

An Agent Enriched Distributed Data Mining On Heterogeneous Networks

Gurpreet S. Bhamra¹, R. B. Patel², A.K.Verma³,

¹M.M. Institute of Computer Tech. and Business Management, MMU, Mullana-133203, Haryana, India

²Deptt. of Computer Engineering, M. M. Engineering College, MMU, Mullana-133203, Haryana, India

³Deptt. of Computer Engineering, TIET, Thapar University, Patiala-147004, Punjab, India
{preet_bhamra@rediffmail.com, patel_r_b@indiatimes.com, akverma@tiet.ac.in}

Abstract- Advances in computing and communication over wired and wireless networks have resulted in many pervasive distributed computing environments. Many of these environments have different distributed sources of voluminous data and multiple computing nodes. Analyzing and monitoring these distributed data sources require data mining technology designed for distributed applications. Distributed Data Mining (DDM) considers the Data Mining in a broader context where it pays careful attention to the distributed resources of data, computing and communication in order to consume them in a near optimal fashion.

Distributed applications often download large amount of server resident information over a network, process it, and generate comparatively small amount of result data. If these applications are instead written using mobile agents (MAs), the agents can execute on server and access server data without using the network. Only the result data is carried back over the network for presentation to the user. Thus, the bandwidth requirements of the application can be saved.

In this paper, we present a Mobile Agent based Distributed Data Mining System (MADDM) to extract some trends or patterns representing the knowledge from the Distributed Pattern Databases on Heterogeneous Network. The focus is on creating basic mechanisms and leaving policy choices to application developers. With intelligent pattern based search, the developed agent system supports scalability and bandwidth utilization and avoids indexing and query flooding problems of most existing systems based on meta-data search. Platform for Mobile Agent Distribution and Execution (PMADE) is used for the implementation. The main reason behind the use of PMADE is that it supports large scale multi-agent system & simplifies the application development and deployment by freeing the application developer of all low-level details including communication, security and scheduling.

Keywords - Mobile Agents (MAs), Distributed Data Mining (DDM), Pattern, MADDM, PMADE.

I. INTRODUCTION

A Mobile Agent (MA)[1, 6] is an autonomous program that can migrate under its own or host control from one node to another in a heterogeneous network. In other words, program running at a host can suspend its execution at an arbitrary point, transfer itself to another host (or request the host to transfer it to its next destination) and resume execution from point of suspension. Once it is launched by a user, can travel from node to node autonomously, and can continue to function even if the user is disconnected from the network. It performs its job wherever and whenever it is found appropriate and is not restricted to be co-located with its client. Thus, there is an inherent sense of autonomy in the mobility and execution of

an agent. Agents can be seen as automated errand boys who work for users. MA research evolved over the past years from the creation of many different monolithic mobile agent systems (MASs), often with similar characteristics and built by research groups spread all over the world, for optimization and better understanding of specific agent issues [1,2].

In the area of networked environments, MAs can be seen as a new paradigm for the implementation of fully distributed software systems with a balanced peer-to-peer concept [3]. Especially interesting is the case where the network provides a dynamic environment [5], e.g., if mobile network nodes and services appear and disappear and where agents act as intelligent entities by determining their own path at runtime dynamically in the continuously changing landscape. The migration of MAs is associated with different movement costs, viz., transmission time, round trip time, number of hops, etc. Costs are also generated by executing the agent's algorithms on the agent host (AH) [4]. The focus of this research is to optimize the autonomous navigation through a network in general. We do not care for the agent's autonomy on the user task level, e.g., negotiations and fulfill user tasks, etc. We also do not optimize the technical (hardware) infrastructure of the underlying network. The movement of MAs is based on a logical network view which is based on the AH equipped nodes on the network.

Many systems have been developed to aid users as they work, by performing automatic Web search for information to support tasks such as Web browsing, query generation, and document authoring, by mining the Web and other resources. Reflecting context has been recognized as important to realizing the potential of Web search in general, and context-sensitivity plays an especially crucial role in proactive retrieval systems. The extent to which the system can provide context-relevant information determines whether the system will be an aid or an annoyance. Unfortunately, fully exploiting contextual information during Web search is challenging. In current search engines, there are strong limits on query length (e.g., Google's query length limit of ten terms), making it difficult to provide enough terms to describe rich contexts. Even if an adequate context description can be included within the limits, there is no guarantee that the vocabulary used to describe the context will match the vocabulary by which the resource is indexed.

Context searching is when an end-user can select which tags (patterns) or elements (patterns) within a document to search on within one or more documents. Element indexes documents according to individual in a document, rather than

just to the document level as most search engines do. Element provides an intuitive interface that allows users to easily select what contexts they wish to search, without having to know the structure of the document nor complex Path syntax. Element technology can also highlight search terms and link directly to the point in the document in which the "hit" is located, when searching and displaying document. This allows common users to find the information they need, when they need it, within just a few clicks.

This paper presents a Mobile Agent based Distributed Data Mining System (MADDM) to extract some trends or patterns representing the knowledge from the Distributed Pattern Databases on a Heterogeneous Network. Most of the existing DM techniques were originally developed for centralized data and need to be modified for handling the distributed case. The increasing demand to scale up to massive data sets inherently distributed over a network with limited bandwidth and computational resources available motivated the development of methods for parallel(PKD) and distributed knowledge discovery(DKD) [25]. The related pattern extraction problem in DKD is known as Distributed data mining (DDM). DDM is expected to perform partial analysis of data at individual sites and then to send the outcome as partial result to other sites where it is sometimes required to be aggregated to the global result [14].

MA improves the search performance, i.e., means to optimize a MA's path through a network of AH (nodes). Thereby, an agent visits only those AHs which provide a service of interest. Furthermore, the agent uses a fast path through a network based on known infrastructure characteristics (as QoS). Finally, an agent optimizes its transmissions between AHs with the help of several migration strategies described in [6]. The focus is on creating basic mechanisms and leaving policy choices to application developers. With intelligent context based search, the developed agent system supports scalability and bandwidth utilization and avoids indexing and query flooding problems of most existing systems based on meta-data search. Platform for Mobile Agent Distribution and Execution (PMADE) [4] is used for the implementation. All components have been prototyped and are currently evaluated. We look at their interaction, trace typical scenarios and discuss their efficiency in a number of situations. Based on the results of our experiments, we will discuss the applicability of the proposed framework and pinpoint relevant advantages as well as inherent constraints.

Rest of the paper is organized as follows. Section 2 gives brief overview of PMADE. Section 3 presents system architecture. Implementation of system is shown in Section 4. Section 5 explores on related works finally article is concluded in Section 6.

II. OVERVIEW OF PMADE

Figure 1 shows the basic block diagram of PMADE (Platform for Mobile Agent Distribution and Execution). Each node of the network has an Agent Host (AH), which is responsible for accepting and executing incoming autonomous Java agents and an Agent Submitter (AS) [4, 6], which submits the MA on behalf of the user to the AH. A user, who wants to perform a

task, submits the MA designed to perform that task, to the AS on the user system. The AS then tries to establish a connection with the specified AH, where the user already holds an account. If the connection is established, the AS submits the MA to it and then goes offline. The AH examines the nature of the received agent and executes it. The execution of the agent depends on its nature and state. The agent can be transferred from one AH to another whenever required. On completion of execution, the agent submits its results to the AH, which in turn stores the results until the remote AS retrieves them for the user.

The AH is the key component of PMADE. It consists of the manager modules and the Host Driver. The Host Driver lies at the base of the PMADE architecture and the manager modules reside above it. It is the basic utility module responsible for driving the AH by ensuring proper co-ordination between various managers and making them work in tandem. Details of the managers and their functions are provided in [6]. PMADE provides weak mobility to its agents and allows one-hop, two-hop and multi-hop agents [6]. PMADE has focused on Flexibility, Persistence, Security [13], Collaboration and Reliability [4].

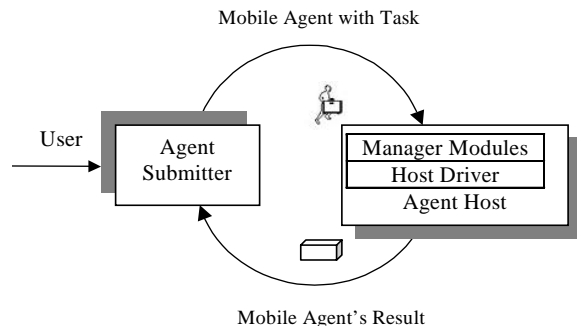


Figure 1. Block architecture of PMADE

III. SYSTEM ARCHITECTURE

Context searching with pattern allows the end-user to pre-select which tags to search in *before* pressing the search button. This will return only relevant and less results. In many cases, much less results because they are more relevant, we will find the information we need much faster because we should not have to shift through thousands of results. For providing reliable solution we have developed a multi-agent system using MAs as shown in Figure 2. This system is divided into four sections- interface, agents, policy and Agent-Agent Communication Layers. The interface is used for the communication with external world via PMADE as well as for agent-agent communication. It maintains a buffer. The buffer is used for storing the messages temporarily whenever communication delay arises. It also helps to provide fault tolerance to the message on system failure. For providing the security to messages it uses PMADE security [6]. We have considered few assumptions and identified some policies & agents in the development of the system. The details about

policy, agents, and agent-agent communication are discussed next.

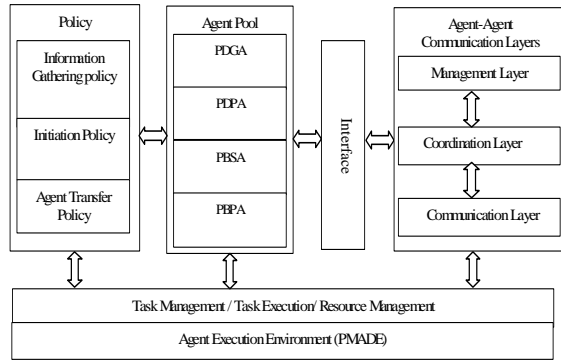


Figure 2. Mobile Agent based Distributed Data Mining System (MADDM)

A. ASSUMPTIONS

We have considered the following assumptions:

- Servers taken into an account are heterogeneous in nature. They can have different hardware configuration, operating system and processing power.
- Each sever is capable of processing the client’s agents and cooperate with each other in order to share the workload when frequency of arriving agents are very high.
- To provide dynamic capability to the server capacity, can be changed due to the variation of agents density.
- A MA can be proprietary to a server where it is created and perform dedicated operations for the owner.
- A MA can be shared among a group of servers to act on behalf of these servers. MAs can interact with each other by direct data exchange.
- A MA can interact using the stigmergy technique in which the MAs can collect the information from the traces left in the environment by one another. A MA can gather the information placed on a server by other MAs who have previously visited there. The applications based on this principle are robust and adaptive. The stigmergy is an indirect method for the interaction between MAs, which can reduce the network traffic and achieve quick decision-making.

B. POLICY

We have defined four policies according to the need of agents we have founded. These policies are governed by system administrator and updated according to the need of searching schemes.

Information Gathering Policy (IGP) specifies the strategy for the searching/collection of information including the frequency and method of information gathering. The frequency is determined based on a tradeoff between the accuracy of load information and the overhead of information collection. This policy also assures (trust value) the accuracy of information gathered.

Initiation Policy (PI) determines who starts the searching process (MA). The process (MA) can be initiated by a client and it is monitored by intermediate servers. PMADE activated servers are being in-charge to process coming agent on a server as they are arriving.

Server Selection Policy (SSP) selects an appropriate server based on the load information, i.e., density of MAs to which the workload on an overloaded server can be reallocated. Different strategies can be applied to the selection. For example, the find-best strategy selects the least loaded server among all servers and this strategy selects the first server whose load is below a threshold. The least loaded server has been taken into best category as it has very less load and can be selected as an appropriate server for processing the agent and responding. In find-first, we do not take into account less or overloaded factor but threshold value results in providing the appropriate select ion of the server. The very first server who is having the less threshold value will be taken into consideration.

Agent Movement Policy (AMP) determines when agent reallocation should be performed and which agent(s) (i.e., client requests) should be reallocated. Agent reallocation is activated by a threshold-based strategy. In a sender-initiated method, the agent movement is invoked when the workload on a server exceeds a threshold. In a receiver-initiated method, a server starts the process to fetch agents from other servers when its workload is below a threshold. The threshold can be a pre-defined static value or a dynamic value that is assessed at runtime based on the load distribution among the servers. When agent reallocation is required, the appropriate agent(s) will be selected from the agent queue on a server and moved to another server. Adequate administration is required for implementing this policy as threshold value is being determined statically and dynamically. This policy is initiated when congestion on a server is created, i.e., number of agents on a server is increased from given number. Then we required to move few agents from heavily loaded server to light loaded server of the same category.

C. MOBILE AGENTS (MAS)

We have identified four agents, as shown in Figure 2, two of these are MAs and other two are stationary intelligent agents. Detailed relationship among these agents are illustrated in Figure 3. MAs are- *Pattern Database Generator Agent (PDGA)* and *Pattern Based Search Agent (PBSA)*. The relationship among these agents is shown in Figure 3. These agents maintains dynamic itinerary, whenever required this can be updated at any node at any time in the itinerary. These agents maintain two containers (Result Container and State Container) one for transporting result data across the network and other for state variables and their intermediate values.

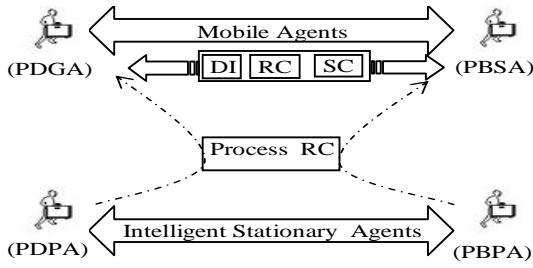


Figure 3. Mobile Agents in MADDM

Pattern Database Processor Agent (PDPA) and *Pattern Based Processor Agent (PBPA)* are the stationary intelligent agents. The PDPA and PBPA process the Result Container of PDGA and PBSA, respectively. Brief introduction of these agents and the notations used are as follows:

$\alpha \Rightarrow \{F_1, F_2, \dots, F_k\}$ – A set of files in the Public Store (PS) at each node for which Pattern Database is to be created by PDGA.

$\beta \Rightarrow \{F_1, F_2, \dots, F_n\}$, $\beta \subset \alpha$ – A set of Files containing the pattern.

PV - Pattern vector, which contains all the different patterns or words in a file that are distinguished by the blanks in between the words.

FV - Frequency vector, which contains the corresponding frequency of each pattern or word stored in the PV that is how many times that pattern exists in a file.

LV - Location vector that contains all the corresponding locations of the words stored in the pattern vector, i.e., all the locations at which that pattern or word exists in a file.

DI – Dynamic Itinerary.

PS- Public Store containing α .

PD- Patten Database containing the document vectors (File Name, PV, FV, LV) for each file containing the Pattern.

PTN- The input pattern to be searched in the Distributed Pattern Database.

SC - State container containing the state variables and intermediate values.

RC – Corresponding Result Container of the mobile agents PDGA, and PBSA to be transferred back to AS.

Pattern Database Generator Agent (PDGA) - PDGA takes as input a pattern from the client or user and then starts its journey {Figure 4}. This agent searches all the files in Public Store (PS) at a node in which that pattern exists in the itinerary. After getting all the files containing the pattern given by user, this agent generates three *document vectors* for each file. One of these document vectors is called the *pattern vector (PV)*, which contains all the different patterns or words in a file that are distinguished by the blanks in between the words. The second document vector is called the *frequency vector (FV)*, which contains the corresponding frequency of each pattern or word stored in the pattern vector that is how many times that pattern exists in a file. The third document vector is called the *location vector (LV)* that contains all the corresponding locations of the words stored in the pattern vector, i.e., all the locations at which that pattern or word exists in a file. It then creates and Pattern Database (PD) at that node and stores File Name, PV, FV, and LV in it and also

appends the contents of PD into the *Result Container (RC)*. Now PDGA is ready to move to next destination in the itinerary. When this agent visits its final destination it submits Result Container to Agent Reply Manager (ARM)[4] of the current visiting host and finally ARM sends result to the client (AS) of the agent.

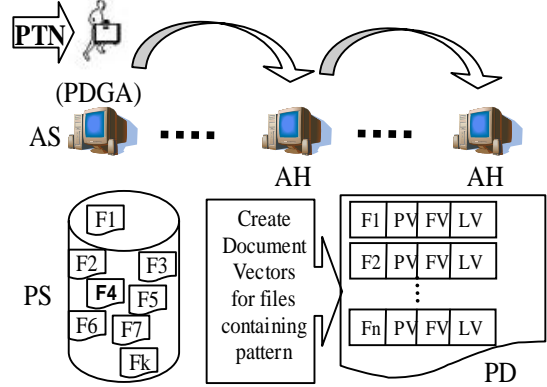


Figure 4. Working of PDGA

Pattern Based Search Agent (PBSA) takes as input a pattern from the client or user and then starts its journey {Figure 5}. This agent searches all the PV in PD on a node in which that pattern exists in the itinerary. This agent puts results (File Name, PV, FV, LV) on a node into the RC. Now PBSA is ready to migrate to next destination in the itinerary. When agent visits its final destination it submits Result Container to Agent Reply Manager (ARM) of the current visiting host and finally ARM sends result to the client (AS) of the agent.

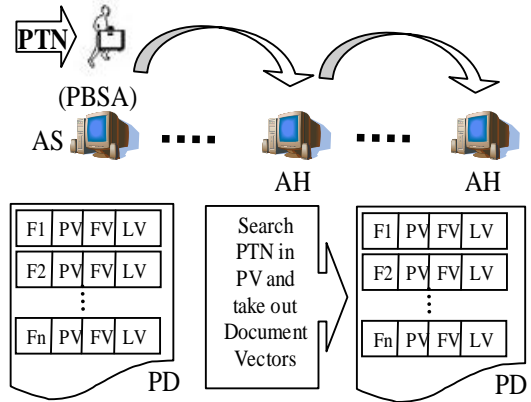


Figure 5. Working of PBSA

Pattern Database Processor Agent (PDPA) processes the Result Container prepared by the PDGA during the itinerary {Figure 6}. It retrieves three vectors from the Result Container (*document vectors*, *pattern vector* and *frequency vector*) for every visited node in the itinerary. By processing these vectors it regenerates all the original files and stores at local client node.

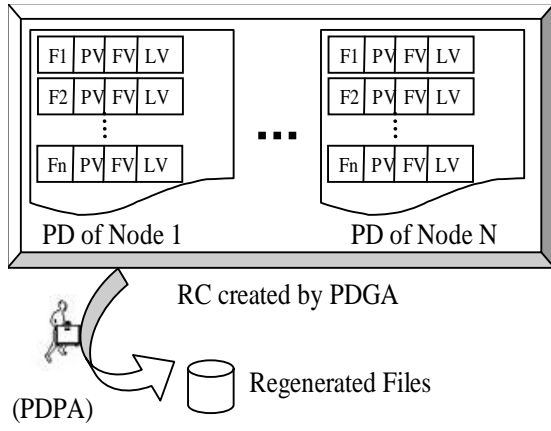


Figure 6. Working of PDPA

Pattern Based Processor Agent (PBPA) processes the Result Container prepared by the PBSA during the itinerary {Figure 7}. It retrieves three vectors from the Result Container (*document vectors*, *pattern vector* and *frequency vector*) for every visited node in the itinerary. By processing these vectors it regenerates all the original files and stores at local client node.

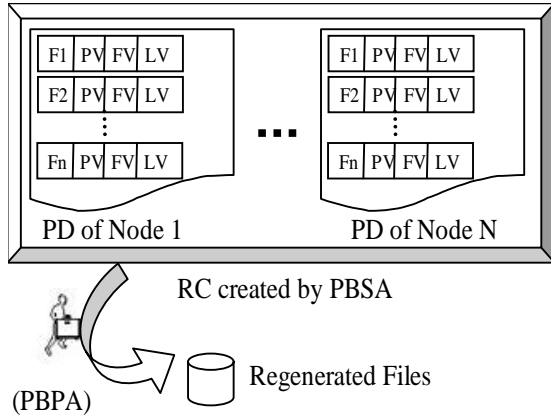


Figure 7. Working of PBPA

D. INTER-AGENT COMMUNICATION

The framework for context searching consisting of multi-agents, each agent has a specific role to play and have facility for inter agent communication as shown in Figure 2. The functions of various layers are:

Communication and Coordination Layers: Agents in the system communicate with each other or with users using mobile group approach for coordination of MA. The request an agent receives from the communication layer should be explained and submitted to the coordination layer, which decides how the agent should act on the request according to its own knowledge. We assumed a distributed system as a collection of agents, locations and communication channels. A location represents a logical place in the distributed environment where agents execute. When a MA migrates, it moves from one location to another. Agents communicate by

exchanging messages through reliable communication channels, i.e., transmitted messages are received uncorrupted and in the sequential sent order, as long as the message sender does not crash until the message is received (reliable channels is implemented over unreliable channels by tagging transmitted messages with sequential numbers, delivering such messages according to the sequential order and asking for retransmission in case of missing messages). As implied by reliable channel assumption, we assume that network partitions do not occur or, when they occur, they are repaired within a finite amount of time and communication reestablished. No bounds on message transmission or relative agent execution times are assumed. Agents and locations are assumed to fail only by crashing (without producing any further action) and the agents of a faulty location are assumed to have crashed. The failure of a given location is not directly handled. Instead, it is only detected when the associated agents are detected faulty. An agent that never crashes is named correct. Let L denote the set of all possible locations. Let A be the set of all possible agents. A mobile group is denoted by the set of agents $g = \{a_1, a_2, \dots, a_n\}$, $g \subset A$. On a mobile group, five operations are defined:

$join(g)$: issued by an agent, when it wants to join group g .

$leave(g)$: issued by an agent, when it wants to leave group g .

$move(g, l)$: issued when an agent wants to move from its current location to location l .

$send(g, m)$: issued by an agent when it wants to multicast a message m to the members of group g .

$receive(g, m)$: issued by an agent to receive a message m multicasted from the group g .

An agent a_i of a group g also installs views, named $v_i(g)$. In mobile groups a view $v_i(g)$, $\{v_i(g) \subset \{(a, l) | a \in g \text{ and } l \in L\}\}$, is a mapping between agents of group g and locations l . A view represents the set of group members that are mutually considered operational in a given instant of the group existence and indicates the locations where these members are, (a pair (a, l) in a view indicates that agent a is currently at location l). This set can change dynamically on the occurrence of agent crashes (suspicions) or when agents deliberately leave, join, or move to another location [7]. In this way these agents communicate with each other using mobile group communication defined above for updated information about all the system resources and other valuable information.

Management Layer: This layer is responsible for submitting local service information to the coordination layer for agent decision-making. In a mobile computing environment, the composition of service provider nodes is dynamic, every service provider node is likely to enter a busy state at any time and thus lower its performance, so when selecting service provider nodes for information searching, CPU utilization cannot be the sole factor of consideration for information

searching among participating service provider nodes. Other factors affecting the service provider nodes are the nodes past completion rate, possibility of the resource utilization, agent queue length, memory utilization, etc. Thus, a value function is proposed to evaluate the value of each service provider node and provide reference for selecting service provider nodes. In this value function, relative value of each resource including CPU memory, size of available memory, transmission rate, past completion time is treated as a decision variable in calculation of value function and we calculate the relative value by score of each variable, i.e., this value function is the benchmark to select or reject a particular service provider node for executing the arriving agents. In addition, to search for a service provider node that is most demanded for executing agents (information searching), different weight values will be given to such service provider nodes in accordance with the level of preference for the task, so as to select the service provider nodes most suitable for the execution of the task. Therefore, the value function is shown as given by following equation.

$$Z_i = \sum_{j=1}^n w_j f(x_{i,j}) \quad 1 \leq j \leq n$$

$$w_1 f(x_{1,j}) + w_2 f(x_{2,j}) + w_3 f(x_{3,j}) + \dots + w_n f(x_{n,j}) \quad \text{for } 1 \leq j \leq n$$

where $\sum_{i=1}^n w_i = w_1 + w_2 + w_3 + \dots + w_n = 1$, $1 \leq j \leq n$,
 $0 \leq f(x_{i,j}) \leq 1$

Where,

$f(x_{i,j})$: Score of decision variable i in node j .

Z_i : The estimated value of node i .

i : The decision variable in the value function and there are total n decision variables.

j : The node j in cluster, there are n nodes in cluster, and

w_i : The weight value of each decision variable.

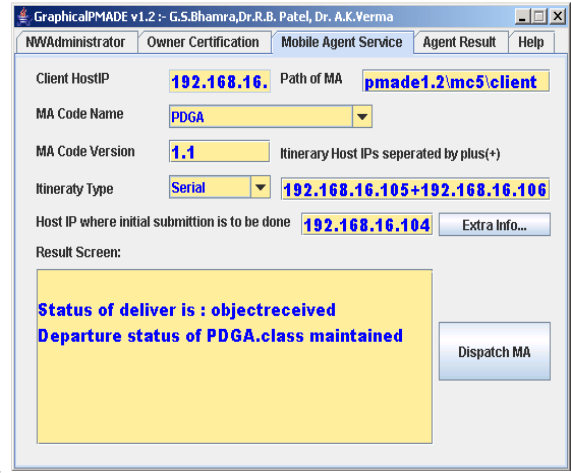
Now based upon the above defined value function for each service provider node an effective node is chosen for executing an agent and the information searching among the participating service provider nodes. Score of decision variable defines how effective that resource is with respect to available resources of a service provider node.

IV. IMPLEMENTATION AND PERFORMANCE STUDY

Internet-scale environments are subjected to greater variations due to unpredictable communication latencies, excessive resource consumption, and changing resource availability. The focus is on techniques for distributed keyword search [8], i.e., we aim to find the documents that contain a given set of query terms when the collection of documents is distributed. Formally, assuming that D_n is a set of documents that are stored on node n , and each document d is characterized by a set of keywords, the result to a query q (itself as a Boolean expression of keywords) should be the answer set $\{ (d,n) \} | n$

is a node and $q \subset s(d) \& d \in D_n$, where $s(d)$ is the unordered set of keywords in d .

We have tested developed multi-agent system over three ad hoc networks. These networks comprised of 24 nodes in all with each network containing 8 nodes. Graphical user interface for the system is shown in Figure 8. One system runs the security manager this authenticates all the clients by issuing them unique X.509 certificate. Another node is running Agent Submitter and 22 other nodes are running AH (PMADE). We have measured trip time, i.e., “the time between the dispatch of MA with request by the client and receive of response against request assigned to it” which depends on several parameters- network load, network speed/bandwidth, server speed over which agent is executing, load on the server on which agent is executed,



etc.

Figure 8. GUI of the Multi Agent System

Figure 9 shows the performance of PBSA and PDGA. When large size of data is collected, then the Agent Trip time, increases. Network traffic also play the major role, due to this reason, we have performed this analysis when network is very lightly loaded and busy and taken average case. The performance of the PDGA changes as collected data size changes. When large size of data is collected the Agent Trip time increases.

When we compare the performance of both the agents, then the Trip time in case of PDGA is larger than that of in the PBSA. The reason behind this is that the PDGA has to perform more computations in processing the files for generating the Pattern, frequency and locations document vectors. So the agent trip time in case of PDGA comes out to be larger. But the advantage of using the PDGA over the PBSA is that the PDGA consumes less network bandwidth because it transfers the processed document vectors of a file, which are compacted in size; rather than the complete data of the file. Thus, based upon our requirement, we can use any agent system.

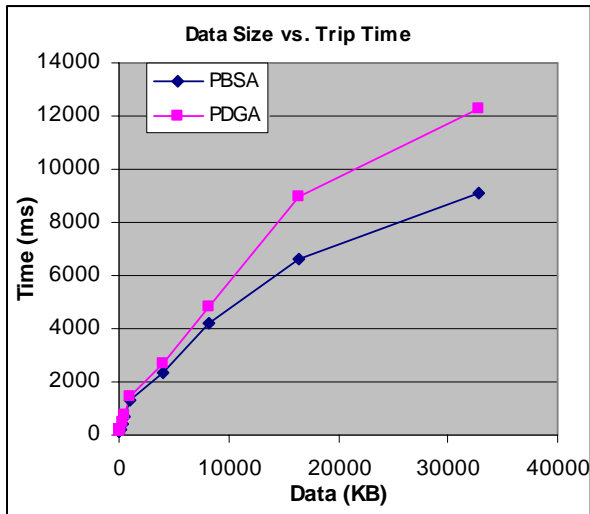


Figure 9. Performance of the PBSA and PDGA

RELATED WORKS

A consequence of the proliferation of Internet technology is the availability of a vast amount of data. Recent trends in accessing this data propose the use of mobile agent technology. The most important advantages of this approach are imposed by the aspect of mobility which allows to considerably reducing the amount of data that has to be shipped over the network and to reduce network latency. By using mobile agent technology, a user no longer has to manually browse for certain data and/or services but rather to submit a personal MA that accesses and processes information on her/his behalf (i.e., places orders or bids in some electronic auctions, or fixes dates). To this end, MAs operate on top of a peer-to-peer network spanned by the individual providers of data and services.

Issues related to context based searching are divided into two categories- Contextual Aspects of Relevancy and Similar vs. Relevant Resources [9]. A few context based search mechanisms are available PeerSearch [11, 12] and PlanetP [12]. Different from our proposed system most current context based search systems use the inverted indexing mechanism that associates each keyword with a set of files. There are mainly two approaches to content based search: Distributed indexing and global directory. In Distributed Indexing each node maintains a small part of the network indices, while in a Global Directory [11] approach each node stores all network indices.

Most current MA [10] approaches focus on the support of applications which gather and filter data. However, in order to support another practically relevant class of MA applications, the aspect of *information processing* has also to be considered. The extension of the capabilities of MASs from information gathering towards information processing in a peer-to-peer-based system environment now goes along with additional requirements that users impose on their MA applications. Essentially, MAs have to leave the system in a consistent state, even in the case of failures or when different agents access shared resources and/or services concurrently. This urgently requires support for correct concurrency control and failure

handling. Such requirements are well understood in the development of our multi-agent system. Performance of the developed system shows that it is very useful when it will be transported over Internet.

A DDM system is inevitably a very complex entity that is comprised of many components; mining algorithms, communicative subsystems, resource management, task scheduling, user interfaces, etc. It should provide efficient access to both distributed data and computing resources, monitor the entire mining procedure, and present results to the user in appropriate format. BODHI[15], Java agent for Meta-learning (JAM)[17], Papyrus[18], InfoSleuth[20], PADMA[21], Parallel and Distributed Data Mining Application Suite (PaDDMAS)[22], WoRLD[23], Distributed Knowledge Networks (DKN)[24] are some of Distributed Data Mining Systems.

CONCLUSION

In this paper, we present a Mobile Agent based Distributed Data Mining System (MADDM) to extract some trends or patterns representing the knowledge from the Distributed Pattern Databases on Heterogeneous Network. In this system currently we have identified four agents for four different operations. Other system components (Policy, Agent-Agent, Interface, and Agent-Agent Communication layer) provided in the framework to help the MAs to improve their performance as well as their autonomy and pro-activity for the navigation through the agent system infrastructure. For the evaluation, our focus was the quality of the modules as well as their performance advantage for MAs. Currently we are in the process comparing the performance of our system with google search engine.

REFERENCES

- [1] Picco, G.P., Mobile Agents: An Introduction. *Microprocessors and Microsystems*, 25, 2001, 65-74.
- [2] Tripathi, R., T. Ahmed and N.M. Karnik, Experiences and future challenges in mobile agents programming, *Microprocessors and Microsystems*, 25, 2001, 121-129.
- [3] Vigna, G., Mobile code technologies, paradigms and applications, Ph.D. Thesis, Politecnico di Milano, Italy, 1998.
- [4] Patel, R.B. and K. Garg, A new paradigm for mobile agent computing, *WSEAS Transaction on Computers*, 1, 2004, 57-64.
- [5] Erfurth, C. and W. Rossak, Characterization and management of dynamical behavior in a system with mobile agents, In *Innovative Internet Computing System-Second International Workshop, IICS 2002*, Kuhlungsborn (Germany), T. Böhme and A. Mikler (Eds.), H. Unger, June 2002, LNCS 2346, 2002, 109-119, Springer-Verlag.
- [6] Patel, R.B., Design and implementation of a secure mobile agent platform for distributed computing, Ph.D. Thesis, Department of Electronics and Computer Engineering, IIT Roorkee, India, 2004.
- [7] Raimundo, J., A. Macêdo, F.M. Assis Silva, The mobile groups approach for the coordination of mobile agents, *J. Parallel Distributed Computing*, 65, 2005, 275-288.

- [8] Demetrios Zeinalipour-Yazti, Vana Kalogeraki, pFusion: A P2P Architecture for Internet-Scale Content-Based Search and Retrieval, *IEEE Transactions on Parallel and Distributed Systems*, 18(6), July 2007.
- [9] Dichev C., Dicheva D., Context-based Search in Topic-centered Digital Repositories, in *Proceedings of 3rd International Semantic Web User Interaction Workshop (SWUI06)*, collocated with 5th International Semantic Web Conference (ISWC'06), Athens, GA, November 6th, 2006.
- [10] Klaus Haller, Heiko Schuldt Hans-Jörg Schek, Transactional Peer-to-Peer Information Processing: The AMOR Approach, in *Proceedings of the 4th International Conference on Mobile Data Management (MDM'2003)*, Melbourne, Australia, January 2003, LNCS 2574, pp. 356-361, Springer-Verlag.
- [11] C. Tang, Z. Xu, M. Mahalingam, PeerSearch: Efficient Information Retrieval in Peer-To-Peer Networks, in *Proceeding of HotNets-I, ACM SIGCOMM*, 2002.
- [12] Maxim Rodionov and Hui Siu Cheung, Intelligent Content-Based Retrieval for P2P Networks, in *Proceedings of International Conference on Cyberwords (CW'03)*, pp. 318-325, 2003.
- [13] R. B. Patel and K. Garg, A Flexible Security Framework For Mobile Agent Systems, *Control and Intelligent Systems*, 33(3): 175-183, 2005.
- [14] Byung-Hoon Park and Hillol Kargupta. Distributed data mining: Algorithms, systems, and applications. *Data Mining Handbook*, 2002.
- [15] H. Kargupta, B. Park, D. Hershberger, and E. Johnson. *Advances in Distributed and Parallel Knowledge Discovery*, chapter 5, *Collective Data Mining: A New Perspective Toward Distributed Data Mining*. AAAI/MIT Press, 2000.
- [16] Matthias Klusch, Stefano Lodi, and Gianluca Moro. *The Role of Agents in Distributed Data Mining: Issues and Benefits*
- [17] S. Stolfo A. L. Prodromidis, S. Tselepis, W. Lee, D. W. Fan, and P. K. Chan., "JAM: Java Agents for Meta-Learning over Distributed Databases," *Proc. 3rd Int'l Conf. Knowledge Discovery and Data Mining (KDD 97)*, AAAI Press, 1997, pp. 74-81.
- [18] S. Bailey, R. Grossman, H. Sivakumar, and A. Turinsky. Papyrus: a system for data mining over local and wide area clusters and super-clusters. In *Proc. Conference on Supercomputing*, page 63. ACM Press, 1999.
- [19] Fayyad, U., Piatetsky-Shapiro, G., Amith, Smyth, P., and Uthurusamy, R. (eds.), *Advances in Knowledge Discovery and Data Mining*, MIT Press, Cambridge, 1996.
- [20] Martin, G., Unruh, A., & Urban, S., An agent infrastructure for knowledge discovery and event detection (Tech. Rep. No. MCC-INSL-003-99). Microelectronics and Computer Technology Corporation (MCC).
- [21] Kargupta, H., Hamzaoglu, L., Stafford, B., Hanagandi, V., & Buescher, K. (1996). PADMA: Parallel data mining agent for scalable text classification. In *Proceedings of conference on high performance computing '97* (pp. 290-295). The Society for Computer Simulation International.
- [22] Rana, O., Walker, D., Li, M., Lynden, S., & Ward, M. (2000). PaDDMAS: Parallel and distributed data mining suite. In *Fourteenth international parallel and distributed processing symposium* (pp. 387-392). Cancun, Mexico.
- [23] Aronis, J., Kulluri, V., Provost, F., & Buchanan, B. (1997). The WoRLD: Knowledge discovery and multiple distributed data bases. In *proceeding of florida artificial intelligence research symposium (FLAIRS-97)*.
- [24] Honaver, V., Miller, L., & Wong, J. (1998). Distributed knowledge networks. In *IEEE information technology conference*. Syracuse, N.Y.
- [25] H. Kargupta, B. Park, D. Hershberger, and E. Johnson. *Advances in Distributed and Parallel Knowledge Discovery*, chapter 5, *Collective Data Mining: A New Perspective Toward Distributed Data Mining*. AAAI/MIT Press, 2000.