An Agent - Based Technique For Network - Wide File Search In Heterogeneous Peer - To - Peer Systems

T.Amudha[1] Manu Rajan Nair[2]

Abstract

Heterogeneity is a major characteristic of all modern computer networks, as a consequence of the proliferation of different types of Operating Systems (OS) acting as nodes in such an environment. This presents unique problems to common activities such as locating resources on a network. Agent computing presents us with several solutions when combined with the Peer-to-peer networking to these resource discovery problems. We approach this issue by presenting the solution to a seemingly innocuous problem, i.e., locating a particular file or files that match a given regular expression on the available network.

Keywords: Agents, peer-to-peer system, multi-agent system, heterogeneous network

1. Introduction

Search problem Scenario

To define this problem, the problem space is defined, here referred to as its "Search Space". The search space when defined with respect to a single node in a network is called the "Local Search Space" (LSS), which contains all searchable files and folders on that node for a normal user. The "Global Search Space" (GSS) is defined as all files and folders on that network that are search-able by a normal user on any node on the network. The LSS and GSS may vary from node to node depending on the user on that node. Further the GSS is an extremely dynamic environment with various nodes connecting/ disconnecting from the network and the files/folders on the nodes varying, as the files/folders are added/ removed/changed. Further there are limitations on the files/folders accessible from a given node due to security restrictions imposed on the network. The inherent dynamism and unpredictability of the problem space demands a solution that may be well based on Agent computing and Peer-to-peer networking.

The standard file searching solution available on most OS is simplistic and assumes that the user has knowledge about locating the file and provides a "search term" that is reliable enough, further it is assumed that the user is aware about the limitations imposed by the network. Simply stated, the user knows what search term will help him to locate a particular file and that the file is located in an accessible location. This general simplification applied to the problem may be effective for searching the LSS but may fail in the larger context of the GSS. This "Goal-directed" behavior presents us with the opportunity to select a solution that best reflects the problem, in this case "Agent Computing".

Agent Computing Paradigm

Agents can be described as a small program running on a node tasked with a specific purpose. The agent is singularly responsible for finishing the task, with several

[1] Lecturer ,[2] Research Scholar,School of Computer Science and Engineering, Bharathiar University, Coimbatore-641046, Tamil Nadu. E-mail: amudha.swamynathan @gmail.com, manurajan10@yahoo.co.in

other agents with other assigned tasks executing simultaneously on that node. The agent is independent, self-regulating and goal-oriented; it may communicate with other agents on the node or network. Functional autonomy being one of the defining features of agent paradigm, social-communication is another. Agent autonomy implies that the entity can observe its environment and act upon it in a fashion to further the goal assigned to it, without explicit directives from a user. Here the user is expected to only set the goal to the agent; the agent by it has to figure out ways to reach that goal. Agents may be Mobile or Static, intelligent or dumb.

Peer-to-peer and Agent Based Solutions

Peer-to-peer (P2P) systems are normally characterized by decentralized control, large scale and extreme dynamism of their operating environment. Such P2P networks may be of several types, such as self-regulating centralized node tracking based systems with limited intelligence used in P2P file-sharing tools such as Gnuttella and Bit-torrent networks, or intelligent distributed node tracking based systems such as Anthill.

All existing models based on P2P Agents can be broadly classified along traditional lines as Static or Mobile and further as intelligent or dumb, but all of them still use the central-tracking model for locating and tracking the peers and the publish/subscribe model for exchanging data. Taken together these provide several advantages such as lower communication overheads in locating resources or services and exchanging data. But such systems compromise on critical Agent-computing features such as Agent autonomy by stipulating that an Agent should be part of a peer network only after it subscribes to another published agent on that network. Furthermore such agents need to be user-pre-configured or pre-programmed to handle interaction between peers thereby increasing maintenance requirements for the whole system. Several solutions have been implemented to solve the above problems, one of them is the usage of a Client-Server based peer tracking system, and another method is the use of mobile agents.

2. Methodology

In the context of the Search Problem, an agent should display goal-directed behavior; hence only intelligent, static agents are considered for the solution. With respect to the Search problem, generally we assume that a static agent is available on each node with access to the LSS on that node; the GSS is made up of all LSS accessible to all the agents on each of the respective nodes. The network is assumed to allow some form of datagram communication among the agents.

In this Agent computing based solution, the Initializer agent is set the goal of locating matches to the search term from the GSS. The Initializer agent communicates this goal to other Search agents in the network using datagram broadcasting, the other agents on receiving the broadcast can "choose" to execute the search request, and in doing so act as "Search Executers" or "Executers".

The user is expected to pass a "search term" which may be a file name that the user needs to locate or a regular expression to a particular set of files to the Search Agent on that node defined as the "search manager" or "Initializer". The goal of the Initializer is to return a list of matching locations on the GSS to the user in the smallest possible time. To define a performance metric, we use the First Positive Response Time (FPRT) defined as the time required by an agent to return the first matching location on the GSS. In these normal search methods, the FPRT depends on several factors such as size of GSS, communication delay between nodes and most critically,

the number of nodes in the GSS. The FPRT is essentially the sum of all the LSS FPRTs with respect to the Initializer node. This method ignores the parallelizability of the problem; the Initializer node has to search the LSS on each node serially.

3. Specific Goals

The specific goals of the system are

- **To develop a File search application** that
  - Allows searching for specific files/ folders on the entire network
  - Allows searching for files/folders matching a pattern on the entire network
  - Display results for a given search from the entire network.
- **Robustness**
  - System is not adversely affected by network communication
  - System is not adversely affected by individual node breakdown.
- **Efficiency**
  - System gets maximum positive responses in minimum time
  - Concise communication (very small communication overhead)
- **Reliability**
  - Results are always correct
  - No redundancy

4. Features Of the Approach

It is obvious that any solution to the Search Problem in a network must have the following features:

1. Low maintenance overhead

There must be minimum pre-requisites to setup and install the agents.

2. Low Communication overhead

Inter-agent communication must take up minimum bandwidth.

3. Best-case FPRT

Search results must be returned to the Initializer with minimum delay.

With the above aims, an Agent-based peer-to-peer application was developed in JAVA. This uses datagram/ UDP protocol for inter-agent communication (ensuring minimum communication overhead), using features of the JAVA environment to ensure low maintenance costs and implemented machine learning based intelligence on the agents to optimize searching to improve the FPRT. Datagram based communication is the basis for most modern Agent and P2P based systems, as this allows for minimum bandwidth usage on a network. Since in the proposed system, all communication between agents is in the form of single-line text messages, UDP-protocol based messaging system gives the best solution. The platform independent JAVA environment provides the application with a uniform application environment protecting the agent from variations in OS implementations.

The critical aim of the application is to achieve minimum FPRT for each agent on a given GSS; hence optimizing the search at both ends, i.e. the Initializer and executor becomes necessary. Since the OS and file-systems impose limits on the efficiency of search strategies, other methods are necessary to improve the FPRT. The executor and Initializer both implement a simple machine learning based intelligence, that allows the agent to optimize the "search term" based on previous experience for the Initializer and use optimized search strategy for the executor also based on learned knowledge. This introduces the concept of Agent "aging" whereby an agent when installed is an

"infant" with no knowledge of its LSS or GSS. As the agent participates in the search process either as Initializer or executor, it gains experience by relating search terms to search results for its LSS, hence increasing in "maturity".

A mature agent executor, in theory can return a search result to an Initializer agent with an FPRT of nearly zero. Further we use the concept of "social communication" among the mature agents to share the knowledge gained with "younger" agents on its GSS. This allows a GSS to mature overtime and lowers the FPRT for all searches which in turn leads to lower bandwidth usage and maintenance costs. The system consists of a static intelligent Initializer agent; a static intelligent executor agent and a UDP based communication protocol.

The executor shows certain qualities of mobile agents as the various agents exchange experience with each other. Hence an agent based P2P JAVA application allows the design of a robust and efficient network-wide search solution which can then be extended to cover other functionality such as network resource management and monitoring.

5. Operational Parameters Of The System

The environment in which the system operates can be divided into two levels, Files and Folders on the local node and Files and Folders on all other accessible nodes. Local file searches are limited only by node efficiency, but for searching other nodes, efficiency varies greatly depending on various unpredictable factors such as network traffic, network security limitations and communication reliability.

As the agent matures with every search that it participates in, the search accuracy increases with increasing maturity. Consequently, even though at the beginning, search results may not reflect all the matches, it is more than compensated in later stages by the increased search efficiency that is far greater than traditional approaches to the system.

The solution is based on the platform independent JAVA framework, containing an Agent-based application that clearly follows the Agent paradigm. The solution consists of several similar agents installed on various nodes in a network; each node solution consists of a Search gateway and Search Agent. A user at any one of these nodes can use the gateway to search for files on all the nodes in the solution. A Search-Agent communications protocol is implemented on top of the UDP Datagram protocol system in order to achieve maximum efficiency in data communication. The various agents are completely autonomous, with each deciding for it how it goes about efficiently searching its own node. This acts like a distributed application, but it displays a communal intelligence by working towards a common aim.

6. Agent Design

The overall system organization is given in the following figure:

The main module is the top-most module and acts as the system initializer. The search agent module represents the actual agent that executes the search, on initializing it waits listening for other agents broadcasting search queries, on receiving a query it evaluates the best strategy for executing the search. The search gateway module acts as a gateway to the GSS of the solution, where all search terms are entered and answers sought. The gateway on the solution is completely decoupled from the underlying search agent so as to allow the agent to be truly independent. The communications module provides the Listener for both single and multi-casting; the queries are passed to the Search agent and results to

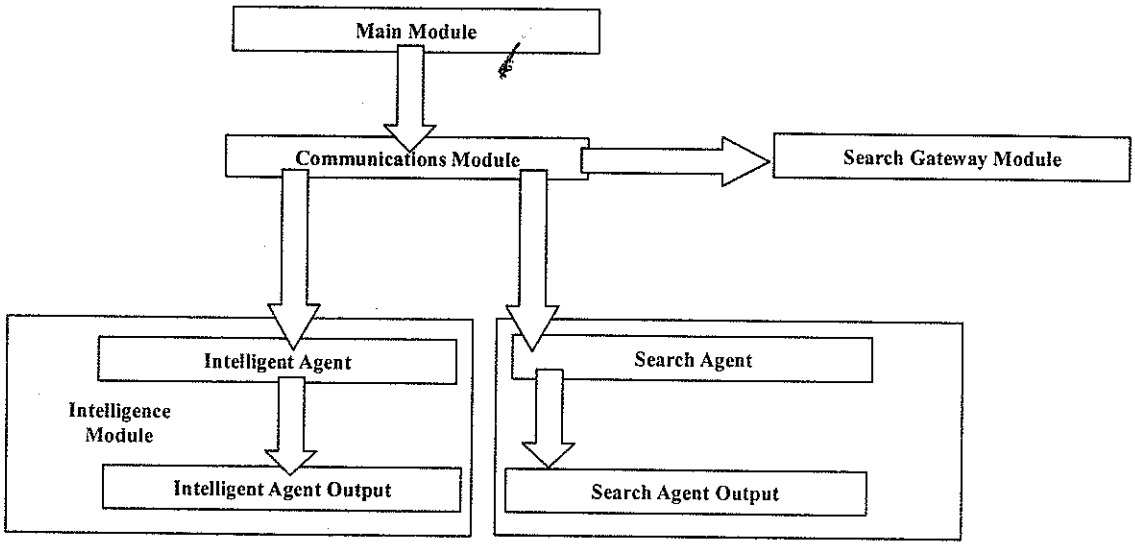

Figure1: System Organization

the Search gateway. The communications module processes the search information cache and derives generalizations for search patterns, when the agent is idle; it tries to share the learned knowledge with other agents. The Search Agent, Gateway and Intelligence agents act as the recipients, whereby messages with the search query tag are passes to the search agent, display result tags to search gateway for displaying and Intel exchange tags to Intelligence module.

7. Search Strategies

The Search Agent class uses Native JAVA file I/O functions to search the file system or the Local Search Space LSS for matches to the given search term. It utilizes the JAVA regular expression model to ascertain matches with the search term. The LSS search strategies are of three types

1. Hard Search
2. Directed Search
3. Directed Hard Search

In Hard Search, the entire LSS of the Node is searched in a Breadth first algorithm for locating matches, i.e., all directories from the top level onward are searched recursively going down the directory tree.

In Directed search, only the given directory is searched for matches. In Directed hard search only the specified levels starting from the given directory down are searched.

The selection of the strategy for locating a file depends on the result of evaluation conducted on the search term by the agent. The SearchAgent first tries to match the search term with any axioms in the Axiom database, on locating a match; it uses the directed or directed hard search strategy. If no matches are found, it resorts to the hard search strategy. If the axiom corresponds to an earlier successful search, then the directed search strategy is used to validate the location, else if the axiom corresponds to a search directive, then directed hard search is used to search the directories below it.

When a "Hit" is achieved, i.e., a positive match is located during search, the Search Agent or executor sends this information back to the Initializer agent using a uni-casted packet. This forms the heart of the implementation

model used by this solution. The query from another agent is the trigger that initializes the search; each hit is then sent back to the Initializer for display.

The intelligent agent based solution for locating files on an enterprise-wide network is developed using JAVA. It leverages JAVA technologies to develop a datagram protocol implemented on top of JAVA UDP protocol communication. It also uses the XML file format as the backend for structured data storage, analysis and retrieval. The XML file format based backend is required as it allows the developer to define a database-like format for data storage and retrieval, separating the database logic from the program logic. This thin-database allows user to store concise amounts of data without redundancy. JAVA provides user with functions that allow the developer to easily add, remove or update information in an XML file.

The Search Strategy Database is physically an XML file containing structured data, each node is made up of an axiom, and each axiom is a valid association between a search term and a location on the LSS. Each axiom also contains a value describing its strength as an integer, greater than or equal to zero. All axioms have an initial value of 1, every time a positive validation of the axiom is done; the current value is multiplied by 2, Every negative validation decrements the value by 1. When an axiom value on validation is found to be zero, it is deleted from the database.

8. Results From Testing And Analysis Of GSS Using FPRT

The various agents together form a Global Search Space (GSS) that can be searched from any agent on that GSS. The agents allow search execution in a distributed manner, where each agent is tasked with searching its own LSS. The main aim of the system is the reduction of the time required to locate the minimum amount of matches, i.e., the solution is designed for maximum sensitivity and responsiveness. The peer-to-peer system allows the solution to act in a one-to-one communication model, where no other intermediates are involved. This allows the solution to provide an interactive search system where a user can directly be aware of the goings on in the GSS. This further allows the minimum overhead in communication within the GSS.

Intelligent or "smart" agents display an Artificial Intelligence (AI), which is used to increase the efficiency of the solution. Searching can be made more efficient by using smart agents, which can interpret the cache to derive axioms. This derivation of axioms allows the creation of a much smaller search database, which does not depend on the number of searches. This also implies that axioms can grow in strength with increasing number of searches conducted on the GSS. The agents are allowed to share this information with other willing agents, thereby allowing a new agent that becomes a part of the GSS to quickly build up its search database without having to wait to take part in search execution.

The problem space dealt with by the solution is extremely dynamic and unstable; it consists of various nodes, each with its own local terrain defining its LSS. The sum of all these LSS form the GSS, the solution acting as a system that allows the user to locate any file that belongs in the GSS from any node attached to the GSS. Since the structure of the LSS and the search term that is being used are a subjective criterion, i.e., they may vary from system to system and network-to-network, any performance metrics based on them may provide values that cannot be replicated at another location. Hence a new metrics based on the FPRTs suggested as an

objective criterion for evaluating the behavior and performance of this solution.

The Search Gateway calculates the First Positive Response Time where the search is initialized. The FPRT for each agent on the GSS is calculated and displayed at the end of the search. Initially, The FPRT values for a GSS made up of infant agents is very high, as all of them have to resort to searching the entire system in order to locate a match. Once the search is completed and the information indexed by the respective agents, further searches using the same search term is lowered dramatically.

A test-bed is setup and the performance of the system is analyzed to determine performance parameters of the Global Search Space (GSS) under real life environments. The GSS consists of three nodes NODE1, NODE2 and NODE3. Each node is made up of three entities, the SearchAgent (SA), the SearchGateway (SG) and the IntelligenceAgent (IA). They are represented node-wise respectively as follows-

On NODE1 - SA1, SG1 and IA1, On NODE2 – SA2, SG2 and IA2 and On NODE3 – SA3, SG3 and IA3

The First Positive Response Time is determined at the Initializer and is calculated for each node on the GSS that has an executor. For a given GSS with "N" nodes, there will be a maximum of "N" FPRTs as well for each search. The FPRT thus represents the time taken in seconds for the first positive response from an executor node reaches the Initializer node and is calculated at the Initializer end of the GSS.

The Testing procedure is as follows, the Test Battery consists of five search terms S1, S2, S3, S4 and S5. Three searches "First-Search", "Second-Search" and "Third-Search" are conducted for each search term, one each of the searches are initialized from each of the different nodes on the GSS.

S1 = "*.gif"

S2 = "*dll"

S3 = "*exe*"

S4 = "*.tmp"

S5= "*ico"

NODE1 – SYSTEM32 with ip 10.0.0.32, LSS size = 28411 files

NODE2 – SYSTEM30 with ip 10.0.0.30, LSS size = 42173 files

NODE3 – SYSTEM27 with ip 10.0.0.27, LSS size = 15149 files

First-Search is initialized from SG1 on NODE1

Second-Search is initialized from SG2 on NODE2

Third-Search is initialized from SG3 on NODE3

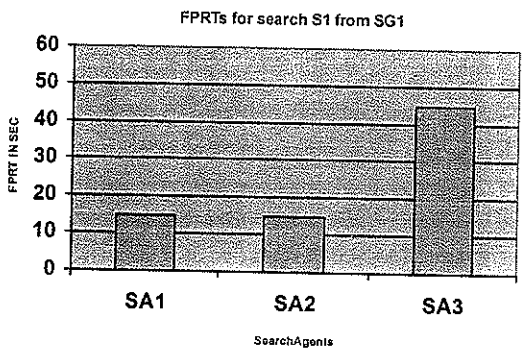

Figure 2: FPRTs for search S1 from SG1

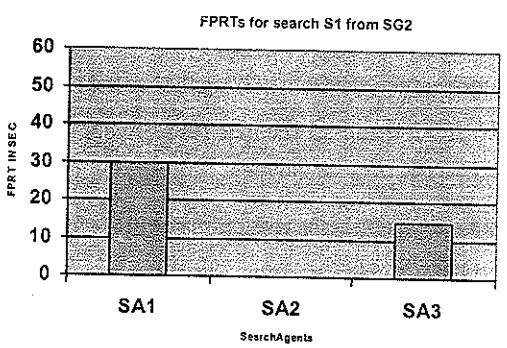

Figure 3: FPRTs for search S1 from SG2

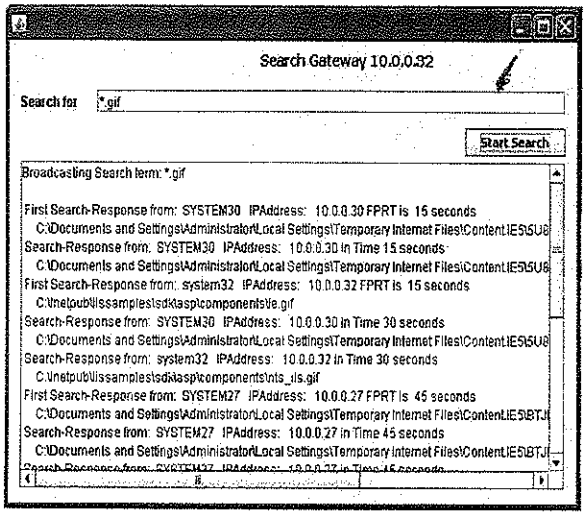

Figure4: First-Search results on SG1 for termS1

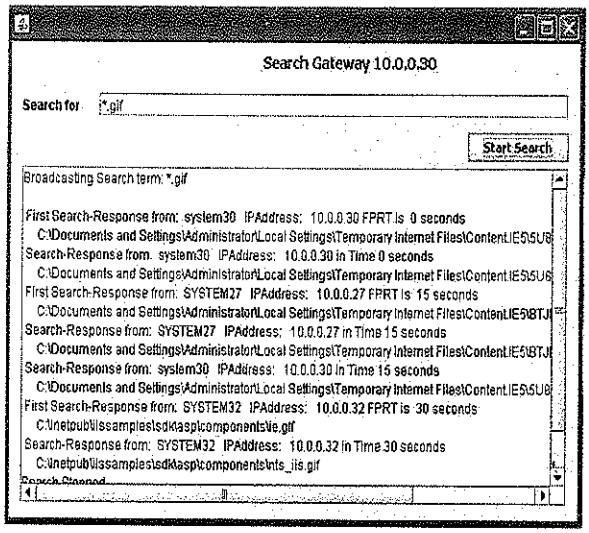

Figure5: First-Search results on SG2 for termS1

For a given search term S1 input from the SearchGateway SG1, the FPRTs calculated from the Initializer SG1 for each of the three SearchAgent SA1, SA2 and SA3 is 15, 15 and 45 respectively in seconds. The test is repeated with the same search term from SG2 on the second node yields FPRTs of 30, 0 and 15 seconds respectively from SA1, SA2 and SA3 respectively. These results clearly indicates to two factors, one is the reduction in FPRTs for NODE2 and NODE3, which translates to faster results for the above search term on all searches on the GSS. The second factor is the inherent dynamism of the GSS, which has caused a rise in FPRT for Node1. It is further clear that there is an overall significant improvement in the searching speed on the GSS indicated by the drop in average FPRT for the search from 25 seconds to 15 seconds, an improvement of nearly 40%.

9. Conclusion

The dynamic nature of the networked environment demands that any solution for locating network resources such as files and folders be able to inherently deal with the unpredictable nature of its operating environment. The peer-to-peer agent based system suggests a remedy in the form of independent agents located at each node with complete autonomy of action.

This research work clearly proves that unlike the commonly used solutions for achieving improved search throughputs such as indexing and caching, this solution achieves even higher performance with much lower communication and processing overheads. The solution can demonstrate that by using the suitable paradigm to solve the problem, in this context, Intelligent Agents, a dramatic increase in solution efficiency can be achieved. A learning based artificial intelligence (AI) is used to increase the efficiency of the solution. The AI analyses the performance of the attached search agent and derives ways to make future searches more efficient.

References

[1] Andrew Garland and Richard Alterman, *"Autonomous Agents and Multi-Agent Systems"*, Actapress, Vol. 8, PP. 267-301, 2004.

[2] Andrej Lucny, *"Computing and Informatics"*, Vol. 26, 2004

[3] Caroline C. Hayes,Agents in a Nutshlell , *"A Very Brief Introduction IEEE Transactions on knowledge* and data engineering", Vol. 11, No. 1, January/February, 1999.

[4] Erika Horn, Mario Kupries and Thomas Reinke , *"Properties and Models of Software Agents and Prefabrication for Agent Application Systems"*, Proceedings of the 32nd Hawaii International Conference on System Sciences – 1999.

[5] Georgakarakou, C. E. and Economides, *"A Software Agent Technology: An Overview, Agent and Web Service Technologies in Virtual Enterprises"*, N. Protogeros (ed.), 2006.

[6] Henry Lieberman, *"Autonomous Interface Agents, Massachusetts Institute of Technology," 1999.*

[7] Hyacinth S. Nwana, *"oftware Agents: An Overview"*, Knowledge Engineering Review, 1996.

[8] Hyacinth S. Nwana and Divine T. Ndumu, *"Perspective on Software Agents Research"*, Research Report No. 97-03, Commerce Net, 1997.

[9] James Odell, *"gents: Technology and Usage, (Part 1) Distributed computing architecture/e-business advisory service Executive Report"*, Vol. 3, No. 4, 2000.

[10] Jeffrey M. Bradshaw, *"In Introduction to Software Agents"*, An introduction to agent technology PP 44, Chapter 1,1998.

[11] N. R. Jennings and M. Wooldridge, *"Applications of Intelligent Agents"*, IEEE Transactions on knowledge and data engineering, 1997.

[12] Nicholas R Jennings and Michael J Wooldridge, *"Agent Technology Foundations applications and markets"*, Springer, 1998.

[13] Michael Luck and Peter McBurney , *"Challenges for Agent Technology Moving towards 2010"*, UPGRADE Vol. V, No. 4, August 2004.

[14] Michael Luck, Peter McBurney, *"Onn Shehory and Steven Willmott"*, AgentLink III, September 2005.

[15] Marina Roesler and Donald T. Hawkins, *"Intelligent agents"*, Online, Vol. 18, No. 4, PP. 18-32, 1994.

[16] Pattie Maes, *"Agents that reduce work and information overload"*, Communications of the ACM, Vol. 37, No. 7, PP. 30-40, 1994.

[17] Ralf Steinmetz, Klaus Wehrle (Eds), *"Peer-to-Peer Systems and Applications"*, Lecture Notes in Computer Science, Vol. 3485, September 2005.

[18] *"Foundation of Peer-to-Peer Computing"*, Special Issue, Elsevier Journal of Computer Communication, (Ed) Javed I. Khan and Adam Wierzbicki, Vol. 31, Issue . 2, February 2008.

[19] Stephanos Androutsellis-Theotokis & Diomidis Spinellis, *"A survey of peer-to-peer content distribution technologies"*, ACM Computing Surveys, 36(4):335–371, December 2004.

[20] Detlef Schoder and Kai Fischbach, *"Core Concepts in Peer-to-Peer (P2P) Networking"*, In: Subramanian, R.; Goodman, B. (eds.): P2P Computing: The Evolution of a Disruptive Technology, Idea Group Inc, Hershey, 2005.

Author's Biography

Ms T Amudha received her B.Sc Degree in Physics, Masters Degree (MCA) in Computer Applications and M.Phil in Computer Science in 1995, 1999, and 2003 respectively, from Bharathidasan University, India. She has qualified UGC-NET for Lectureship in 2003. She has 10 years of academic experience and is currently serving as Lecturer, School of Computer Science and Engineering, Bharathiar University; India. She is currently pursuing her doctoral research at Bharathiar University in the area of Agent Based Computing. She is engaged in active research guiding MPhil research scholars and has successfully produced 8 MPhil so far. Her research interests include Object Technologies, Distributed Systems and Agent

Based Computing. She has for his credit more than 15 publications in International/National Conferences/ Journals. She is member of Computer Society of India[CSI], International Association of Engineers[IAENG], International Association of Computer Science and Information Technology[IACSIT].

Mr Manu Rajan Nair received the B.Sc Degree and M.Sc Degree in Computer Science from University of Kerala, India. He pursued his MPhil in Computer Science under the guidance of Ms T Amudha, during 2007-2008 in the School of Computer Science and Engineering, Bharathiar University, India. His research interests include agent technologies and Computer networks.