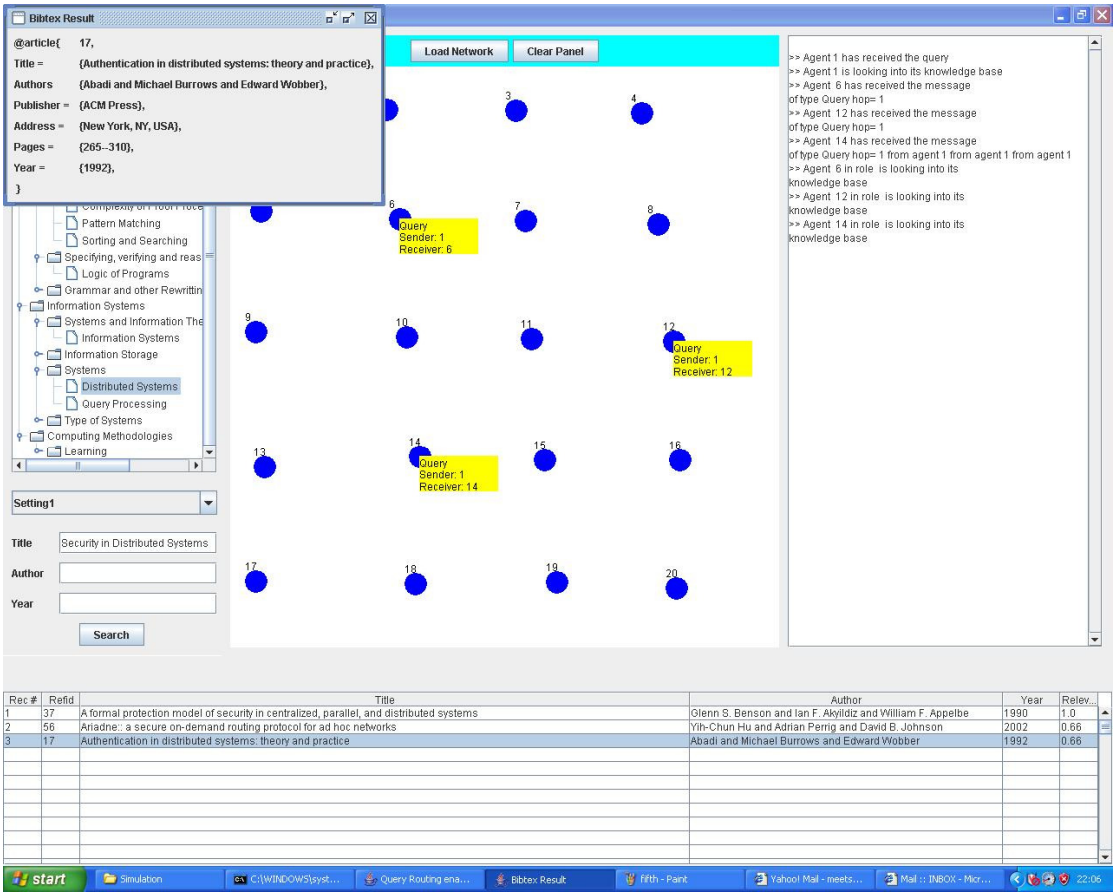


Figure 17 Result Details

The user can repeat the search process by entering different keywords, changing ACM topic from the tree or changing the setting.

Chapter 7

7.Evaluation

Evaluation of a system helps us to compare it with other developed systems in the related area as well as study the effect on the performance of overall system with changing certain input parameters. There are many known ways to evaluate conventional P2P networks, but the focus of this project is more towards semantic issues of P2P systems which is not very mature area and there are no standard methods available to evaluate this kind of P2P system. However some of the important evaluation functions are described below.

7.1.Relevance

Relevance is the measure that whether the retrieved information is of importance with respect to query or not. This can be calculated using many methods. In this project we have done relevance calculations using keyword matching. The possible values of relevance can be between 0 and 1.

7.2.Recall

Recall is the measure of all relevant documents from the retrieved set [26]. It is one of the standard measures of information retrieval domain. Recall can be defined as:

$$\textbf{Recall} = \frac{\textbf{Relevant} \cap \textbf{Retrieved}}{\textbf{Relevant}}$$

Where “Relevant” is the set of relevant documents and “Retrieved” is set of those retrieved.

7.3.Precision

Precision is defined as the proportion of retrieved set that is relevant [26]. It is also one of the standard measures of information retrieval domain.

$$\textbf{Precision} = \frac{\textbf{Relevant} \cap \textbf{Retrieved}}{\textbf{Retrieved}}$$

In this project as we are matching exact queries, therefore retrieved queries will also be relevant. So Precision in our case will always be 1.

$$\textbf{Precision} = \frac{\textbf{Relevant}}{\textbf{Relevant}} = 1$$

7.4.F-measure

"F-measure describes the symmetric difference between retrieved and relevant documents" [38]. It is one of the most common combinations of above two parameters.

$$\textbf{F-measure} = \frac{(\beta^2 + 1) + PR}{\beta^2 P + R}$$

where $\beta = P/R$ [38].

7.5.Network Load

Network load is a measure of messages sent on the network per query. It is being used to determine how much network is being flooded as a consequence of each query. Some other evaluation functions are Information Loss, Reliability and Real time [38]. For the purposes of this project, recall and network load are the most related ones. The same parameters are used for the evaluation of SWAP project [26]. So, we have calculated these evaluation functions using different combinations of input parameters.

We will now discuss evaluation of this system, using varying input parameters (cf. chapter 4.11):

We have used three different routing mechanisms in this project namely overt, covert and hybrid. The only difference between them is that they send messages via different agents. Due to this, different numbers of messages will be sent across the network for a particular query for each of these schemes. Figure 18 shows the results of messages sent on the network using these three mechanisms. As can be seen from figure 18, overt and

covert involve almost the same number of messages. They differ in number of messages only in those queries in which same agent is selected again for sending query in one of them and not in other mechanism. Hybrid always involves less or an equal number of messages in all cases.

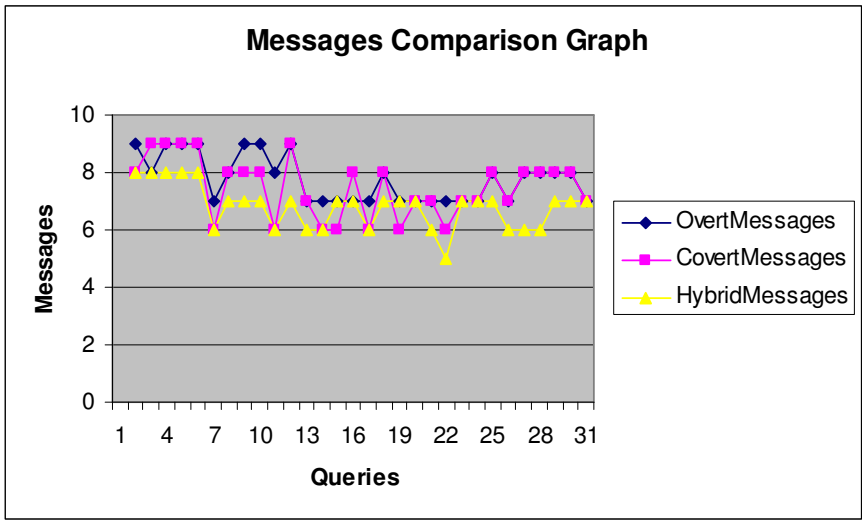


Figure 18 Number of messages comparison

The above graph is both for semantic and non-semantic settings (cf. section 4.10). The number of messages does not depend upon semantic or non-semantic settings. Hence in terms of number of messages, hybrid always performs equal or better than the other two mechanisms.

We now compare naïve settings of all these three routing mechanisms with their semantic ones. The selection criteria of the agents to whom the query will be forwarded is different in naïve and semantic cases. The naïve algorithm randomly selects any two agents from the known agents of the querying agent and forwards the query to them, while the semantic algorithm selects the two agents on the basis of their expertise. In Figure 19, 20 and 21 data distribution, number of queries, number of known agents and routing mechanism are kept constant. The size of network is different in Figure 19, 20 and 21. Figure 19 has a network of 20 agents, Figure 20 has a network of 40 agents and Figure 21 has a network of 60 agents.

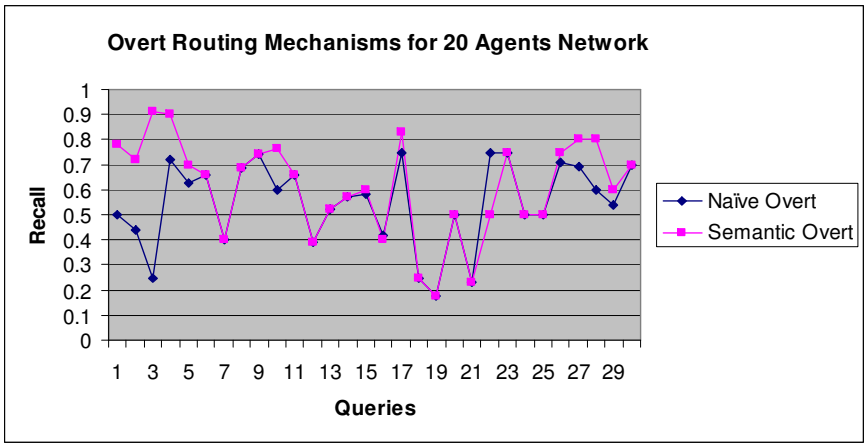


Figure 19 Naive and semantic overt routing for 20 agents network

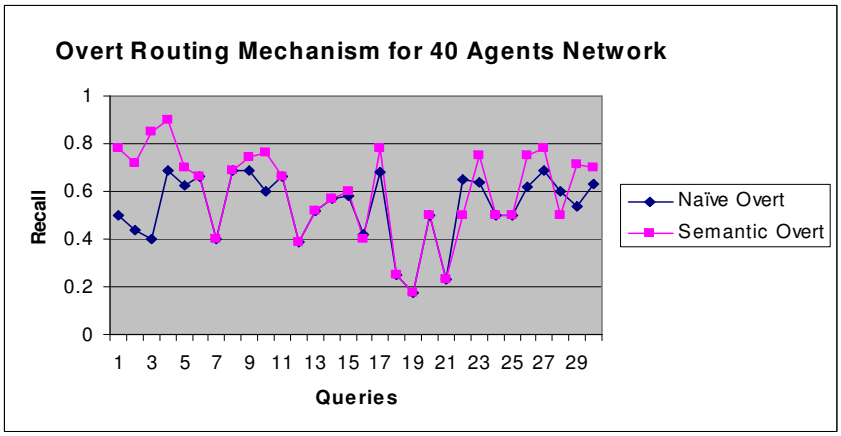


Figure 20 Naive and semantic overt routing for 40 agents network

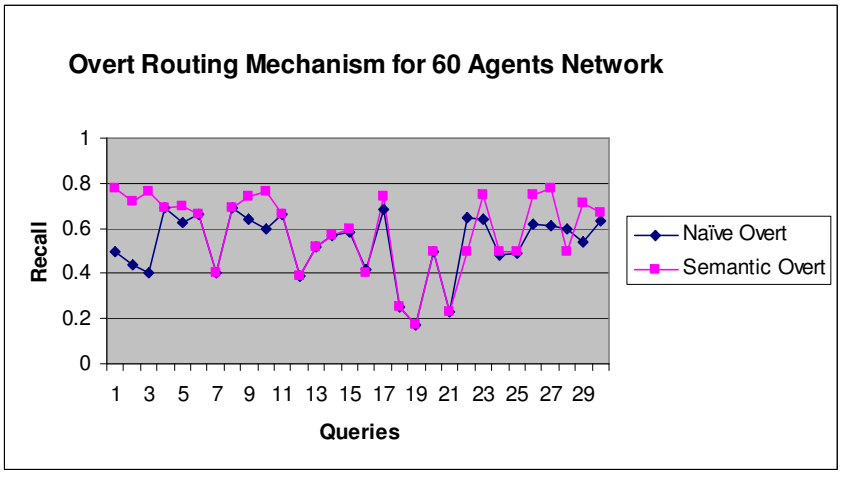


Figure 21 Naive and semantic overt routing for 60 agents network

In the case of 20 agents' network, the recall rate of naïve algorithm (cf. section 4.10) is 54% and semantic algorithm (cf. section 4.10) is 60%. Though the semantic recall is 6% more than the naïve one, this difference is small for the following reasons:

First, the network is small. It consists of only 20 agents and the hop count (cf. section 2.11) is 2 which mean that the query will be forwarded to about 7 agents which is more than one third of the network. For this reason, there are high chances that the naïve case (cf. section 4.10) has also been able to find that agent which contains the majority of the answers (as this is topic distribution). So if an agent is reached whose expertise is same as that of querying agent, then majority of the relevant results will be retrieved. As it can be seen in Figure 20 and 21, when the size of the network has been increased the recall rate of both techniques has been lowered, but the naïve has lowered proportionally more than the semantic.

The second reason is that as agents are assigned to other agents without considering their expertise, so there can be a chance that both naïve and semantic selects the same agents in some cases and therefore the difference between them is not that much.

In Figure 19, 20 and 21, the routing mechanism is overt. Now we look at covert and hybrid mechanisms and study the effect of increase in network size on recall. In Figure 22, 23 and 24 routing mechanism is covert while in Figure 25, 26 and 27 the routing mechanism is hybrid. From Figure 22 to 27, we can see that semantic algorithms (cf. section 4.10) are performing better than the naïve algorithm in terms of recall.

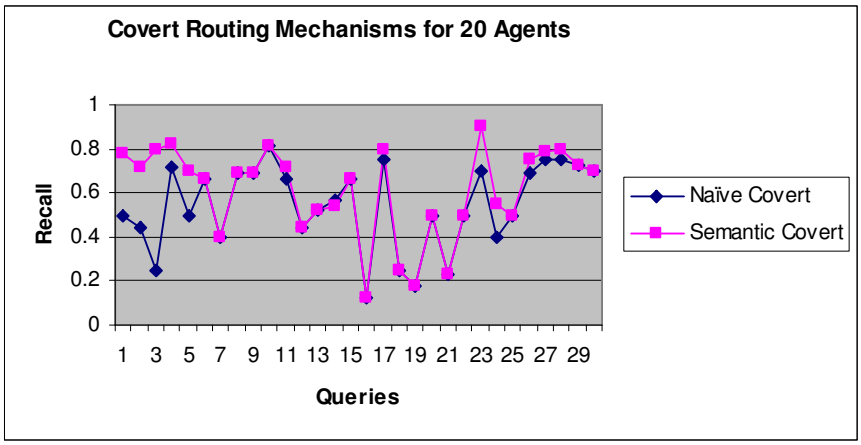


Figure 22 Naive and semantic covert routing for 20 agents network

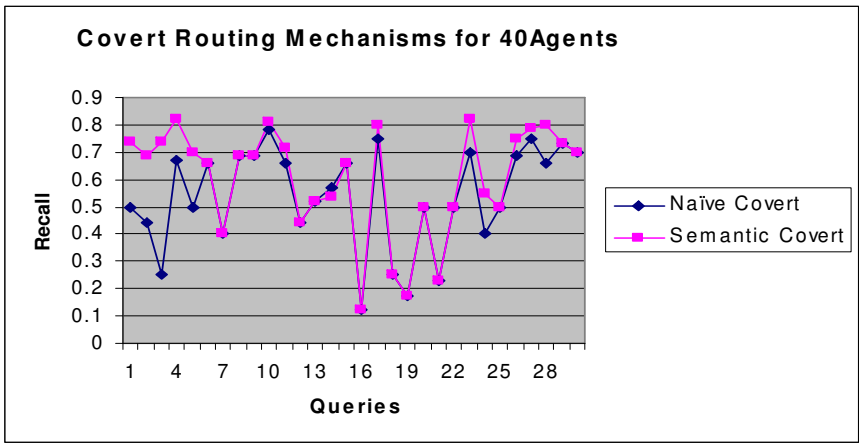


Figure 23 Naive and semantic covert routing for 40 agents network

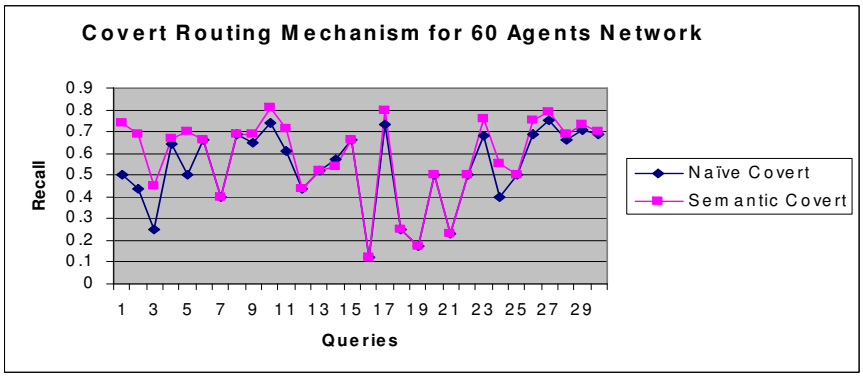


Figure 24 Naive and semantic covert routing for 60 agents network

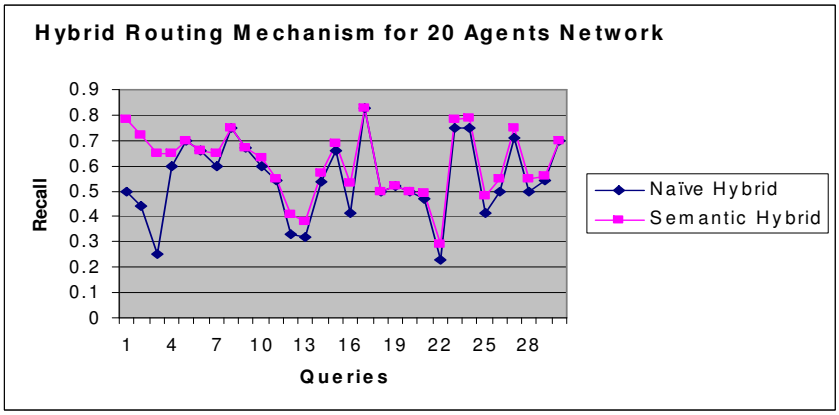


Figure 25 Naive and semantic hybrid routing for 20 agents' network

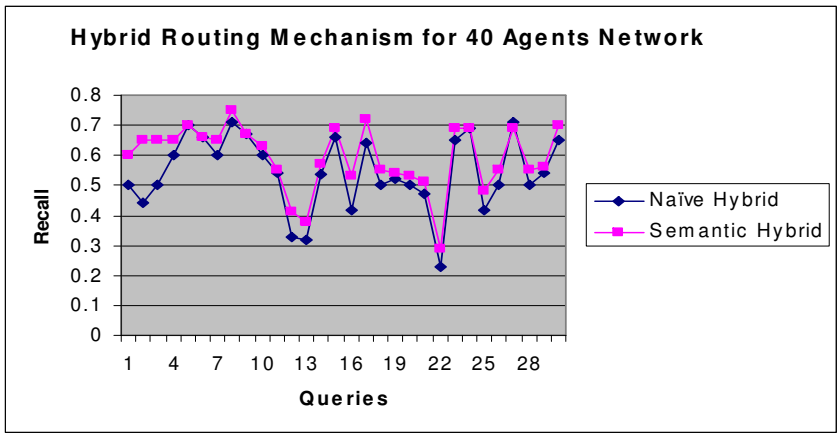


Figure 26 Naive and semantic hybrid routing for 40 agents' network

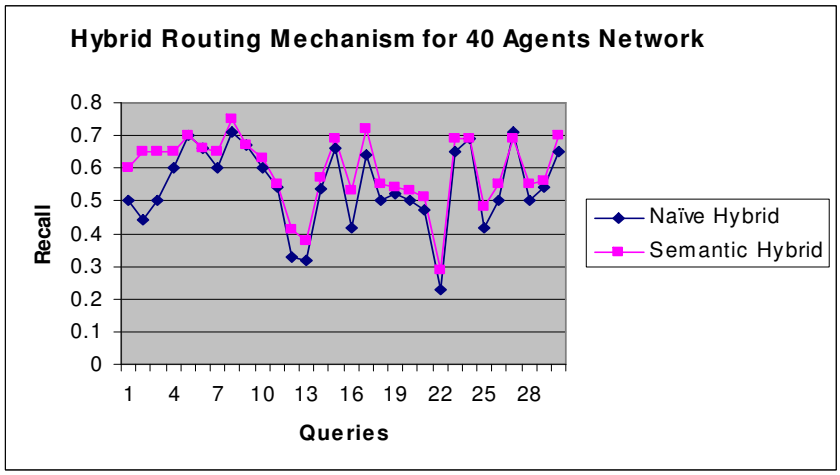


Figure 27 Naive and semantic hybrid routing for 60 agents' network

In both cases above, we have seen that the hybrid routing mechanism (cf. section 4.4.3) involves less messages and the semantic algorithm (section 4.10) gives more recall compared to the naïve algorithm, so we select the semantic hybrid setting (i.e. setting 6 cf. Table 1) for the remaining evaluation. In the above scenarios we have set hop count (cf. section 2.11) limit=2. Now we study effect of change in the recall with changing the value of hop count. We start with setting its value to 0, which means no routing at all and we continue till n=5 as shown in Figure 28.

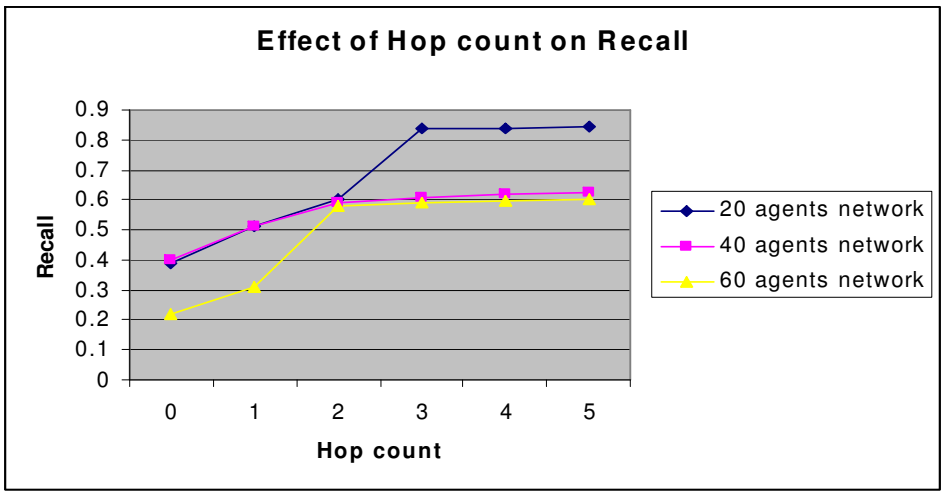


Figure 28 Recall with various hop counts in network of different sizes

Recall increases as the hop count is increased. In simulation the first agent is selected based upon the query related topic selected by the user. The above graph shows that on average 40% of the answers are coming from that first agent (hop count=0, which means no query routing at all, only local search is performed by the querying agent). At hop count values 1, 2 and 3, there is significant increase in recall values but after that recall is almost constant.

It can be seen that recall is reduced when more agents are added into the network. This is because agents are assigned randomly as agents known to others. As there are more agents in the network, so there are more chances that a particular agent has those agents as its known agents whose expertise are not the same as that of its own.

In all of above settings, we have set our selection criteria for keyword matching at 50% which means that if half of the keywords entered by the user are found in bibliographic reference, then it will be considered relevant and will be displayed to the user. However, as a user may have more strict requirements for relevance depending upon his/her needs, we have also performed a sensitivity analysis. In this analysis we have performed two more tests on the same set of queries and in the same network but this time selection criteria was 75% and 100% instead of 50%.

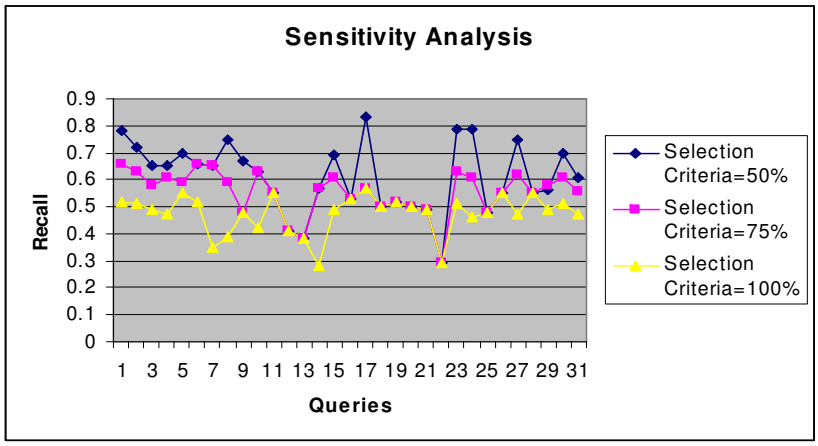


Figure 29 Sensitivity Analysis with different selection criteria's

It can be seen in Figure 29 that recall has been lowered in those cases where selection criteria are set to 75% and 100%. Number of relevant results decrease with the increase of selection percentage. So, if the agents which contain these results are not known agents of the querying agent or of agents within reach of it, then those results will not be included in the final results and recall will decrease.

We now study whether recall is improved if an agent has only those agents as its known agents whose expertise are the same as that of its own or which are closely related to its own expertise, recall can be improved.

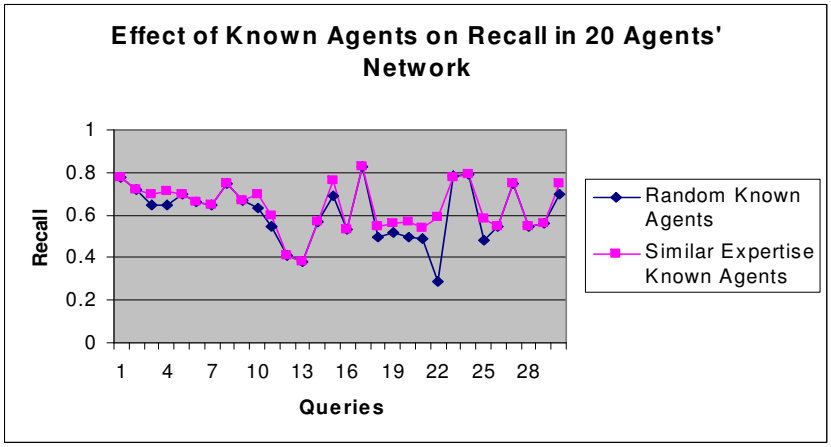


Figure 30 Recall with different known agents in 20 agents' network

There is an increase of 4% in recall in case of similar expertise known agents, which is quite significant. Recall has increased because now the querying agent and all intermediate agents are those who are likely to have the answer of the query, as

their expertise is related to the querying agent.

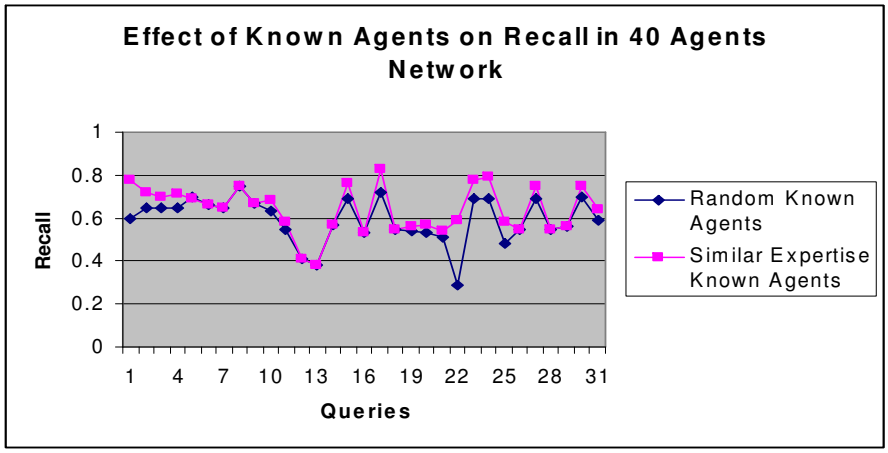


Figure 31 Recall with different known agents in 40 agents' network

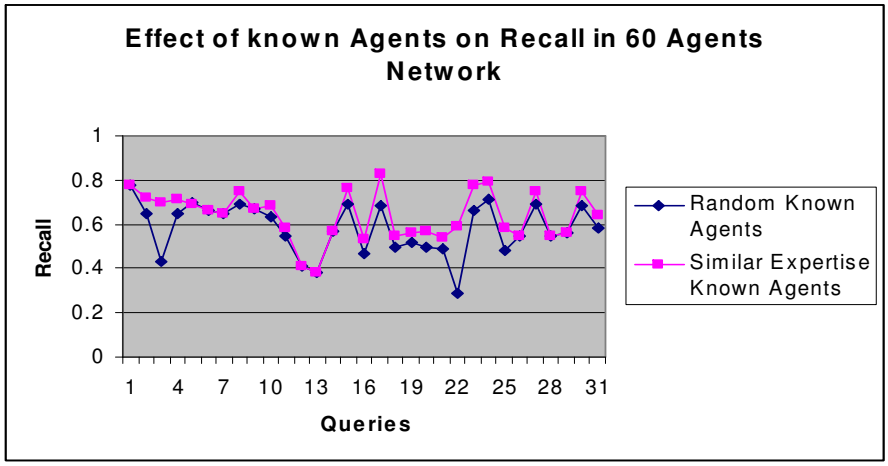


Figure 32 Recall with different known agents in 60 agents' network

It can be seen from Figure 31 and 32 that recall did not lower for similar expertise known agents when we introduce more agents in the network. This is because querying agent has not been effected by new agents because it will still forward the query to only those known agents which are related to the subject of the query. If new agent's expertise is also in the same area, then it must send an advertisement message to this agent and they both become known agents of each other. In this way they form a semantic cluster of agents with similar expertise.

From all of the above tests, we can infer the following:

- In all cases, fewer or an equal number of messages are sent across the network using the hybrid routing mechanism as compared to overt and covert routing mechanisms.
- The semantic algorithm gives better recall compared to the naïve algorithm
- Known agents with similar expertise in the semantic algorithm give better recall as compared to random known agents.

We have repeated all the above experiments for random data distribution (or proceeding data distribution). The above three statements hold true in that data distribution as well. We have not included graphs for random data distribution in this chapter because they are almost the same as the results already described. However for completeness purposes, they have been shown in Appendix A.

The only difference between the results of topic distribution and proceeding distribution is that in case of proceeding distribution naïve algorithm gives lower recall as compared to naïve algorithm recall in topic distribution.

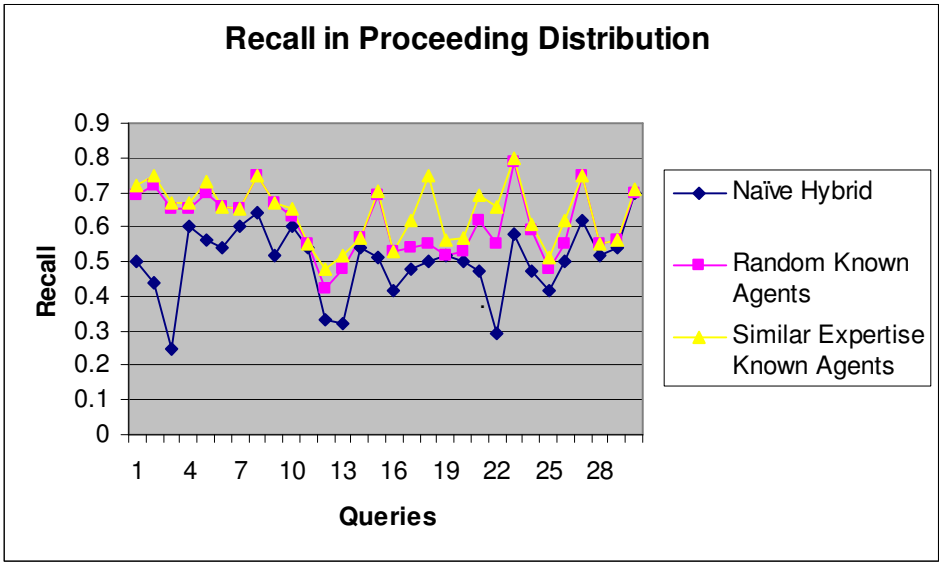


Figure 33 Recall in proceeding distribution with different settings

The reason for this reduced recall is that every agent has multiple expertise and data is

spread across the network. So, in order to get most or all of the results, only relevant agents should be selected. As in the naïve algorithm, agents are selected randomly so recall is lowered in this case. In topic distribution, as most of the results are with only one agent then if by any chance that agent is selected randomly by the naïve algorithm, then most of the results will be returned.

Chapter 8

8.Comparison with Intelligent Agents

As we have described in chapter 2 and chapter 4 that this project is carried out as a part of joint research study in which we performed a comparison of

- Advertisement based algorithms and lazy learning algorithms
- Simple agents with intelligent LCC protocols and intelligent agents.
- Different query routing mechanisms (e.g. overt, covert and hybrid)

The comparison of query routing mechanism has been discussed in chapter 7 in detail and we have seen that least number of messages is sent on the network in case of hybrid mechanism. In intelligent agents system [33] agents use hybrid mechanism for sending messages from one agent to other agent. So both systems perform equally in case of query routing mechanism. Before comparing our results with intelligent agents [33], we present a brief overview of their system.

Intelligent agents system is based on lazy learning algorithms which mean they simply store the results without performing any processing on them. In agent based P2P system, the main challenge is to identify the agent who is expected to contain the answer of the query without involving those agents which are not related to that query. The following is a brief overview of agent selection in [33]:

- Agents keep semantic reference index about other agents and update it when their knowledge about other agents changes.
- The first agent will forward the query to the content holders (if it can find any for this query). Content holders are those agents which have replied to the similar query previously.
- If the first agent is unable to find any content holder, then it will search for counselors. Counselors are those agents which acted as intermediate agents when the similar query was asked previously. So they have information about that

querying agent which received the query previously. The counselor will forward the query to the querying agent and the querying agent will forward the query to the content holders.

- If neither content holder nor counselor is found, then query is relaxed and appropriate agents are selected as per relaxed query but actual query is passed to these agents.

It is important to note that in [33], the main emphasis is laid on searching from already answered queries, while in our case as we only store agents' expertise; we select appropriate agents by looking at their expertise.

A major advantage of our approach is that we do not need to populate foreign knowledge bases for any agent, which is one of the major limitations of lazy learning algorithms. As an example, if we have a network of 20 agents and every agent is expert in only one topic in ACM hierarchy, then every agent can have a maximum of 20 entries about the expertise of other agents in the network, while in the case of lazy learning algorithms, agents need to store information about all content providers and counselors of that query. So a huge number of foreign knowledge bases will be populated at every agent.

We now compare both systems in terms of recall:

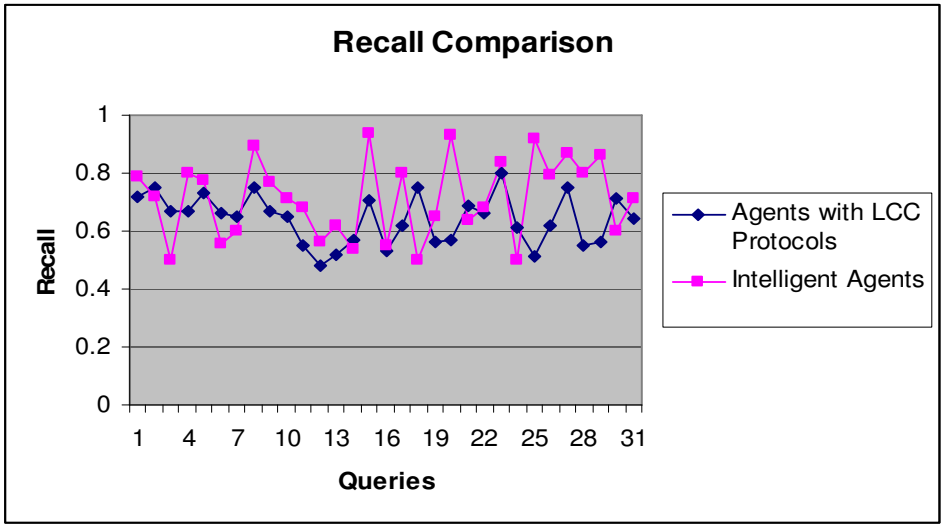


Figure 34 Recall comparison of agents with LCC protocols and intelligent agents

Our system gives recall of 64%, while intelligent agents [33] give a recall of 71%. There are two reasons for this lower recall of our system:

- In our system, the querying agent does not store the protocol executions, after returning the answers to the user, so each time our agents need to calculate the semantic similarity and forward the query to the appropriate agent, while in [33] information about content providers and counselors is stored for every query.

- As the notion of LCC is to keep protocols lightweight, we do not want to add additional clauses of searching content providers and counselors for every query at every agent as done in [33].

Chapter 9

9.Conclusion

This chapter summarizes some of the important lessons we have learned from this project:

- Simple agents with intelligent LCC protocols for query routing can work as well as intelligent agents. We demonstrate our claim that we no longer need to build intelligent but task specific agents. Instead simple agents with task specific protocols can work effectively.

- Agents' selection on the basis of their expertise gives better recall of answers as compared to the random selection of agents.

- Storing knowledge about the expertise of other agents is sufficient and agents do not need to store the actual answers (which results in generating major chunks of foreign knowledge bases).

- Sending advertisement about the expertise of an agent, when it joins the network, of those agents with which it can communicate is better than learning by observation, because in that case a lot of queries need to be flooded on the network so that agents can learn about the expertise of other agents.

- Use of ontologies and semantic algorithms provide an “added value” in our system and similar applications (e.g. Bibster [27]) by introducing more sophisticated search capabilities [26].

- By grouping agents of similar expertise together in the network and creating a semantic cluster of agents of similar expertise, recall can be improved and the number of messages sent across the network can be lowered [26].

- Expertise-based agent selection combined with ontology-based matching performs better than random agent selection and searches based on exact matches [26].

Chapter 10

10.Future Work

There are many fruitful directions in which this work can be enhanced:

- The LCC interpreter which we have developed in this project has limited capabilities. This interpreter could be enhanced so that it can deal with every type of LCC protocol instead of only query routing.

- Currently the LCC interpreter is embedded into the network simulator and cannot be used separately. This interpreter should be made independent of the simulator so that it can be used for other purposes as well.

- Another improvement would be to enable agents will be able to store the whole execution of protocols and next time when it receives the same query they will be able to determine from previously executed protocols that which agents have answered the query last time and simply pass the query to them.

- Agents might be made more selective so that they can discard any advertisement message which is not related to their own expertise. Currently our agents accept all the advertisements and they are not capable of discarding those advertisements which are not related to their own expertise.

- Currently agents in our system only understand ACM topic hierarchy, which is fairly simple and works well for bibliographic scenario. Other domains may require more complex ontologies. So agents might be enhanced to be able to incorporate more ontologies.

- To verify the results of our simulation in the real world, these protocols and agent selection algorithms need to be tested in real applications.

Appendix A-Random Distribution Graphs

As it has been discussed in chapter 7 that experiments with random data distribution (cf. section 5.8.2) yields almost the same results as that of topic distribution (cf. section 5.8.1) except for those which have been discussed in chapter 7. However for the sake of completeness purpose, all the graphs with random data distributions are shown in this section:

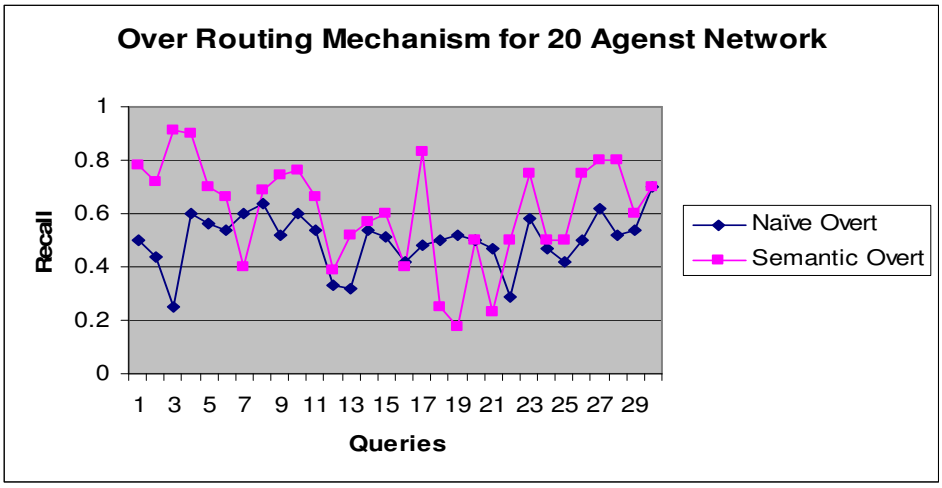


Figure 35 Naive and semantic overt routing mechanism for 20 agents' network

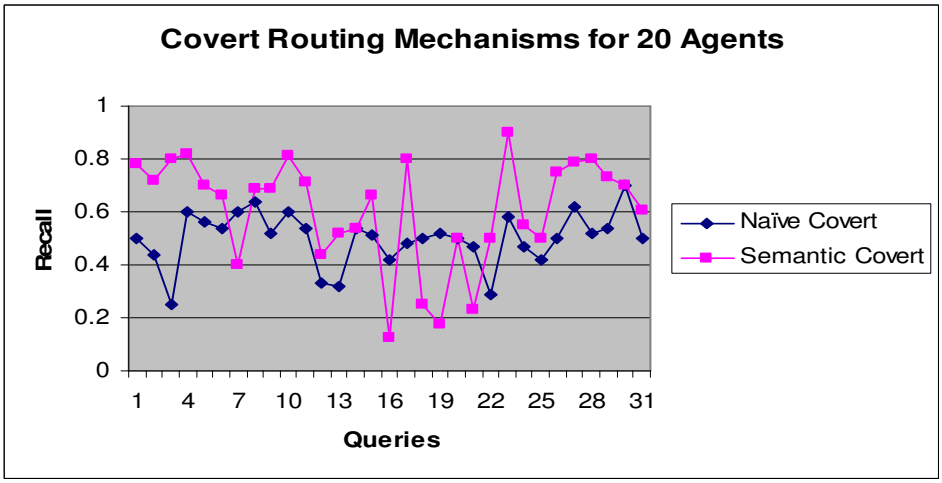


Figure 36 Naive and semantic covert routing mechanism for 20 agents' network

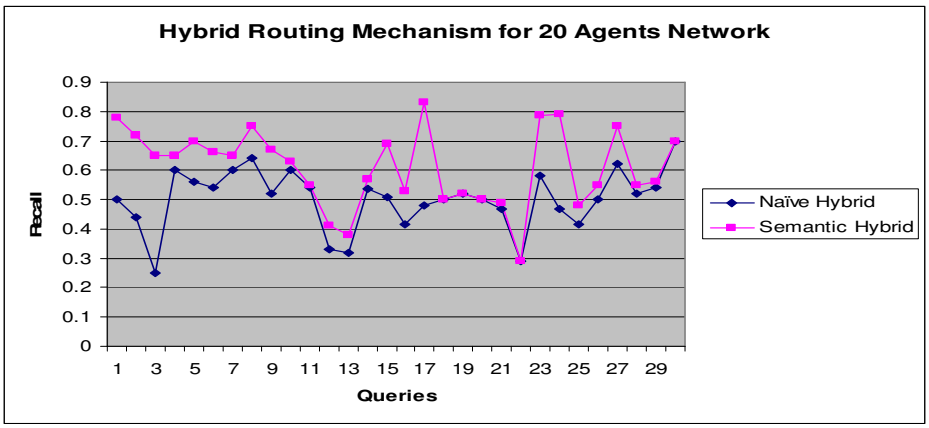


Figure 37 Naive and semantic hybrid routing mechanism for 20 agents' network

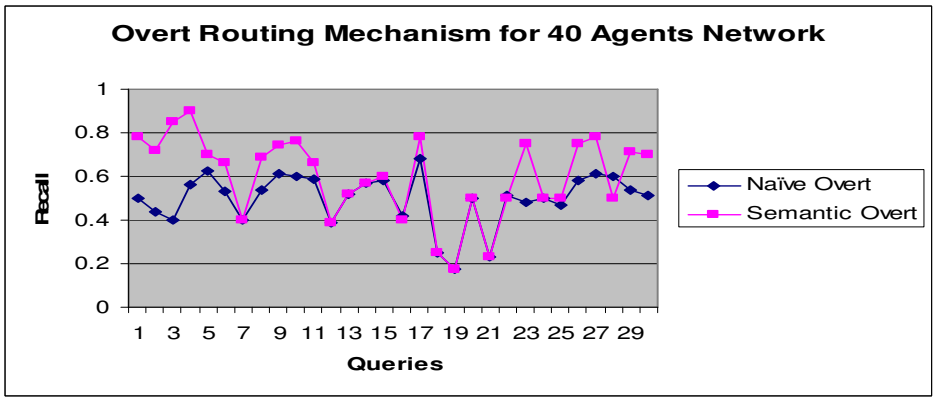


Figure 38 Naive and semantic overt routing mechanism for 40 agents' network

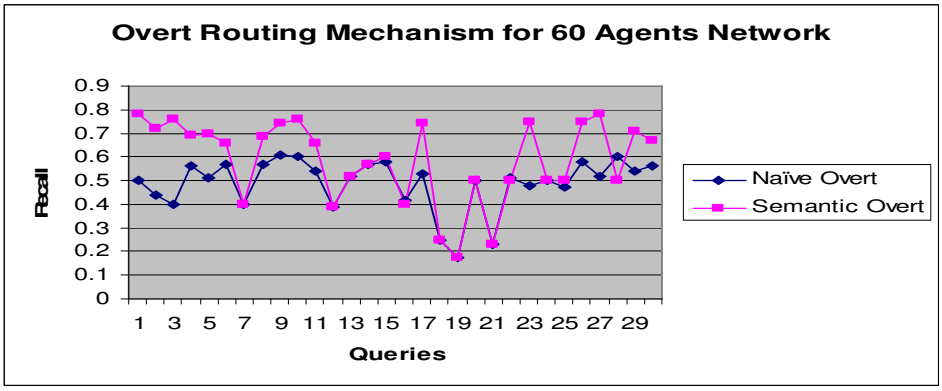


Figure 39 Naive and semantic overt routing mechanism for 60 agents' network

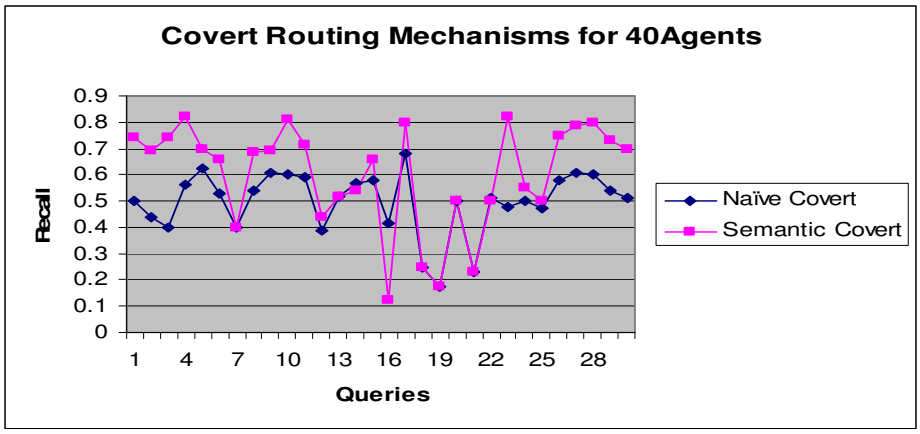


Figure 40 Naive and semantic covert routing mechanism for 40 agents' network

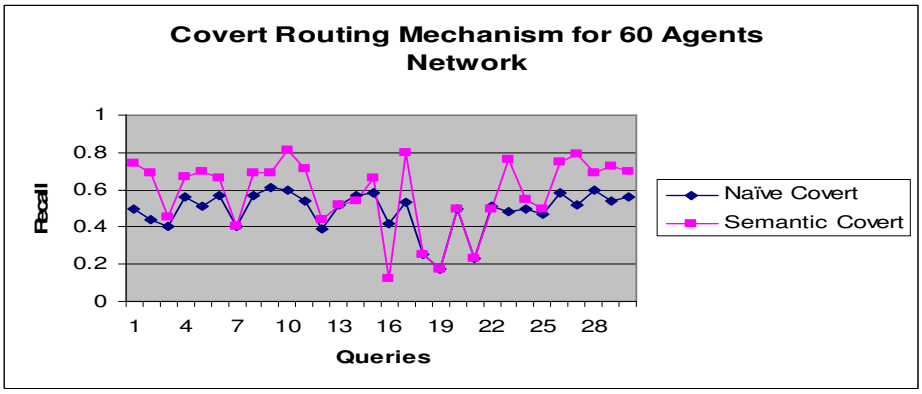


Figure 41 Naive and semantic covert routing mechanism for 60 agents' network

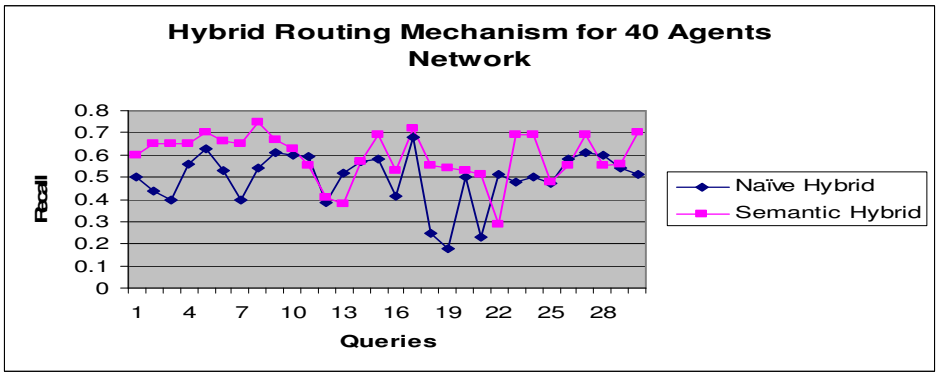


Figure 42 Naive and semantic hybrid routing mechanism for 40 agents' network

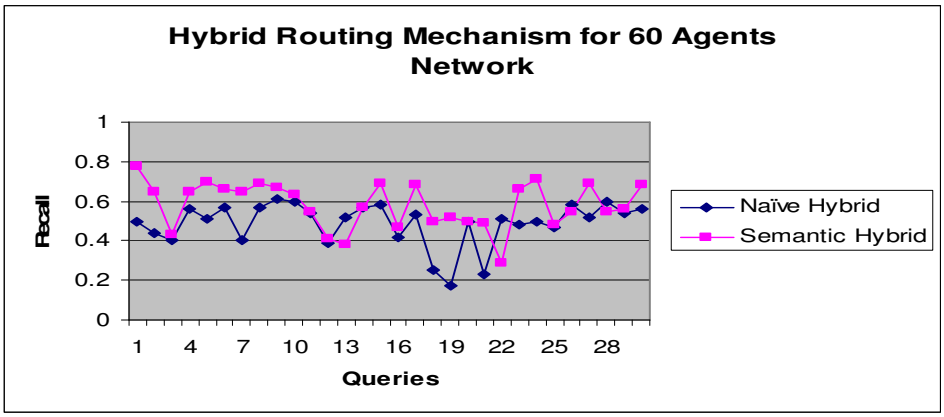


Figure 43 Naive and semantic hybrid routing mechanism for 60 agents' network

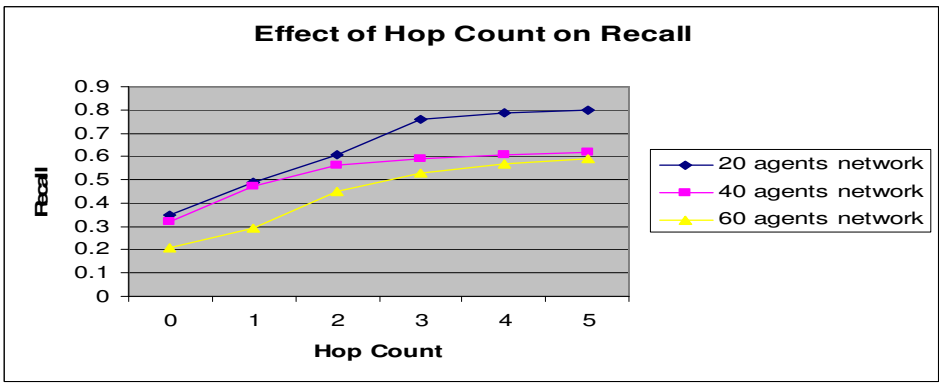


Figure 44 Recall with various hop counts in network of different sizes

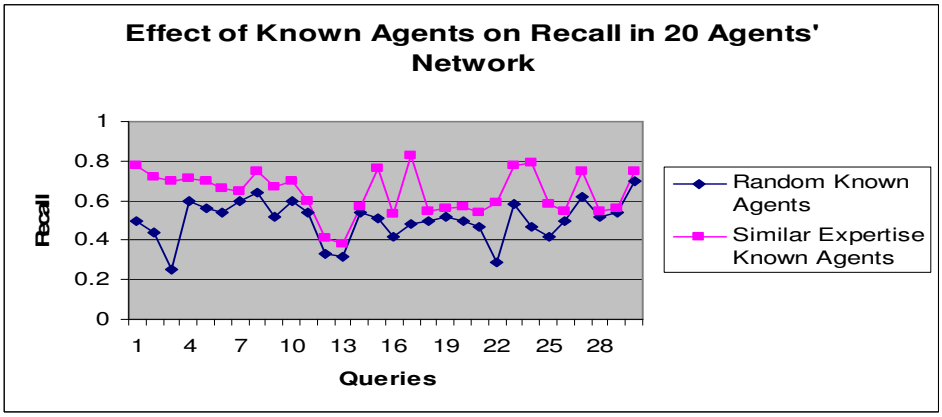


Figure 45 Recall with different known agents in 20 agents' network

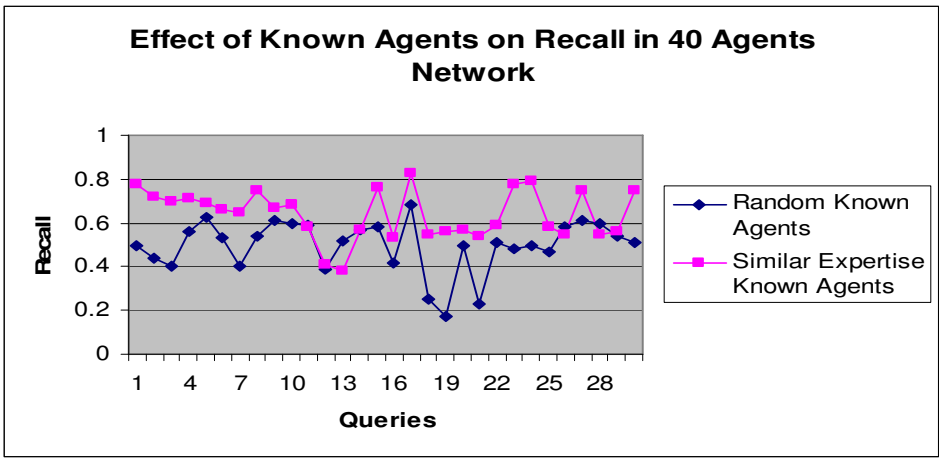


Figure 46 Recall with different known agents in 40 agents' network

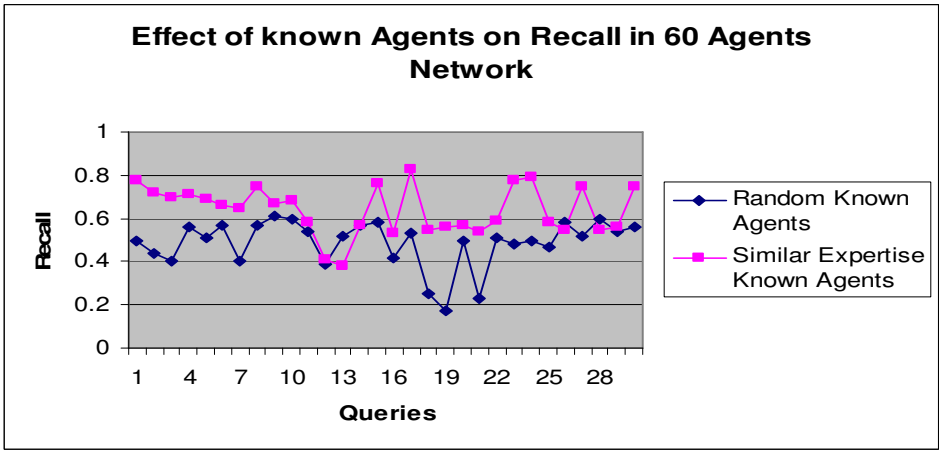


Figure 47 Recall with different known agents in 60 agents' network

References

[1] Fensel, D., Wahlster, W. and Lieberman, H. 2002. Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. MIT Press.

[2] Liu, L., Pu, C., Buttler, D., Han, W., Paques, H., and Tang, W. 2000. AQR-toolkit: an adaptive query routing middleware for distributed data intensive systems. In Proceedings of the 2000 ACM SIGMOD international Conference on Management of Data (Dallas, Texas, United States, May 15 - 18, 2000). SIGMOD '00. ACM Press, New York, NY, 597. DOI= http://doi.acm.org/10.1145/342009.336579

[3] Yang, B. and Garcia-Molina, H. 2002. Improving Search in Peer-to-Peer Networks. In Proceedings of the 22 Nd international Conference on Distributed Computing Systems (Icdcs'02) (July 02 - 05, 2002). ICDCS. IEEE Computer Society, Washington, DC, 5.

[4] Menascé, D. A. 2003. Scalable P2P Search. IEEE Internet Computing 7, 2 (Mar. 2003), 83-87. DOI= http://dx.doi.org/10.1109/MIC.2003.1189193

[5] Klusch, M. 2001. Information agent technology for the Internet: a survey. Data Knowl. Eng. 36, 3 (Mar. 2001), 337-372. DOI= http://dx.doi.org/10.1016/S0169-023X(00)00049-5

[6] Wooldridge, M. 2002. Intelligent Agents: The Key Concepts. In Proceedings of the 9th ECCAI-ACAI/EASSS 2001, AEMAS 2001, HoloMAS 2001 on Multi-Agent-Systems and Applications Ii-Selected Revised Papers V. Maríık, O. Stepánková, H. Krautwurmova, and M. Luck, Eds. Lecture Notes In Computer Science, vol. 2322. Springer-Verlag, London, 3-43.

[7] Camorlinga, S., Barker, K., and Anderson, J. 2004. Multiagent Systems for resource allocation in Peer-to-Peer systems. In Proceedings of the Winter international Synposium on information and Communication Technologies (Cancun, Mexico, January 2004). ACM International Conference Proceeding Series, vol. 58. Trinity College Dublin, 1-6.

[8] Moro, G., Ousel A. M., Sartori C. 2002. Agents and peer-to-peer computing: A promising combination of paradigms. In *Proceedings of AP2PC-02, Bologna, Italy*. pages 1–14. Springer-Verlag LNCS 2530.

[9] Robertson, D. 2004. Multi-agent coordination as distributed logic programming. 20th International Conference, ICLP 2004, Saint-Malo, France.

[10] Russell, S. J. and Norvig, P. 2003 Artificial Intelligence: A Modern Approach. 2. Pearson Education.

[11] Stone, P. and Veloso, M. 2000. Multiagent Systems: A Survey from a Machine Learning Perspective. Auton. Robots 8, 3 (Jun. 2000), 345-383. DOI= http://dx.doi.org/10.1023/A:1008942012299

[12] Special issue on Intelligent Agents, Comm. ACM, vol. 37, no, 7, July 1994.

[13] Wooldridge, M. and Jennings, N.R. Intelligent agents: Theory and practice. The Knowledge Engineering Review 10, 2 (1995), 115–152.

[14] Woolridge, M. and Wooldridge, M. J. 2001 Introduction to Multiagent Systems. John Wiley & Sons, Inc.

[15] Mayeld, J., Y. Labrou, and T. Finin: 1996, Evaluating KQML as an Agent Communication Language. In: M. Wooldridge, J. P. Muller, and M. Tambe (eds.): Intelligent Agents II (LNAI Volume 1037). Springer-Verlag: Berlin, Germany, pp. 347-360.

[16] FIPA Foundation for Intelligent Agents. FIPA Specification part II agent communication language. April 1999.

[17] Esteva M., Rodriguez J.A., Arcos J.L., Sierra C., Garcia P. (2000); Formalising Agent Mediated Electronic Institutions, Catalan Congres on AI (CCIA 00), pp. 29-38.

[18] Robertson D. 2002 Distributed Agent Dialogues. Edinburgh University.

[19] Milner, R., 1980. Lecture Notes in Computer Science Vol 92*:* A Calculus of Communicating Systems. Springer-Verlag

[20] Adamic, L. A., Lukose, R. M., Puniyani, A. R., and Huberman, B. A. Search in Power-law Networks. Physical Review E 64 (2001).

[21] Tempich, C., Staab, S., and Wranik, A. 2004. Remindin': semantic query routing in peer-to-peer networks based on social metaphors. In Proceedings of the 13th international Conference on World Wide Web (New York, NY, USA, May 17 - 20, 2004). WWW '04. ACM Press, New York, NY, 640-649. DOI=

http://doi.acm.org/10.1145/988672.988759

[22] Walton, C. 2004. Multi-Agent Dialogue Protocols. Proceedings of the 8th International Symposium on Artificial Intelligence and Mathematics

[23] Nejdl, W., Wolpers, M., Siberski, W., Schmitz, C., Schlosser, M., Brunkhorst, I., Loser, A. 2003.: Super-Peer-Based Routing and Clustering Strategies for RDF-Based P2P Networks. In Proceedings of the 12th International World Wide Web Conference (WWW), Budapest, Hungary.

[24] Kautz, H., Selman, B., and Shah, M. 1997. Referral Web: combining social networks and collaborative filtering. Commun. ACM 40, 3 (Mar. 1997), 63-65. DOI= http://doi.acm.org/10.1145/245108.245123

[25] Yolum, P. and Singh, M. P. 2003. Dynamic communities in referral networks. Web Intelli. and Agent Sys. 1, 2 (Dec. 2003), 105-116.

[26] Haase, P.; Siebes, R.; and van Harmelen, F. 2004. Peer selection in peer-to-peer networks with semantic topologies. In International Conference on Semantics of a Networked World: Semantics for Grid Databases, 2004, Paris.

[27 Broekstra, J., Ehrig, M., Haase, P., van Harmelen, F., Kampman, A., Sabou, M., Siebes, R., Staab, S., Stuckenschmidt, H., and Tempich, C. 2003. A metadata model for semantics-based peer-to-peer systems. In Proceedings of the 1st Workshop on Semantics in Peer-to-Peer and Grid Computing at the 12th International World Wide Web Conference. Budapest, Hungary..

[28] The Semantic Web Research Community Ontology.

http://ontobroker.semanticweb.org/ontos/swrc.html.

[29] The ACM Topic Hierarchy.

http://www.acm.org/class/1998/.

[30] How Big Is The Internet? http://metamend.com/internet-growth.html

[31] So, E., Collins M. (2002). P2P Search Engines.

http://ntrg.cs.tcd.ie/undergrad/4ba2.02-03/p8.html

[32] Koubarakis 2003. MultiAgent Systems and PeertoPeer Computing: Methods, Systems, and Challenges. Invited Talk in 7th Int. Workshop on Coop. Information Agents, Finland.

[33] Elahi, S. 2005. Semantic Query Routing in agent-based P2P systems. Thesis (MSC). University of Edinburgh.

[34] Li Y., Bandar Z, and McLean D. An approach for measuring semantic similarity between words using multiple information sources. IEEE Trans. on Knowledge and Data Eng., 15(4):871--882, 2003.

[35] Rada R, Mili H., Bicknell E. and Blettner M., “Development and application of a metric on semantic nets IEEE Trans. Systems, Man, and Cybernetics, Jan./Feb. 1989, vol. 19 , no. 1, pp. 7-30.

[36] How to Classify Works Using ACM? Computing Classification System http://acm.org/class/how_to_use.html

[37 Ehrig, M., Schmitz, C., Staab, S., Tane, J., Tempich, C. 2003.: Towards evaluation of peer-to-peer-based distributed knowledge management systems. In: Proceedings ofthe AAAI Spring Symposium “Agent-Mediated Knowledge Management (AMKM-2003)”.

[38]Crestani, F. and Rijsbergen, C. J. 1997. A Model for Adaptive Information Retrieval. J. Intell. Inf. Syst. 8, 1 (Jan. 1997), 29-56. DOI= http://dx.doi.org/10.1023/A:1008601616486

[39] Aha, W. 1998. "Feature weighting for lazy learning algorithms." Feature Extraction,

Construction and Selection: a Data Mining Perspective, edited by H. Liu and H. Motoda. Norwell, MA: Kluwer.

[40] Ehrig, M., P. Haase, et al. 2003. "The SWAP Data and Metadata Model for Semantics-Based Peer-to-Peer Systems." Lecture Notes in Computer Science 2831: 144-155.

[41] Brunskill, E. 2001. Building Peer-to-Peer Systems with Chord, a Distributed Lookup Service. In *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems* (May 20 - 22, 2001). HOTOS. IEEE Computer Society, Washington, DC, 81.

[42] Barker, A.D. 2004. Coordination is the key. Thesis (MSC). University of Edinburgh.

[43] Esteva, M., Rodriguez-Aguilar J. 2000. Institutionalizing Open Multi-Agent Systems. In *Proceedings of the Fourth international Conference on Multiagent Systems (Icmas-2000)* (July 10 - 12, 2000). ICMAS. IEEE Computer Society, Washington, DC, 381.

[44]Robertson D. 2004. A lightweight method for coordination of agent oriented web service. *In Proceedings of AAAI Spring Symposium on Semantic Web Services, California, USA*.

[45] Walton, C. D., and Robertson, D. 2002. Flexible Multi- Agent Protocols. In Proceedings of UKMAS 2002. Also published as Informatics Technical Report EDI-INF-RR-0164, University of Edinburgh.