An Agent Based Peer-to-Peer Network with Thesaurus Based Searching, and Load Balancing

Sabu M. Thampi
Department of CSE
L.B.S College of Engineering
Kasaragod-671542
Kerala, India
smtlbs@yahoo.co.in

K. Chandrasekaran
Department of CSE
National Institute of Technology Karnataka
Surathkal-575025
Dakshina Kannada, Karnataka, India
kch@nitk.ac.in

Abstract

This paper describes a search mechanism for files in an unstructured peer-to-peer network. Most of the existing P2P architectures cannot autonomously locate services on the P2P network. Peers can hardly work and cooperate as a team. Using well-designed agents can improve the efficiency of the operations and data communication in the P2P applications. In the proposed system, agents are residing on peers and almost take care of every thing. They receive a problem from the user or other agents, send each job to the responsible agents and merge the sub-solution gathered from them to present a final solution. The communication among nodes takes place through mobile agents. The key features include content matching, parallel downloads, agent based load balancing and thesaurus based searching.

1. Introduction

A P2P system is a decentralized and distributed network of nodes that is capable of sharing and distributing resources between themselves. P2P offers exciting new potential in distributed information processing. The prime objective of a node in a P2P network is to search and acquire resources and services available on other nodes in the network and, simultaneously allow other nodes to access resources available on the node itself. P2P Internet applications have recently been popularized through file sharing applications like Napster, Gnutella, FreeNet, KaZaA and others. Within these applications, the P2P concept is mainly used to share files, i.e. the exchange of diverse media data, like music, movies and programs. The growth in the usage of these applications is enormous and even more rapid than the growth of the World Wide Web.

A software agent is a computing entity that carries out some task or tasks on behalf of someone or something. A stationary agent executes only on the system where it begins execution. A mobile agent is an active object that can move both data and functionality (code) to multiple places within a distributed system. There are several good reasons for using mobile agents [2]. Mobile agents:

- reduce network load
- overcome network latency
- encapsulate protocols
- execute asynchronously & autonomously
- adapt dynamically

There are many different implementations of mobile agents in existence [3]. Some of the popular implementations include Telescript, Concordia, Mole, Voyager and Aglets. The Aglets project [5] is a Java based implementation that was originally developed by IBM in Japan. An aglet can be dispatched to any remote host that supports the Java Virtual Machine. This requires from the remote host to pre-install Tahiti, a tiny aglet server program implemented in Java and provided by the Aglet Framework. To allow aglets to be fired from within applets, the IBM Aglet team provided the so-called "FijiApplet", an abstract applet class that is part of a Java package called "Fiji Kit". FijiApplet maintains some kind of an aglet context. From within this context, aglets can be created, dispatched from and retracted back to the FijiApplet.

Unlike conventional client-server file transfer systems, all nodes in P2P networks are both clients and servers, simultaneously. Once a node connects to any point in the network, it can exchange files with other nodes in the network. Most of the existing P2P architectures cannot autonomously locate services on the P2P network. Peers can scarcely work and cooperate as a team.

Using well-designed agents can perk up the efficiency of operations and data communication in

Proceedings of the 2005 International Conference on Computational Intelligence for Modelling, Control and Automation, and International Confe
Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'05)
0-7695-2504-0/05 $20.00 © 2005 IEEE

P2P applications. In the proposed system, agents are residing on peers and almost take care of every thing. They receive a problem from the user or other agents invoke necessary resources after consulting with other agents (peers). They also send each job to the responsible agents and merge the sub-solution gathered from them to offer a final solution. The communication among nodes takes place through mobile agents.

This paper is organized as follows: Section 2 gives an overview of different types of agents used in the proposed system. Key features are described in section 3. Section 4 discusses the model of the system. P2P algorithm is explained in section 5. Results and discussion is in section 6. Section 7 concludes the paper.

2. Agent architecture

The mobile agent based system is developed using Aglet Software Development Kit. The system makes use of several types of agents. Cooperation of the agents enables the system to perform the required tasks efficiently. This section gives a brief introduction to various agents employed in the system (Figure 1).

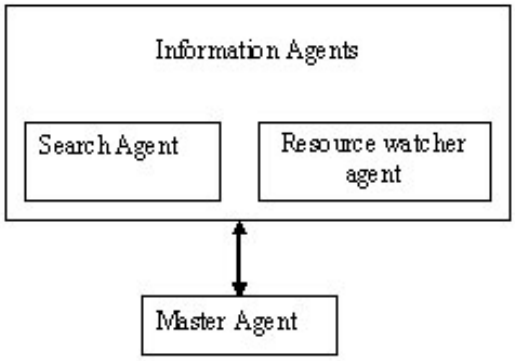

Figure 1. Agents in a node

Master Agent manages and controls the collection of other agents and provides the basic user interface of the application. It is designed to send essential information to other agents. Agents that are dispatched from other nodes submit their queries to master agents to catch the results. There are two types of information agents: *search agent* and *Resource watcher Agent.*

Each peer participating in the network may utilize its resources available to other peers. When a user needs a particular resource such as a file, *search agent* discovers a match for the needed resource in the host. *Resource watcher agent* acquires the specifications of an available resource and forwards the same to the master agent. The watcher agent decides the duration for using the resource and the time of starting and ending of the service. Watcher agent stores and retrieves information to and from the local database. It also notifies status of incomplete downloads to the user with the assistance of master agent.

In addition to stationary agents, the system uses java based mobile agents to provide communication and data transfer among nodes. Download agent is a special type of mobile agent to assist downloading different sections of a file.

3. Key features

The proposed agent based P2P system has four key features, which distinguish it from typical P2P systems: content matching, parallel downloads, load balancing and thesaurus based searching. In this section, we discuss the design of and algorithms used to implement these features.

3.1 Content matching

In the proposed system, files are identified by content; we use the SHA1 hashing algorithm to generate a 160-bit "fingerprint" for each file. Hashes are precomputed for all the files a user is willing to share. These hash values are stored in a database. Users are then able to search for files they want in two ways - either by the file hash or by the string query search. For every file that is hashed, the file path/hash pair is added to a database. On the other hand, when a user decides that they no longer want to share particular files, the associated database entries are deleted.

The file watcher agent verifies any directories, which are shared, for newly added or removed files and appropriately updates the database.

3.2 Load balancing

Load balancing enhances the performance of P2P systems. Incoming requests should be evenly distributed among nodes to achieve quick response. Traditional load balancing approaches are implemented based on message passing paradigm. Mobile agents provide a novel technology for implementing load balancing mechanism in P2P networks. The mobile agent can embed policies of load balancing, travel to other nodes, and interact with them on the site to acquire latest load information. The mobile-agent based approach can minimize the network traffic and enhance the flexibility of a load balancing mechanism.

The proposed load balancing scheme (Figure 2) makes use of two types of agents – Node Management Agent (NMA) and Load Information Agent (LIA). NMA is a stationary agent that sits at a P2P node, responsible for monitoring the workload. When a node

Proceedings of the 2005 International Conference on Computational Intelligence for Modelling, Control and Automation, and International Confe
Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'05)
0-7695-2504-0/05 $20.00 © 2005 IEEE

is overloaded, the NMA on it begins the reallocation process. NMA dispatches the jobs to other neighboring nodes whose load is below some threshold. LIA is a mobile agent accountable for information gathering. It travels around the neighboring nodes and gathers the load information. Meanwhile it broadcasts the load information to the neighbors.

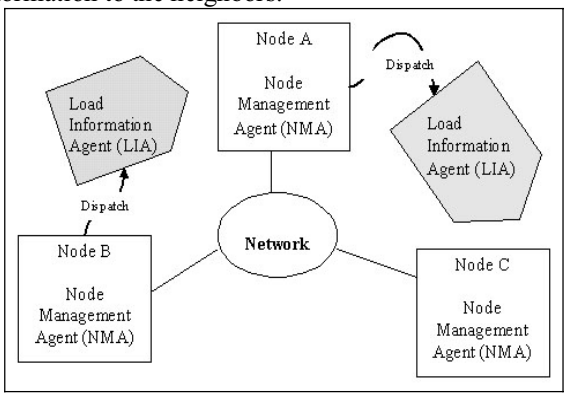

Figure 2. Agent based load balancing

The NMA activates the LIA to accumulate the load information from neighbors while the load exceeds a threshold. The LIA navigates through neighbors to gather the updated values of cpu_load, no_connect and free memory on each node and calculate the load metrics. The load on a peer is computed as:

$$Load = w1 * cpu\_load + w2 * no\_connect / MAXON + w3 * free\_mem$$

Where cpu_load is the workload on the peer, measured in the length of job queue, no_connect is the number of requests being processed on the node.

The load balancing scheme employs *find-best strategy* for node selection. The LIA visits every neighboring node and selects the nodes that have a load below a threshold.

3.3 Parallel download

Parallel-Download is a technique used to fetch files from multiple sources. Instead of selecting a particular node to download a file, a number of P2P nodes are selected and content is downloaded in parallel, getting different bits of the content from different nodes. Clients experience a transfer rate equal to the sum of the individual transfer rates of the peers contacted. Figure 3 shows an example of parallel download session. There are three nodes that can serve the file and the client requests one-third from each of them.

There are many different algorithms to perform parallel download. The algorithm that we used in this paper is a variation of the ones proposed in [15]. It is a dynamic algorithm based on load balancing. This scheme tries to request more bytes from peers by monitoring the load on each of the selected nodes. Master agent dispatches download agents to the least loaded peers for downloading the required file. The file is divided into many small portions. Each peer is assigned one of them. When one node finishes serving a piece, it is assigned a new one. This process continues until all sections of file have been downloaded.

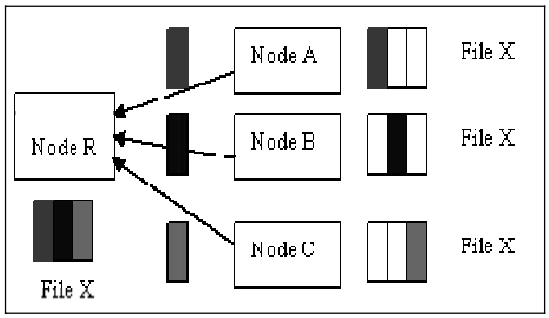

Figure 3. Example of a parallel download session

3.4 Managing interrupted transfers

The agent submits request for a section of a file, which may be the entire file from the sender. While downloading a file, the hash value of the original file needs to be recorded, so that it can be used to locate the file again later. This hash value will differ from that of the local copy of the file before the download has completed. Hence we need to differentiate files which are the result of complete downloads, and those which are incomplete. Table 1 shows an example of the information stored in the database for a complete file and an incomplete file.

Table 1. Complete and incomplete download

File Path	Hash	File Info.
download/file1	412672bbe831 45610cf66f879 3858a03fbf29a9b	CMP 10485760
incomplete/file2	7e264558f0eb6 ecdc7e2357b80a 5383af367e92e	5431340-452-554-2400

Hash values are stored as null-terminated strings. Complete files have the CMP label followed by the size of the file stored in the "file info" field. Incomplete files have the size of the target file, followed by details about sections of the file that have been successfully downloaded. The file info field is used when recalculating hash values. Whenever a user wants to recommence the interrupted download of a file (*watcher agent informs the details of the downloaded file to the user*), they can search for an

identical file to resume their transfer. This is made by searching for files with the same hash values as that of the original file they are trying to obtain.

3.5 Thesaurus for searching files

By guiding indexers and searchers about which terms to use, it can help to improve the quality of retrieval. By listing groups of words that have similar meanings to each other, a thesaurus offers a choice of alternative words that can be used in place of one that you already have in mind. A file name is a natural language description of the file, so when a user searches for a file, he would presumably wish to search by file name.

People tend to group similar files in directories, which have names, that can potentially provide more information about the files with in. For example, consider a user sharing a file with the name "classic music/das/fourpeople.mp3". If an incoming search request is for "music", the different equivalent directory names are searched in the thesaurus. The system returns back details of files in the matching directories. Look at how many words there are for the word *music*, for instance: *classical, folk, harmony, hymn, jazz, melody, tune, ragtime, rap, refrain, rock, song, soul, swing, tune, art music, chamber music, longhair music, operatic music, semi classical music, concert music, symphonic music.* Names of directories and subdirectories that bear any of these words will be selected and the user downloads files from these directories.

4. System model

The model of the proposed system is shown in figure 4 [4]. While a user submits a query, search agent searches its own resources. If required file is found, the watcher agent retrieves the corresponding information. If a resource is not found on the searched nodes, the peer dispatches mobile agents to its neighbors whose load is below some threshold. The neighboring agents search their nodes and in turn transmit the query to their neighbors by dispatching mobile agents, and so on. A journey time also known as time-to-live is set for every process and the search ends when the journey time expires.

Each mobile agent is assigned a user request-id while it is dispatched. In addition, they are provided information on nodes visited previously. Hence, incidence of revisits is reduced. If two mobile agents from the same creator peer arrive at the current peer, the request-id of the second agent is verified and if it is same as the first one, the second agent destroys itself.

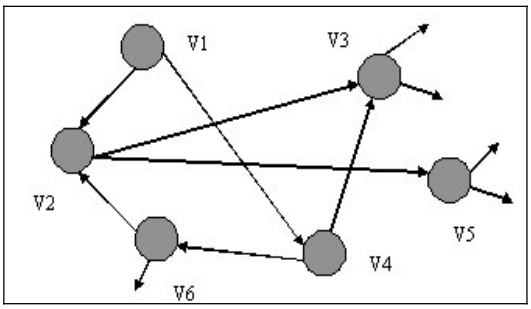

Figure 4. System model

5. P2P algorithm

The algorithm we used is as follows:

1. The user submits a query.
2. Master agent receives the query and forwards it to the search agent.
3. Search agent checks the query to check whether it is a file name with an extension, or a string
   If it is a filename, search for the file.
   else
   Perform thesaurus-based search.
4. Search agent submits the result back to master agent.
5. Resource watcher agent retrieves the required information
6. Node dispatches mobile agents to its least loaded neighbors for searching file(s).
7. Set user request-id and TTL for each mobile agent.
8. Neighboring agents propagate the query to their neighbors.
9. Mobile agents present the result back to the creator peer.
10. The agent decrements its journey time (TTL) as soon as it arrives on a peer.
11. In order to reduce incidence of revisits, mobile agents are provided with information on already visited nodes.
12. If two mobile agents from the same creator peer arrive at the current peer and the user-request-id of both agents are same, the second mobile agent destroys itself.
13. Clones of mobile agents are created to visit the nodes until the expiry of journey time (TTL).
14. Master agent gathers information on node last visited by each mobile agent.
15. In case the search fails during journey time, master agent dispatches a mobile agent to a node selected randomly from the list prepared in step (14). The selection is not based on load at the nodes.
16. Repeat steps 7 to 15 to complete the search.

Proceedings of the 2005 International Conference on Computational Intelligence for Modelling, Control and Automation, and International Confe
Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'05)
0-7695-2504-0/05 $20.00 © 2005 IEEE

17. The creator peer merges the subsolution gathered from various nodes to offer a final solution.
18. Master agent displays the search results.
19. Master agent dispatches the download agents to download sections of a file (download from multiple sites is supported).
20. Watcher agent notifies the user about incomplete downloads. Procedure described in section 3.4 (managing interrupted transfers) is followed to download the remaining portions of the file.
21. When a node is overloaded, NMA dispatches the jobs to its neighbors based on load information.

6. Results and discussion

An agent based P2P system is implemented and tested to evaluate the performance of the system. For performance evaluation and peer-to-peer comparison of the system, three popular distributed applications - Gnutella, HTTP server and FTP server along with Agent Based P2P system are tested in a common environment and circumstances.

The performance evaluation compares the time taken for file transfer, file browsing and file searching among four applications. The File browsing time for both Gnutella and HTTP server are not available for performance comparison. Only the search time for Gnutella and Agent based system is available. Figures 5, 6 and 7 compare the performance of the applications in terms of file transfer, file searching and file browsing. Despite the use of both stationary agents, mobile agents and other key features, which exist in the P2P system, the performance is still competitive with others.

For comparing download performance of the dynamic algorithm, we plot all the measured waiting times for different number of nodes used. As we increase the number of nodes that contain the required file (file size 7.2 MB), we get better performance. Figure 8 shows our dynamic algorithm compares to an algorithm with no parallel download. It is observed from the graph that the traditional method of download from single site performs much worse that the dynamic method based on load.

The performance of agent based load balancing scheme is evaluated by comparing its performance with the message-passing paradigm. The performance is assessed in terms of network traffic. Figure 9 compares the network traffic of two schemes.

The system throughput is measured in the number of requests processed per second. Figure 10 shows the system throughput of message passing paradigm and agent based load balancing. The agent based scheme can outperform the message passing scheme when the number of clients is above 250.

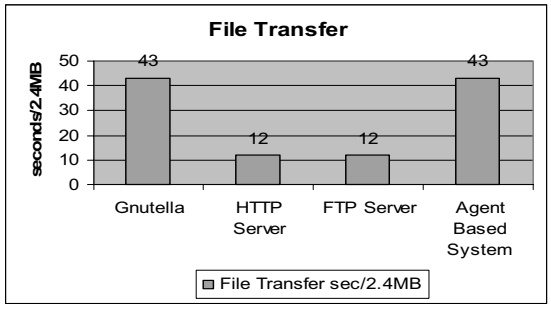

Figure 5. File transfer performance

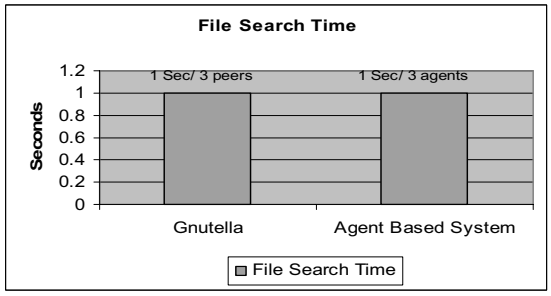

Figure 6. File search in group

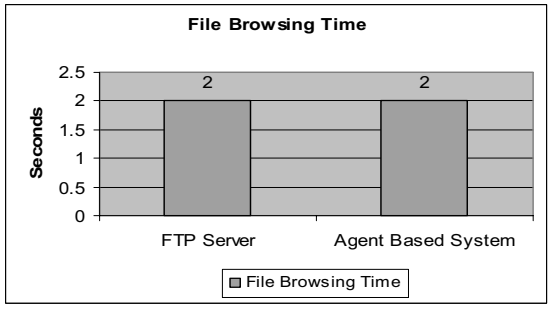

Figure 7. File browsing performance

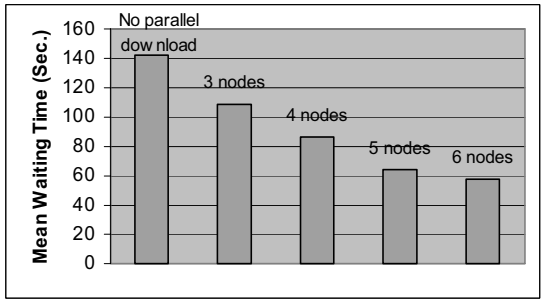

Figure 8. Comparison of performance of download algorithms

Proceedings of the 2005 International Conference on Computational Intelligence for Modelling, Control and Automation, and International Confe
Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'05)
0-7695-2504-0/05 $20.00 © 2005 IEEE

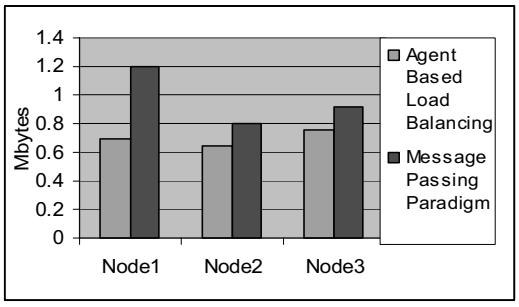

Figure 9. Network traffic

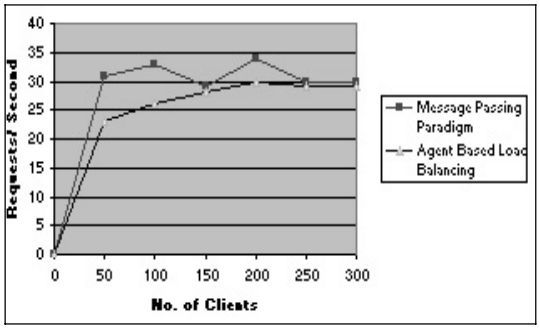

Figure 10. System throughput

7. Conclusion

A P2P system based on load balancing, thesaurus based searching, parallel download and content matching is presented. The agent-based load balancing approach outperforms the traditional message passing method. Dynamic parallel download technique based on load can adapt better to changing network conditions. Since we have not provided any sort of caching or replication scheme, any files the node store are inaccessible while they are heavily loaded. The system can be extended to incorporate some form of probabilistic selection criterion with a bias against heavily loaded peers, to prevent sudden shifts where every client avoids sending requests to heavily loaded peers.

8. References

[1] Belmon, S. G., and Yee, B. S, "Mobile agents and intellectual property protection," In Rothermel and Hohl, pp. 172–182.

[2] R. Gray, D. Kotz, G. Cybenko and D. Rus. Mobile Agents: Motivations and State-of-the-art systems. Dept. of Computer Science, Dartmouth College, 2000.

[3] Dan Connolly, Mobile Code Systems, *http://www.w3.org/Mobile Code/*

[4] Leontiadis, Vassilios, Evaggelia Pitoura, Cache updates in a Peer-to-Peer network of Mobile Agents, *http://femto.org/p2p2004/papers/leontiadis.pdf.*

[5] IBM Japan Aglets, *http://www.trl.ibm.com/aglets/index.html*

[6] RothermeL, K., and Popescu-Zeletin, R, Eds. Mobile Agents (MA '97), vol. 1219 of Lecture Notes in Computer Science. Springer Verlag, Berlin Heidelberg, 1997.

[7] Qin Lr, Pei Cao, Edith Cohen, Kai Li, Scott Shenker, Search and Replication in Unstructured Peer-to-Peer Networks, *proc. of the 2002 ACM SIGMETRICS international Conference on Measurement and Modeling of Computer",* 2002, pp258-259

[8] Antenella Di Stefano, "Locating Mobile Agents in a Wide Distributed Environment", IEEE *Trns. On Parallel and Distributed Systems*," VOL.13 pp.844-863, Aug. 2002.

[9] D. Lange, M.Oshima, Programming and Deploying Java Mobile Agents with Aglets, Addison-Wesley Longman, Reading, Mass., 1998.

[10] A.Datta, M.Hauswirth, and K.Aberer, "Updates in highly unreliable, relicated peer-to-peer systems", in *proc. of ICDS2003, 23rd International Conference on Distributed Computing Systems,* Providence, Rhode Island, May 2003, pp. 76-85.

[11] V.Pham, A: Karmouch, Mobile Sofware Agents: An Overview, *IEEE Communications Magazine*, 36(1998), pp.26-37.

[12] V.V. Dimakopoulos, E. Pitoura, "A Peer-to-Peer Approach to Resource Discovery in Multi-Agent Systems," *in proc. CIA 2003, 8TH Int'l Workshop on Cooperative Information Agents*, Helsinki, Finland, Aug. 2003, pp 62-77.

[13] D.Lv, P. Cao, E.Cohen, K Li, and S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks," *in proceeding of the 16th ACM International Conference on Super computing*, 2002.

[14] Gnutella Protocol Specification, Version 0.4, in *http://www.limewire.com/developer.*

[15] Pablo Rodriguez, Andreas Kirpal, and Ernst W. Biersack, "Parallel-Access for Mirror Sites in the Internet", *Infocom, 2000.*

[16] John W. Byers, Michael Luby, Michael Mitzenmacher, and Ashutosh Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data", *ACM SigComm, 1998.*

0-7695-2504-0/05 $20.00 © 2005 IEEE