International Journal of Computational Intelligence Systems, Vol. 7, No. 3 (June 2014), 432-455

Local Semantic Indexing for Resource Discovery on Overlay Network Using Mobile Agents

M. Singh [1] *, X. Cheng [2] , R. Belavkin [2]

[1] *School of Engineering and Information Sciences, Middlesex University, The Burroughs, Hendon, London, NW4 4BT, United Kingdom*†

E-mail: ms549@live.mdx.ac.uk

[2] *School of Engineering and Information Sciences, Middlesex University, The Burroughs, Hendon, London, NW4 4BT, United Kingdom*

E-mail: x.cheng/r.belavkin@mdx.ac.uk

Received 25 May 2013

Accepted 27 September 2013

Abstract

One of the most crucial problems in a peer-to-peer system is locating of resources that are shared by various nodes. Various techniques suggested in literature suffer from drawbacks viz. saturation of network, inability to locate multi-keyword based resource or locate resource based on semantics. We present the solution that is more efficient and effective for discovering shared resources on a network that is influenced by content shared by nodes. To reduce the search load on nodes that have uncorrelated content, an efficient migration route is proposed for mobile agent that is based on cosine similarity of content shared by nodes and user query and minimum support. Results show reduction in search load and traffic due to communication, and increase in locating of resources defined by multiple keys using mobile agent that are logically similar to user query. Furthermore, the results indicate that by use of our technique the relevance of search results is higher; that is obtained by minimal traffic generation/communication and hops made by mobile agent.

Keywords: Resource Discovery, Overlay Network, Reconnaissance Agent, Latent Semantic Indexing, Cosine Similarity

1. Introduction

The amount of information that is hosted by servers on network is increasing exponentially. This information though published is not available to users on network immediately. The solution is offered by peer-to-peer systems as opposed to centralised systems where it is not always possible to index the shared resources immediately largely due to the fact that large indexing database has to be updated. The peer-to-peer (P2P) systems offer solution for resource discovery by making the task of hosting distributed. However, it suffers from inefficiency to locate hosted resources.

The P2P systems consist of number of decentralised nodes sharing their resources on an overlay network. Here the resources mean services/files that nodes on the network host. P2P systems offer low cost sharing of information and with high autonomy but discovery of this

* School of Engineering and Information Sciences, Middlesex University, The Burroughs, Hendon, UK
† Research Student at Middlesex University, United Kingdom

shared information is not very efficient.

A classical client and server based centralised solution to a location of resource is offered by Napster.[1,2] In this approach, a client connects to central server and also indexes resources and their location. Upon query about resource location from any other client, the central server issues the IP address of the client where resource is located. This solution cripples autonomy of a client due to centralised sever, as in case of server failure, clients cannot locate resources.

Another approach to resource location is offered by Gnutella, where the decentralised peers communicate to other peers when the resource location query is issued by user.[3,4] This solution offer high degree of autonomy as peers can join or leave the overlay network without effecting rest of the network. When locating a resource, peer floods the user query on the overlay network usually with time-to-live constraint in order to query other peers about required resource. The inefficiency in this approach attributed to three facts:

1. the overlay network is created randomly as there is not structure associated with it
2. the queries for a resource location are forwarded "blindly" from one peer to another peer using technique called flooding due to which there is an unnecessary quantity of message on the network
3. saturation as number of nodes increase.

A more "rigid" approach is taken by a structured overlay that is based on hash functions supports key-based routing such that resource identifiers are mapped to the peer identifier address space and a resource request is routed to the nearest peer in the peer address space.[5,6,7,8] Although such systems are better than unstructured overlay from performance point of view as some heuristics are available for locating a resource (only where the search keys are known exactly), but they are not as effective for approximate, or text based resource location.

The purpose of this paper is to offer the multi-agent system (MAS) and the resource discovery method that overcomes the disadvantages of structured overlay i.e. be able to locate resources even when the keys are unknown, approximate, or text based multiple keys and also offer the flexibility characteristic of autonomous unstructured overlay but by reducing number of message on the network and control or remove unnecessary flooding.[9]

Through this paper we propose the following:

- a flexible multi-agent based approach for dynamic organisation of P2P network that is based on the similarity of content shared by peers. The similarity of content between two or more peers is translated into similarity between peers or a cluster of peers sharing similar content.
- the resource location mechanism that uses semantic similarity between content shared by peers and search keywords deterministically to route a mobile agent called the reconnaissance agent (RA) to peers that host content that is similar to a user query.
- the use of latent semantic indexing(LSI) by RA to locate resource hosted by peer that is best match for a user query (where the user query can be text based or an approximate query).

We demonstrate that this method improves the resource discovery performance i.e. finding a resource with minimum communication and hence reducing search load.

The rest of the paper is structured as follows.

Section 2 surveys the current literature and draw lessons to propose the capabilities that a resource discovery system should possess. In doing so, section 2 also collates a large amount of research work relevant to field of study.

Section 3 describes the design features of proposed MAS based resource discovery system, node clustering based on semantic similarity of content hosted by nodes and RA routing, and the multi-agent collaboration for resource discovery. Furthermore, it describes the implementation done using Java Remote Method Invocation (RMI) and Java Agent Development Framework (JADE).[10]

Section 4 is dedicated for experimentation where the effectiveness of our resource discovery algorithm and resource locating algorithm is compared against flooding (Gnutella) in terms of response time and search load. Furthermore, our node clustering algorithm for routing the RA on the overlay network, messages on network and relevance of results obtained due to user invoked query is compared to some contemporary research work done by other researchers in field of using mobile agents for resource discovery.

Section 5 is dedicated for discussions for assembling and comparing our methodology to other related works in the field of resource discovery

Finally, section 6 provides list the conclusions.

2. Literature Survey

There are diverse set of solutions that are available for resource discovery.[9] These solutions are characterised through the routing strategy and resource searching strategy that is applied by them.[11] We have categorised and reviewed the resource searching techniques used by unstructured and structured P2P systems by initially discussing architectures. We also present most current search techniques that are being introduced to the resource discovery domain.

2.1. Architectures for P2P Overlay

2.1.1. Centralised Client-Server Network

The first most popular P2P Network was Napster which used Central Indexing Server for storing the locations of the resources.[2] Using this network Napster client's in the network can communicate with the other Napster clients. In Napster a dedicated central server maintains an index of the files shared by the active peers on the network. Each peer in the network maintains a constant connection to one of the central server through which the query of file location is sent. When a central indexing server receives the query for a file location it cooperates to process the query and returns the corresponding matching file locations to the peer making the query. After the peer making query receives response from the indexing server about the list of locations of the resource, the peer can now make direct communication with the peers having the resources and initiate the transfer of the resource. Besides maintaining the list of resources in the network, the indexing server also keeps track of each peer that is active or monitors the state of the peer like keeping track of the information of the peer for instance the duration the peer has been active or the connection speed of the peer.[1]

2.1.2. Unstructured P2P System

An unstructured overlay like Gnutella is organised into random graph topology where there is no specific topology that the overlay network follows and it uses flood or random walks to discover resource in the network. This overlay is constructed easily when a node wants to join the network. During the resource discovery each node visited will evaluate the query locally on its data store. Before starting to exchange messages between the servants, a Gnutella servant connects itself to the network by connecting with another servant on the network. Once the connection is established, the addresses of one or more host will be supplied as the servant joins the network. Generally Gnutella works on Port 6346 which is same in our experiment as well. The listening servant is advertised by Pong messages. When another servant is located on the network TCP/IP connection is established and a handshake sequence is initiated. Details of Gnutella resource discovery protocol are discussed in Section 2.2.1.

2.1.3. Structured Network

A structured overlay and DHT based systems like Chord, Pastry, CAN, and Tapestry is the improvement on unstructured overlay to improve the performance of resource discovery.[5,6,7,12] It ensures that any node can efficiently route a search to some peer that has the desired file even in the rare availability.[13] The nodes in the network impose constraints on the topology as well as on the data placement to provide with efficient search mechanism and resource discovery. In all the DHT systems mentioned above files are associated with a key and each node in the network is responsible for storing list of resources hence having list of keys. The first and foremost operation in the DHT system is the look up for the key as lookup(key) which is supposed to return a location of the resource or the key and hence IP address.

Till date there are many load balancing approaches, Chord was the first to propose the concept of virtual servers and hence address the load balancing by having each node simulate a logarithmic number of virtual servers.[14] Using Chord, only $\log(N)$ messages are required to find the resource in the Chord Network where N being the number of active nodes in the network. Chord allows distributed nodes to agree on a single Chord node as a rendezvous point for a given key without any central coordination.[15,7] Chord algorithm does not particularly specify any means for storage of the resource; this is done my DHash which is built on top of Chord and also handles storage of data blocks on the active nodes reliably.[15,7] This is achieved using techniques like replication and erasure coding. Stoica et al show the logical application interface as: $Key = put(data)$ and $Data = get(key)$.[7]

Pastry is completely decentralized, scalable and self organizing network which dynamically adapts to the addition or removal of nodes.[16] Each node in Pastry Network has unique and random identifier called NodeId in a circular 128-bit identifier space. With a message and a numeric 128-bit key, a node can route the message to a node with Nodeid which is numerically close to the key within the live Pastry Network.[6] This results in first order balancing of the storage requirements and query among the nodes in the Pastry network and also does not require global co-ordination.[6]

Routing in Pastry For a given message it checks the following conditions:[16]

- If it falls within the nodeid's leafset then the message is directly forwarded to it.
- Else, the message is forwarded to a node that shares the most common prefix with the key using the routing table
- Else if the routing table is empty or the node is unreachable, then message is forwarded to node that is numerically close to the key.

If given N as number of live nodes in the overlay Pastry Network then expected number of forwarding steps are $\mathcal{O}(\log N)$ and size of routing table for each node is $\mathcal{O}(\log N)$.[6]

Content Addressable Network (CAN) is also a distributed system which is DHT based that maps keys to values on big scale network like internet. As discussed above CANs basic idea is to build a hash table and the basic operations performed are insertion, lookup and deletion of the key, value pairs. In the CAN network each node stores a chunk (also called zone) of the total hash table. Moreover it stores smaller amount of information of adjacent zones.[5]

In CAN the network is formed in a tree like structure where each node is associated to one in the parent level and to a group in a child level. When a query is made, it travels from the top most level going down through the network until the resource is discovered or until the last leaf is reached.[16] The architecture of the CAN is a virtual multi dimensional can be viewed as Cartesian coordinate space. CAN design centres around a virtual d-dimensional Cartesian coordinate space on a d-torus which is independent of the physical location and physical connectivity of the nodes.[5] The overall Cartesian coordinate space is dynamically partitioned among all the nodes such that each node belongs to one distinct zone with in the entire space.[5] To route a query, node maintains a routing table which holds the IP locations as well as the virtual co ordinate zone of each of its neighbour. Using the co ordinates the message is routed towards destination.

CAN construction take place in three steps:

1. A joining node must find a node which is already on the CAN network
2. Using the CAN routing mechanism, it must find a node whose zone will be split
3. Lastly, the neighbours of split zone are informed.

Tapestry another P2P structured overlay network which provides high performance, scalable as well as location independent routing of the messages. It uses adaptive algorithm with soft state to maintain fault tolerance with regards to changing node membership and network faults. Tapestry provides decentralized object location and routing (DOLR), the DOLR interface provides routing of messages to end points like nodes or object replicas.[12] Each Tapestry node is assigned a unique id and more than one node can be hosted by a single physical host. Tapestry utilizes identifier space of 160 bit values with a 40 digit key. The efficiency of the Tapestry increases with the increase in the network size. Moreover to allow multiple applications every message contains an application specific identifier which helps the node to select a process or delivery of message to a specific port.[12]

Table 1 shows the classification of P2P routing infrastructures in terms of their network structure, with typical examples. Table 2 summarises infrastructure for routing and resource discovery location.

	Centralisation		
	Hybrid	Partial	None
Unstructured	Napster	Kazaa, Edutella	Gnutella
Structured			Chord, CAN, Tapestry, Pastry

Table 1. A classification of P2P routing infrastructures in terms of network structures

P2P Infrastructure	Description for Routing and Location
Flooding	Infrastructure that provides functionality for searching "blindly" on overlay networks.
Chord	A scalable peer-to-peer lookup service. Given a key it maps the key to a node.
CAN	Scalable content addressable network. A distributed infrastructure that provides hash-table functionality for mapping file names to their locations.
Pastry	Infrastructure for fault-tolerant wide-area location and routing.
Tapestry	Infrastructure for fault-tolerant wide area location and routing.

Table 2. Summary of infrastructure for routing and resource discovery location

2.2. Resource Discovery and Routing

Table 3 compares various features of routing algorithms used in P2P systems.

2.2.1. Resource Discovery in Unstructured P2P Systems

In unstructured P2P systems for instance Gnutella, various nodes(peers) are organised into a random graph where the edges of the graph are the links between various nodes this constructing an overlay network.[3,4] Flooding technique is used for routing a query through the overlay network. Upon query, the visited node compares the query against its shared resources and is then requested to forward the query to its neighbours. This system of resource discovery is highly robust and offers vast improvement on factor of scalability as compared to Napster or other centralised search systems but suffers from an expensive cost of saturation of overlay network due to large bandwidth consumption.

Feature	Conventional Flooding	Random Walks	DHT	Range Query
Infrastructure	Performed on unstructured P2P networks	Performed on unstructured P2P networks	Performed on structured (DHT) P2P networks	Performed on structured (DHT) P2P networks
Scope	Works same with any network	Works best with multiple queries and peer clustering	Works same with all the DHTs	Works best when semantic proximity of keys is maintained
Search Complexity	On average search is done in $k*N$ time (k = average degree of nodes, N = total number of nodes	On average search is not in *log* time	On average search is done in *log* time	On average search is done in *log* time
Relevance and Results	Returns single result	Returns single result	Returns single result	Returns a set of results
Cost Associated with Resource Discovery	Very wasteful of resources, as every peer processes each query	Less taxing in resources	Not too taxing on resources	Min-max algorithm uses resources wisely
Response Time (Routing and Searching)	Is not very fast, as every peer processes each query	Result are reasonably fast	Routing is very fast	Shower algorithm is very fast

Table 3. Comparison of features of routing algorithms

Various techniques have been introduced to improve the routing efficiency of this system that includes random walks described in Ref.16 , informed searches described in Ref. 17 , and node grouping described in Ref. 18 and Ref. 19 .

Random walks were introduced in Ref. 16 , to improve the issue of saturation by introduction of techniques *time-to-live* (TTL) and checking. Like flooding, random walks is uninformed search technique where the query is randomly forwarded to nodes. As an answer to saturation of the overlay network, the total number of nodes to be visited is defined using TTL. Also, *checking* technique is used where before forwarding to next node, the query originator is "checked with". These techniques of controlled flooding refined resource searching mechanism but suffered from lack of results due to restrictions imposed by TTL.

To increase the effectiveness of search mechanism, informed searches were introduced that offered improvement in performance by using information on nodes and their resources.[21] This information is collected as part of previous queries. In Ref. 17 Crespo et. al. introduced the technique *routing indices* (RI) for informed searches, where queries are routed to nodes that were more likely to provide a resource. In this technique uses distributed-index mechanism that maintains indices on each node. Given a query, the RI data structure returns a list of ranked nodes for forwarding a query. In informed searches, propagating a query to nodes where there is likeliness of discovering a resource help reduce the network load because of less flooding.

Other resource location techniques such as *SETS* and *ESS*, are based on a concept of grouping content to organise nodes.[19,22] The search in SETS is based on topic-segmentation of overlay network. In other words, SETS partitions nodes into topic segments such as nodes with similar content belong to same segment.[20] SETS suffer from single point failure and hence has performance bottleneck.[20] ESS is based on information retrieval algorithms to perform resource discovery on Gnutella-like P2P systems. As in SETS, nodes with similar content are segmented into same semantic group.[22] The concept used by ESS is to place indexes of semantically close files into same nodes with high probability of exploiting information retrieval algorithms and locality sensitive hashing.[23]

A multiple keyword based searching technique called *local indexing* is used for locating resource using multiple keywords.[24] In this approach, the record of terms contained in each resource is stored on that particular node. Upon query, the search keywords are forwarded to each node using flooding technique, where they are compared for relevance. This technique is effective for getting better search results but suffers from classical saturation factor on overlay network.

2.2.2. *Resource Discovery in Structured P2P Systems*

Structured P2P systems have been proposed to provide a more scalable solution as compared to first generation unscalable unstructured P2P systems. In structured systems, a node is associated with keys and their values. When a query is presented it is changed into the search for the key. The hash table on the peer is used pass the query forward to other peer whose address is numerically closer to requested key. The examples of structured systems are Chord, and CAN. In hybrid systems for instance Pastry as described in Ref. 6, the routing structure is comparatively more fluid as compared to Chord as the routing table can suggest the routing of the query to any node that is part of the defined subspace.[25]

Structured systems perform better than unstructured systems with respect to scalability, as DHT has many advantages, such as scalability, load balancing, logarithmic hop routing, fault tolerance, and self organising nature.[9] Although self-organising works as the advantage but as each peer must periodically update all its neighbours and hence result in increased traffic.[26] When the nodes leave or join the network the updated index need to be redistributed and hence the tables need to restructure. This is not the case in unstructured systems as node can leave or join the network without sending stabilisation message. Unstructured systems have provided many strategies for reducing traffic like dynamic querying, routing indices, and super-peers architectures.[3,11] Structured systems have advantage over unstructured systems as these systems provide ability to route the queries in very small number of hops. DHT-based systems are known for exact-match lookups, given a query both Chord and Pastry resolve the queries in $\mathcal{O}(\log(n))$, while CAN requires $\mathcal{O}(n^{\frac{1}{d}})$ steps, where n is number of nodes and d is number of dimensions in CAN.[7,5] As the peers and the resources are based on the hash function – key generated by the hash function is very specific.[7,5] As the queries may not

be exact, it may be difficult to find the resource in the structured network.[26,9]

However, in *keyword-search* the queries do not have to be exact and can comprise of multiple-keywords. The information retrieved in such scenario consists of a set of resources that match the criteria given as a query. The proposed system that support keyword-search on top of DHT-based structured P2P system are categorised by their indexing technique viz. *global indexing* for instance in Ref. 23, Ref. 27, Ref. 28, and Ref. 29 , and *hybrid indexing/optimised-hybrid indexing* for instance in Ref. 23, Ref. 30, and Ref. 31 .

In *global indexing*, the inverted list record is maintained on every node - information about nodes that contain a particular term. Upon query that contains multiple keywords, the query is routed to node containing that keyword. Then the inverted lists are intersected to find resource that contains the requested keywords. This largely reduces the number of nodes that need to be visited, however large amount of communication is introduced during intersecting phase. Moreover, communication cost grows with increase in length of inverted list.[24,23]

In *hybrid indexing*, each node holds the complete inverted list of terms describing the resources on that node and also the inverted list of terms that are forwarding terms for resources shared on this node. Given a multiple keyword based query, the query is routed to node containing the search keywords. Then, this node performs a local search without connecting to other nodes about list of forwarding nodes by querying the inverted list of each found resource on this node. The efficiency of this type of indexing is higher than that of global indexing but suffers from increased cost of publishing term data.[20]

In *optimised hybrid indexing*, the terms that describe a resource is published under resource's top terms (terms that are central to a resource).[24] Clearly, the search may be degraded because of limiting the publishing of keywords under resource's top terms.[20]

Another effective way for resource discovery process is to establish *semantic links* between the nodes that are based on node properties which are described by the resources shared by those nodes.[32,33,34,35,36] In Ref. 33 , the semantics information is used for searching resources in a scalable manner. A. Crespo in his work described in Ref. 34 suggests the semantic overlay network (SON) where the peers are organised based on logical similarity between the content. Semantic information can be used to create P2P networks that are more organised than unstructured overlay and are capable of handling multiple keys for finding resource on network unlike structured overlays. Locality awareness is another version where the peers are organised based on matching tags that are used to describe a resource.[32] pSearch in Ref. 35 introduces the concept of semantic overlay on top of a DHT based structured P2P system. In this overlay, the resources are organised based on their semantic vectors (such as distance). pSearch proposed to integrate semantic storage and retrieval capabilities into CAN, where resource index is stored by using its vector representation as coordinates.[37] GloServ in Ref. 36 uses a keyword-based search on a hierarchical hybrid P2P network to build semantic overlay between nodes that operate in the same domain.[21] Even though this attempt at creating semantic links between nodes and resources may help improve the resource discovery, but no test results have been published yet by the authors.

Both structured and unstructured systems heavily rely on stationary software modules. These modules keep track of all resource discoveries. They use the host computer resources and can potentially drain the local resources and may cause failure of host computer. Backbone of both approaches is P2P communication. P2P communication blurs the distinction between client and server computers. This can potentially saturate the network. Unstructured resource discovery has a linear connection between computers where each computer knows the ping computer. Failure of any computer in the chain results to loss of all down stream resources.

2.2.3. Resource Discovery in Mobile Agent Systems

As an alternative to stationary software modules, multi-agent systems offer following merits make mobile agents in particular suitable for resource discovery in P2P systems:[38]

- Asynchronous: After a mobile agent is dispatched, there is no need for the creator peer to keep track of mobile agent. The thread can be completely released. Theoretically speaking, the creator peer does not even need to remain connected to a network. A mobile agent will perform the given tasks completely in parallel with the creator peer as a separate thread. After all of the

tasks have been fulfilled, mobile agent will return to the creator peer (when it is connected to the network).

- Autonomous: Mobile agents can compute its itinerary as it progresses through the network. It is able to choose the next site according to conditions it is learnt about, and history of visited peers and current peer. Mobile agents may also visit peers that were unknown when it was originally dispatched, which in particular suits network based resource discovery.
- Compatibility: Agent based systems can be combined with successful features from other resource discovery systems.
- Bandwidth Consumption: The mobile agents for resource discovery require lesser bandwidth. As opposed to the multiple interactions between peers, mobile agent packs these interactions and sends them as discrete piece of traffic. Also mobile agents are much smaller in size and grow dynamically as they accommodate more data. In structured or unstructured systems, the communication is synchronous which is not the case with mobile agent which can encapsulate its state and carry on the execution on the different node asynchronously.[10]

Dasgupta et al in Ref. 39 and Kambayashi et al in Ref. 40 are multi-agent systems (MAS) introduced for resource discovery. Both systems are inspired from ant communities for development of their P2P system. They use *Anthill* described in Ref. 41, Ref. 42 and Ref. 43 MAS that emulates the resource coordination behaviour as observed in ants. In this MAS P2P system resources are known as nests and user request to locate resources is carried out by ants. Upon query, the ants visit various nests on overlay network. Ants restrict from communicating to each other but leave information about the service they are implementing in the resource manager found at each nest site. The behaviour has analogy to pheromones that has advantage of allowing network to self-organise over a period of time.[21] Ants greatly improve upon the flooding issue raised in unstructured P2P systems as only one ant visits the nest at one time. The next nest chosen for ant to visit is either deterministic or random, which means that search performance may be slow. This is observed in Ref. 39 and Ref. 40 , where overlay network becomes more "knowledgeable" over a period of time. To improve upon this disadvantage, Kambayashi et al in Ref. 40 build their P2P system on top of structured P2P system called Chord.[7] Mobile agents (ants) in their system may use <key, value> map to find resource in cases when deterministic path cannot be calculated. Kambayashi et al also use indexing (TF-IDF) to calculate logical distance between two nests based on correlation between keywords shared between nodes. The correlation is calculated using primitive form of Jaccard similarity.

2.3. *Critical Review and Observations*

For our work, we understand from the literature survey that semantic links between the nodes is useful for resource location and for node coordination - to be used for deterministic routing of the query. We understand that MAS and mobile agents offer nodes a greater degree of autonomy. We also understand that there is not enough work in the area of overlay network self-organisation. Our objective is to reduce the search load by reducing number of messages or number of hops made by mobile agent during migrations from one node to another. We also aim to dynamically self-organise overlay network as and when new nodes join or new resources are published by the nodes. We further aim to exploit heterogeneity of resources hosted by nodes on overlay network to locate resources in minimum number of hops.

We have conceived multi-agent resource discovery system using mobile agent called Affinity that

1. captures the features of clustering of peers based on semantics of content shared,
2. handles multiple keys to locate a resource by use of LSI similarity, and
3. finally reduce the bandwidth consumption by providing mobile agent with ability to negotiate with peers regarding finding next site for migration and matching resource hosted by peer to user query under given constraints from user.

3. The Proposed LSI based MAS Resource Discovery - Affinity

3.1. *The Proposed Global System Architecture*

The architecture for the conceived system is illustrated in Figure 1. As articulated in the figure, the system has four layers - *interface layer*, *reconnaissance layer*, *directory and resource layer* and *visiting agents layer*. Each layer contains agents dedicated to perform certain task (detail specifications of agents are provided in Section 3.2).

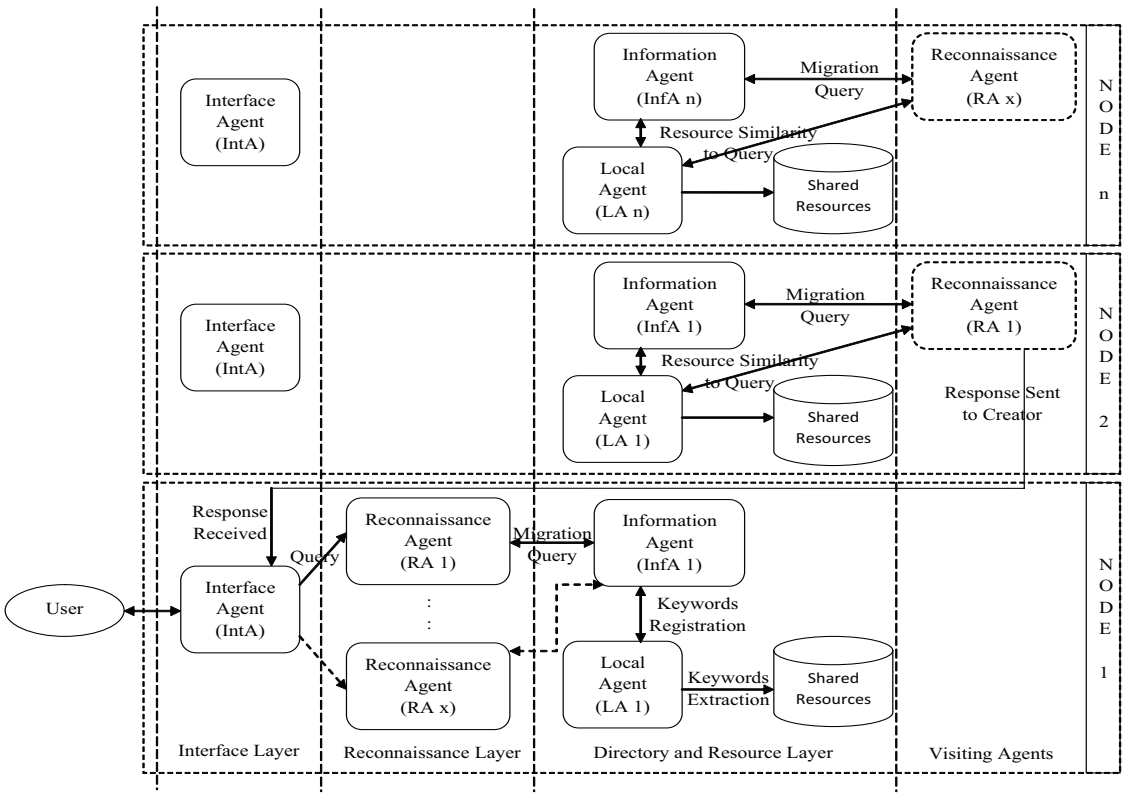

Fig. 1. Global architecture of the system

The purpose of each layer is as follows:

- *Interface Layer*: This layer contains the interface agent that is used by the client to interrogate the system. The goal is to capture the requirements or needs of the user and respond back to them appropriately. User's interaction with system is through interface agent that helps in realisation of the given task. The request from user i.e. the search query facilitates the function of reconnaissance layer. The additional function of transformation of the submitted user request into a feature vector is also realised in layer.
- *Reconnaissance Layer*: This layer contains the reconnaissance agent that is created as a result of submitted query in the interface layer. The function of this layer is to temporarily contain the new created mobile agent while it communicates to stationary agents in directory and resource layer for node address where it can migrate to in order to realise the submitted query.
- *Directory and Resource Layer*: The function of this layer is to receive requests from reconnaissance agent, process them and return the results. This layer holds two stationary agent - local agent and information agent and is responsible for managing the data associated to shared resources on the node and multiple sources of node addresses that are semantically similar to content shared on this node. The task of determining appropriate node address and hence deterministic route to the node that hosts resource similar to given query is completed in this layer. The management of directory of shared resources on this node that are transformed into feature matrix after indexing is the function of local agent and the management of list of peers that are semantically similar to content of this node is done by information agent. The functionality to achieve autonomy is also achieved on this layer where information agent communicates to bootstrap server about its status every $300,000ms$.
- *Visiting Agent Layer*: The function of this layer is to provide platform for the migrated reconnaissance agent that is visiting a particular node. This layer is a class that is capable to provide functionality of sending messages to and receiving messages from directory

and resource layer of the visited node. This layer also provides additional functionality of query matching by collaborating with directory and resource layer for realising the task of finding resource(s) hosted on this node that is semantically similar to submitted query.

3.2. Specification of Agents

The proposed system - Affinity is hybrid system based on the semantic overlay network, unstructured P2P system, and MAS. All peers share their resources that are maintained by the set of collaborating agents on each peer. The collaborating agents on each peer are

1. *Interface Agent (IntA)*,
2. *Local Agent (LA)*,
3. *Information Agent (InfA)*, and
4. *Reconnaissance Agent (RA)*.

The purpose of each is as follows:

- *Interface Agent*: IntA is a static agent that provides user interaction to the system. The user interacts with IntA using the GUI interface that a.) shows search query, b.) informs search results, and c.) inform active RA(s).
- *Local Agent*: LA is another static agent that holds information i.e. keys for defining local resources and the corresponding location of resource on the peer. In addition, it has tasks to serve InfA for keywords request and RA for keyword similarity.
- *Information Agent*: InfA holds information about peers that are semantically similar to this peer. It holds a data structure that contains all peer's GUID, similarity value and keywords that it is sharing. It also serves RA upon request of next site where RA is required to migrate. InfA also communicates to LA to request a list of keywords that a peer is sharing that it in turn is submitted to bootstrap server for registration and finding peers that belong to same cluster.
- *Reconnaissance Agent*: RA is a mobile agent that is created by the IntA upon user's search request. RA migrates to new peers by requesting node address from InfA. RA's task is to migrate to peers and to investigate LA that is responsible for hosting resources (hence keywords) about their possible similarity to user's query and report it to IntA.

In addition to the proposed multi-agent system, the overlay network organisation of this P2P system is improved by InfA registration to bootstrap server. A *bootstrap* server maintains a list of peers that are currently in the system. Upon *registration/joining*, the bootstrap server replies with list of peers that are semantically similar to this peer. The cosine similarity between peers is actually keyword similarity of hosted resources of those peers. The result of this similarity is cluster effect as illustrated in Figure 2. Although, the sparsity of keyword matrix on the bootstrap server is large but still it is overlooked by potential advantage, that each peer is now organised in overlay network (i.e. it only knows the address of neighbours in a cluster). The globally unique identifier (GUID) of neighbours in cluster are used to prepare a hash table that is maintained by InfA. When a neighbour *disconnects* from overlay network, it informs bootstrap and its neighbours to remove its GUID from matrix and hash tables respectively. Upon *creation* of RA, it communicates to InfA to provide it with itinerary (next site) for migration. InfA uses the hash table provided it by bootstrap server to issue a peer GUID that host resources/keywords that are close to requested user query.

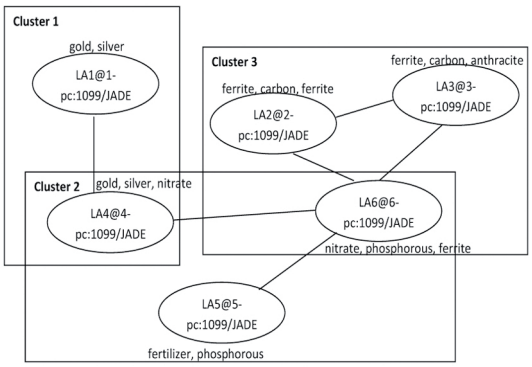

Fig. 2. Peer clustering and overlay organisation achieved using *latent semantic indexing*

3.3. The Proposed Mobile Agent Routing

Peer clustering is based on the conceptual content of resources shared by peers. The objective is to organise an overlay network in such a way that when given a query, small number of peers are selected based on "higher" chance of query hit. The benefit of this strategy is two fold. First in context of peer clustering - the peers to which RA migrates to will have many matches, so that the query is answered faster, and second in context of

RA routing - the peers with "lesser" chance of getting a query hit will be steered clear by the migrating RA, thus avoiding wasting resources on that query (and allowing other queries to be processed faster). Peer Clustering and Agent Routing are accomplished using LSI. The following Section 3.3.1 explains the state vector and singular vector decomposition (SVD) based semantics for peers and keywords hosted.

3.3.1. Latent Semantic Indexing and Singular Value Decomposition (LSI-SVD) for Peer Clustering and Mobile Agent Routing

Latent semantic indexing (LSI) is a variant of a vector space model where low rank approximation to the vector representation of the corpus is computed.[44] LSI considers that latent structures may exist in documents that may not be visible and may very well be hidden due to variability in word choice.[44] Singular value decomposition (SVD) of the corpus is calculated to estimate the structure of lexicon usage across the documents.

The nodes may be represented by number of keywords (lexicons) that it shares. Hence, a set of nodes can be represented by a matrix called *keyword-peer* matrix A. The elements of the *keyword-peer* matrix represent the frequency of each keyword f on a particular node. Let N be the number of peers in a P2P network, and K be number of distinct keywords (lexicons). It should be noted that N can be resources when observed from RA-LA point of view, but generically we assume it as number of peers. The feature matrix called *keyword-peer matrix* is constructed as $A\varepsilon\,[a_{ij}]_{KxN}$where a_{ij} =frequency of the keyword i on node j. $A_{ij} = 0$ if the peer j does not contain the keyword i. Not all keywords appear on all peers and hence matrix A is generally it is a sparse matrix. Now, matrix A denotes <peer, keyword> pairs in the network, which is the knowledge of correlations between peers and keywords. To properly characterise latent semantics and correlations between peers in LSI, that matrix A is factored into product of three matrices using SVD.[45]

$$A = USV^T \tag{1}$$

$U^TU = I_K$and $V^TV = I_N$, $I_K$and $I_N$are identity matrices of order K and N respectively. Matrix S is a diagonal matrix with elements $diag[\alpha_1, \alpha_2, \alpha_3, ...\alpha_{min\{K,N\}}]$, $\alpha_i > 0$ for $1 < i \leqslant d$, and $\alpha_j = 0$ for $j > d$, where d is the dimensionally reduced matrix. SVD is a low rank approximation of matrix A.[45] SVD is used to find the singular vectors corresponding to k largest singular values which dominate the original matrix. Peers and keywords can be characterised by linear combination of singular values i.e. a k-dimensional point in the feature space spanned by k singular vectors.[46] Deerwester et al in Ref. 47 shows the small dimensions are enough to express latent semantic i.e. $k \ll min\{K,M\}$. The resulting singular vector and singular value matrices are used to may keyword-based vectors for peers and queries into a subspace in which semantic relationships from the *keyword-peer* matrix are preserved while keyword usage variations are suppressed.[48] The reduced dimension decomposed matrix as a new pseudo-keyword-peer matrix is given by

$$A_k = U_kS_kV_k^T \tag{2}$$

where columns of U_k contains the eigenvectors of the $A_kA_k^T$ matrix or first k columns of matrix U and the rows of V_k^T are the eigenvectors of the $A_k^TA_k$ matrix or first k rows of matrix V^T. S_k is a diagonal matrix that has its diagonal elements with special kind of values of the original matrix.[47,45] These are termed the singular values of A_k that has first k largest singular values.

In SVD representation of original vector space, $A_k^TA_k$ is a NxN symmetric matrix for inner products between peer vectors, where each peer is represented by a vector of keyword weights. This matrix can be used for cluster analysis for collection of peers. Each column of matrix, $A_k^TA_k$ is a set of inner products between peer vectors in corresponding column of the matrix A, and every peer in the collection. The cosine similarity measure of peers i and j can be computed as follows:

$$sim(i,j) = \frac{\langle i,j\rangle}{|i||j|} \tag{3}$$

For information retrieval in K-dimensional space query Q is treated as another set of keywords and hence query Q becomes $q = Q^TU_KS_K^{-1}$that is compared to the peer represented by $p = p^TU_kS_K^{-1}$. These equations presents the coordinates of the vectors in the K-dimensional space and query-peer cosine similarity is given by

$$sim(q,p) = \frac{\langle q,p\rangle}{|q||p|} \tag{4}$$

All peers share the keywords that inform about the hosted resources. The similarity between the keywords

shared by various peers forms a cluster of peers that are similar to each other and thus forming a cluster and in turn an organised overlay network. This also increases the efficiency of discovering a resource as number of hops that RA has to take to find a resource are decreased. In order to get list of peers, another parameter - *minimum support* is passed by user. The significance of this parameter is give user a level of control over list of known peers by forming a "canopy" on known peers. The value of minimum support ranges between -1.0 to $+1.0$ where -1.0 explains ambiguity - list of all peers registered i.e. ignoring the similarity results, and $+1.0$ explains certainty - list of all peers that are exactly similar to this peer i.e. only peers that are sharing same keywords with same frequency.

RA's routing is directly affected by the minimum support value passed by user during acquisition of peer list i.e. lesser the value of minimum support, larger set of peer list and that means RA has larger number of ambiguous peers to choose from or vice versa. However, another value of *minimum support* for resource discovery and this time it means the similarity of query passed by user to the keywords shared by various peers in peer list allows RA to find the peer where it will migrate to.

Through experimentation it is realised the initial value for peer registration can be $+0.1$ or higher and for resource location $+0.5$ and higher can provide suitable results.

The unstructured network is created at random where to locate/search for particular resources, the message has to be forwarded to number of times. If this is limited by N hops, where N is the number of nodes within the query message's reach, then query routing complexity on an unstructured P2P network is of the order of N, or $\mathscr{O}(N)$. On structured networks, or the MAS that have underlying overlay network based on structured overlay the query routing complexity is typically $\mathscr{O}(\log(N))$, where N is the number of network nodes. In our case, suppose N is the number of nodes and m is the *minimum support* of a node that ranges as $0.0 \leqslant m \leqslant 1.0$, and N_D is the maximum number of nodes that are semantically close where $N_D \ll N$, then the complexity of query routing is given as $\mathscr{O}(N_D)$, when $m = 0.0$ and $\mathscr{O}(N_D^{-\log(m)})$, when $0.0 < m \leqslant 1.0$. As *minimum support* increases the number of nodes required to be visited by migrating RA decrease logarithmically, and when *minimum support* is 0.0, it means the RA has to visit all nodes N_D in this particular domain. It is seen that the query routing complexity for resource location is much more effective is our system as compared to structured and unstructured system because of informed migrations performed by the RA. The results are later justified in experimentation in Section 4.

3.4. The Proposed Multi-Agent Collaboration for Resource Discovery

The system starts by starting up a bootstrap server. The LA locates all the resources that are shared by the peer and preparing the keyword list that defines the resource. The InfA requests the LA to inform it about the keyword list that in turn is used by the InfA to register the peer on bootstrap server. This behaviour is a cyclic behaviour of InfA that is scheduled every $300,000ms$. Upon registration, based on *minimum support* value, the InfA receives the peer list containing list of peers, their similarity value and keywords shared by those peers. User's request for resource location to the IntA is attributed by list of keywords that form a query, *minimum support* value for acceptable results, and number of hops that the RA can make. The detailed interactions between the collaborating agents are shown in Figure 3.

The resource discovery is carried out using following algorithm:

1. When query is passed by user to the IntA, the IntA in turn creates the RA for that specific query.

2. The RA is informed about query, minimum support, and number of hops by IntA.

3. The RA requests the InfA for peer name where it should migrate to. The peer name is informed by the InfA to the RA by looking up the peer name in hash table based on the semantic similarity of the query and the keywords shared by that peer.

4. The RA requests the agent management service (AMS) to find the container/platform where the selected peer is located .

5. The RA migrates to that peer and increments the number of hops by one.

6. The RA requests the LA of this peer to inform it about the resource name whose keywords are

semantically similar to the query and higher than minimum support given by user. The LA provides the resource name to the RA.

7. The RA informs the IntA about located resource i.e. GUID of the peer where resource is located, resource name, cosine similarity value.

8. If number of hops made by the RA is less than maximum number of hops allowed by user then go to Step 9 else go to Step 10.

9. The RA requests the InfA of this peer for a new peer name where it should migrate to (hops to previously visited peers and to creator peer are not allowed). Go to Step 3.

10. As number of hops made by the RA are equal to maximum number of hops allowed, the RA terminates itself.

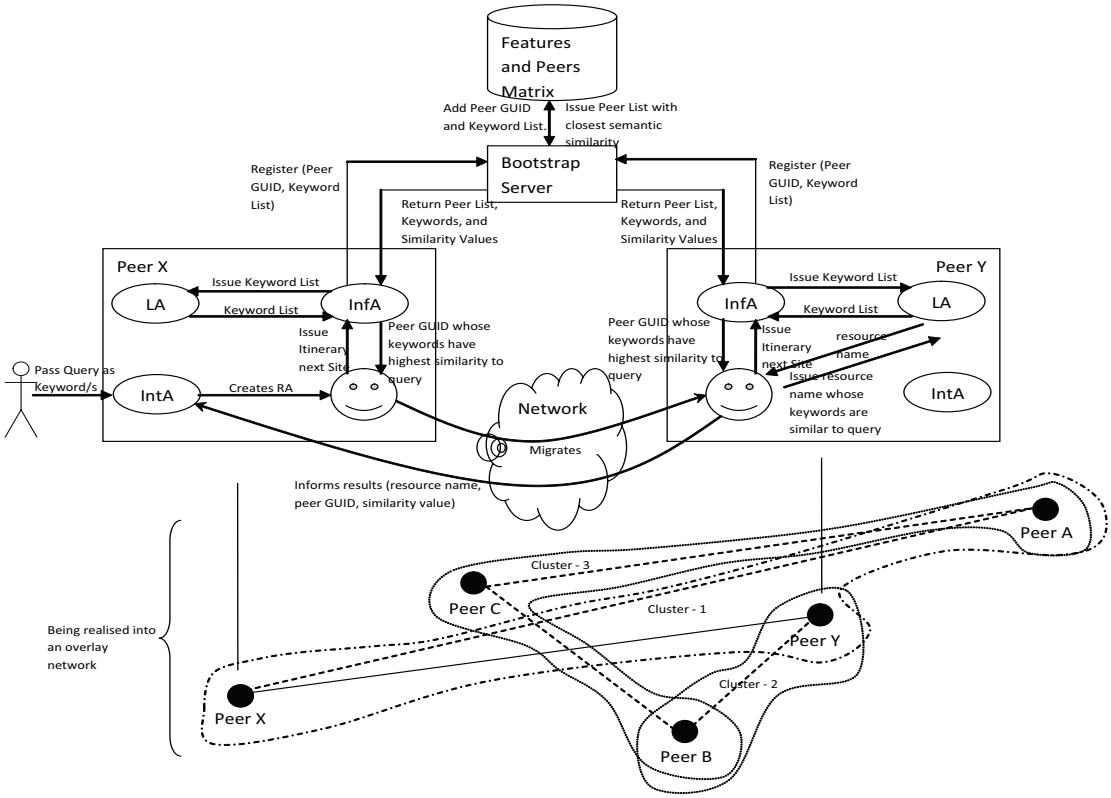

Fig. 3. Interactions between multiple agents for resource discovery and realisation of an overlay network

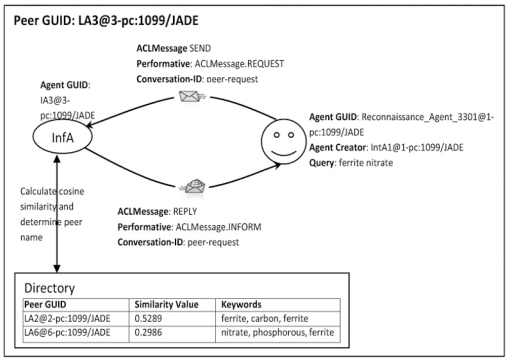

Fig. 4. The RA's interaction with InfA for issuing new peer GUID

Step 3 is shown in detail in Figure 4. The RA requests the InfA for GUID of the peer that it should migrate to; to find the resource. The InfA refers to the directory and calculates cosine similarity value based on degree of match between the query and list of keywords available. The highest similarity value is used to determine the peer GUID by looking up in the directory. Finally, the GUID of selected peer is informed to the RA. The RA now uses the GUID to find the container name from AMS where the corresponding peer GUID resides. This mechanism is better then "blind" or flooding technique as

in this case the RA migrates with certain knowledge i.e. where and why to migrate to a certain peer as opposed to flooding the overlay network with communication messages or with multiple clones of the RA. Essentially, it improves the routing of the RA. The behaviour of the RA has been defined by *beforeMove* and *afterMove* methods. afterMove method is invoked just after migration to increment the number of hops made by the RA followed by checking the termination conditions.

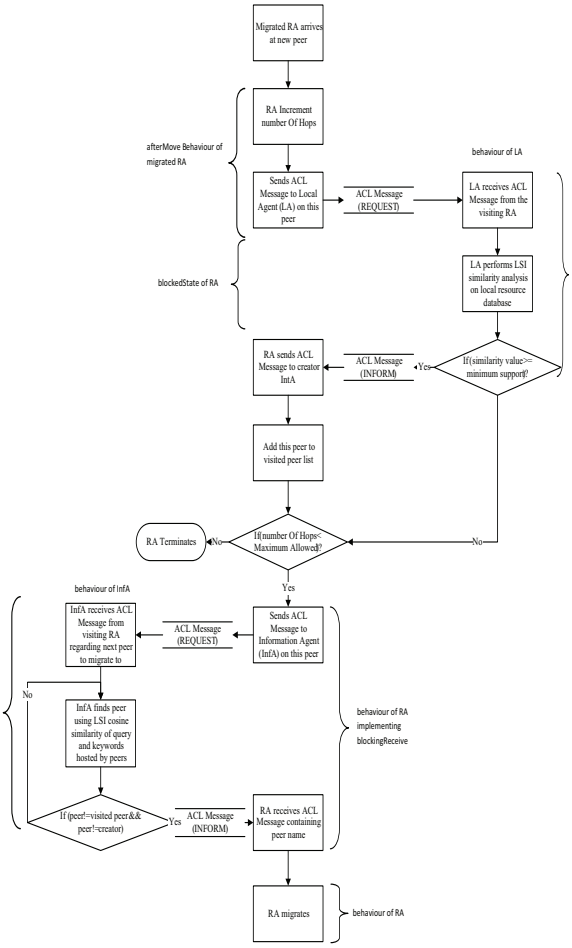

Fig. 5. Flow diagram for behaviour of the InfA and the LA upon arrival of the RA

Figure 5 presents the flow diagram for behaviour of the InfA and the LA when the RA arrives at a certain peer. Shown in the flow diagram are behaviours of three agents the RA, the LA, and the InfA. In addition to behaviours of agents, *blockedState* of the RA and *blockingReceive* of the RA is observed. These methods are invoked based on the ACLMessages in the mailbox of each agent. Essentially, as long as the RA has not received any message that matches the *MessageTemplate* , the RA is in blocked state.

3.5. Implementation

We have used Java RMI and JADE to implement our multi agent resource discovery using mobile agent system. Jade's agent management environment is used for creating multiple containers emulating distributed environment where peers are active. Java Remote Interface has been used for defining and implementing the bootstrap server.

All the agents are developed using FIPA complaint agent framework - JADE. Instead of using RMI or socket based communication between various agents including the mobile agent (RA), agent communication language (ACL) has been used for communication particularly using performative (REQUEST, INFORM, and CLP (Call_For_Proposal)). In addition as the RA is a mobile agent, it is further required to register FIPA standard FIPA_SL0 (slCodec) content language.

Agents do not invoke methods on other agents and communicate through ACLMessages. Hence to handle messages from various agents and/or various kinds of messages we have implemented the use of *MessageTemplate*. A receive method takes a message template as a parameter and only returns messages matching that template. This is an important feature that is implemented for successful multi-agent communication system. The behaviour implemented by the LA includes case:

1. where the RA communicates to the LA to locate resource name whose keywords are semantically similar to user's query

2. where the LA informs the RA about selected peer GUID using ACL.

3.5.1. Agents Communication Implementation

The multi agent system has been designed to receive search requests from the users through the IntA. IntA class has a graphical user interface associated with it that takes input parameters - keywords for the query (search terms for a resource). The *minimum support* and the time to live (*number of hops*) parameters have been defaulted in the experimental setup to be 0.00 and 3 hops respectively. Upon invoking the search, the RA is created by the

IntA in the method *onGUIEvent()* that has an identifier - (GUID) and the minimum support and number of hops as the parameters.

In addition to creation of the RA, the IntA is also responsible for displaying results sent by the RA throughout its life cycle. As mentioned before in Section 3.5, the FIPA specification implemented by JADE does not allow agents to communicate to each other using method invocation or more specifically in this case remote method invocation, the IntA hence offers functionality for receiving messages from the RA through ACL implemented in the inner class *ReceiveMessageRecon*. This inner class extends the *CyclicBehaviour*, that creates instance of *MessageTemplate* for only receiving messages sent by instances of the RA created by this instance of IntA using *MatchCoversationId("results")* and *MatchPerformative(ACLMessage.INFORM)*. The IntA also has an inner class *ReceiveTerminationRecon* that extends *SimpleBehaviour* for receiving termination message from the RA when RA has reached end of its life cycle or if a matching resource has been discovered.

The RA is responsible for discovering the resource on other nodes by migrating to those nodes. The node that is most likely to host the resource is provided by the InfA that holds directory of nodes for routing the RA on the overlay network. Again, the communication between the InfA and the RA is using ACL and the *MessageTemplate* uses *MatchPerformative(ACLMessage.REQUEST)*.

Upon migration the RA communicates to the LA for a matching resource. All the results obtained are communicated back to the IntA through the *MessageTemplate* described above.

4. Experiments, Results and Evaluation

The experiments were conducted to evaluate the effectiveness of our method for resource discovery using mobile agents. The experiment is bifurcated into two parts. Part1 investigates to find out the response time (in secs) that it takes to locate a resource (multiple keywords based query) on an overlay network using RA as compared to flooding and Part 2 investigates the benefit of informed search based on LSI as opposed to flooding and routing algorithm inspired by *AntHill* as in Ref. 41 and Ref. 42 and structured P2P systems by Dasgupta et al and Kambayashi et al by finding out number of messages that are on an overlay network.

4.1. Design of Experiments

The design of experiments has been setup in order to compare the proposed technique for content-based resource discovery in terms of heuristic search and search performance. The benchmarks are provided by flooding technique and by term-matching, Jaccard coefficient techniques as described in Ref. 3, Ref. 34, Ref. 23, Ref. 49 and Ref. 40 . Flooding technique was used as benchmark; as it is widely accepted technique and has been used as backbone for purpose of routing and searching in number of resource discovery techniques including the contemporary techniques as proposed by Dasgupta et al.[39,49] Furthermore, as Dasgupta et al. in Ref. 39 , is using this technique for routing in context to MAS, it becomes all the more important to prove the effectiveness in terms of routing and searching of proposed technique in this context. More contemporary researches from Zhu et al. and Kambayashi et al. have proposed the usage of semantics links based on term-based matching or Jaccard coefficient for resource discovery. Kambayashi et al. uses mobile agent to traverse through overlay network and their technique of preference for matching resources is logical similarity based in Jaccard coefficient.[40] Kambayashi et al. in Ref. 40 further uses DHT based structured overlay for migration of mobile agent. Similar approaches has been used in different flavour however (for instance, using DHT for locating nodes, using flooding for routing mobile agent or using term matching for locating relevant results) have been used by many contemporary research works. As Kambayashi et al. is using number of techniques in their approach, the author believes comparing results of proposed method to their technique would provide comparison and evaluation on high degree of intersection of attributes and techniques and a good benchmark. The experiments conducted measure the performance of the proposed method on the basis of following parameters:

- the response time test
- the effectiveness of search technique
- relevance of results
- degree of similarity

4.1.1. Experiment Environment and Test Bed

For comparison to flooding technique as employed by Gnutella in Ref. 3 and Ref. 4, the experimental setup

used the open source Java API, JTellav0.7.[50] This API can be used to create a P2P overlay network. The setup included 4 peers where 3 peers hosted resources. The setup included 4 peers where 3 peers hosted resources and fourth peer is used for searching resources. Details of each peer including hardware specifications, operating system, IP addresses, number of resources and types of resources is shown in Table 4.

Peer Name	Peer 1	Peer 2	Peer 3
Operating System	Microsoft Windows Vista Home Premium	Microsoft Windows 7 Home Premium	Microsoft Windows XP Professional Service Pack 2
Processor	AMD Athlon Dual Core QL-62 2.00 GHz	Celeron (R) Dual Core CPU T3000 @ 1.80 GHz	Intel Pentium 4 @ 2.50 GHz
RAM	3 GB	3 GB	512 MB
IP Address	192.168.1.2	192.168.1.5	192.168.1.7
Number of Resources Shared	13	8	8
Type of Resources	8 pdf files 3 docx files 2 doc files	6 pdf files 2 rar files	8 pdf files

Table 4. Gnutella flooding peers test bed

As seen in Table 5 total number of nodes participating in Affinity were 10. For the purpose of consistency with benchmark, 4 computers participated in this experiment. In this setup, computer 1 hosted Bootstrap server and 3 computers participated in P2P overlay network. Between these 3 computers, 10 nodes were created, where computer 1 hosted 4 containers hence 4 nodes, computer 2 hosted 3 containers hence 3 nodes and finally computer 3 hosted 3 containers hence 3 nodes. Each container simulated as different node participating in P2P overlay network. The hardware specification of machines is as provided in Table 4. Further specifications regarding MAS and keywords for resources shared are shown in Table 5 and Table 6.

Specification	Value
Total Number of Computers	4
Bootstrap Server	1
Computers Participating	3
Number of Nodes Participating	10
Maximum Number of Hops	2
Total Number of Shared Resources	27
Minimum Support	0.0

Table 5. MAS test bed

Local Agent	Keywords Shared
LA1	sun moon earth mars mercury venus
LA2	moon pluto
LA3	sun stars one two
LA4	one two three four five six mars
LA5	moon
LA6	jupiter saturn neptune pluto
LA7	moon saturn pluto
LA8	two neptune
LA9	pluto earth one
LA10	one sun two moon

Table 6. Keywords used for sharing resource on each node

The objective in test 2 is to compare the effectiveness of indexing technique, relevance of results and degree of similarity. The experiments in test 2 used a MEDLINE data set that consisted of 1033 documents from Cornell University.[51] After removing of stopwords and filtering of nouns, verbs, adjectives, and adverbs, 5735 indexing terms (lexicons) were found. The details of data set can be found on media disc. This data set was used specifically as all resources have already be categorised and attributed with features such as relevance and similarity. The objective was to find out of the proposed technique provides similar results and then to compare the results with techniques used by other research works. Hence, the prepared results served as benchmark for comparing the effectiveness of proposed technique to other relevant works.

4.2. Test 1 - Comparison to Flooding Technique

4.2.1. Experiment 1 - Response Time and Evaluation

The objective of experiment 1 was to calculate the response time for query on an overlay network. Once the response time is available it can be concluded that which method is more effective with respect to amount of time it takes to locate a resource on an overlay network. It is observed from the bell curve that amount of time it takes

to find a particular resource in our method is consistent and ranges between $5s$ to $6s$. Flooding, however does not have any consistency in response time. It is observed from bell curve shown in Figure 6 that response time can vary from few seconds to few minutes. Furthermore, it is observed that in flooding 14 queries out of potential 28 queries has response time of $< 5s$ which approximates to 42% of total number of queries, where when using our method we observed that 67% of queries were replied with resource location in $< 5s$ and 23% of queries replied in $< 6s$.

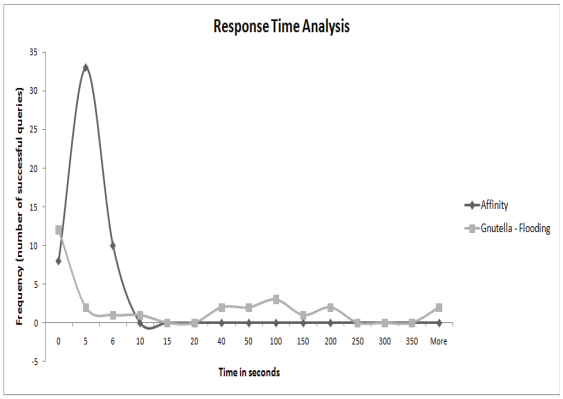

Fig. 6. Frequency distribution of response time analysis - Gnutella vs. Affinity

It is evaluated that overall response time, when using the proposed method is lesser than the case of Gnutella using flooding technique. But it should be noted that a lower response time does not measure the effectiveness of search technique in terms of successful results as described further in Section 4.2.2. The author concludes that lower response time is attributed to mainly two reasons. Firstly, as Gnutella is pure P2P network, it is required of participating peers to communicate their status using PING and PONG messages on the overlay network. This results in high amount of traffic on overlay network and results in saturation. It is observed, as mentioned in Section 4.2.3, that PING and PONG activity together amount to 97.5% of messages. This results in latency and hence low response time. Secondly, as resources to be located are searched based in multiple keywords do not always match the file name of resource to be located, it amount to low response time as query may not match the resource completely.

4.2.2. *Experiment 2 - Effectiveness of Search Technique and Evaluation*

In experiment 2, the objective was to investigate the effectiveness of search using our method as compared to flooding. For achieving this objective the experiment setup was to compare successful queries to unsuccessful queries. It was realised through experiment using our method that out of 30 queries, 24 responded with query hit, 4 queries did not have any response, and 2 queries replied as NaN network (See Figure 7).

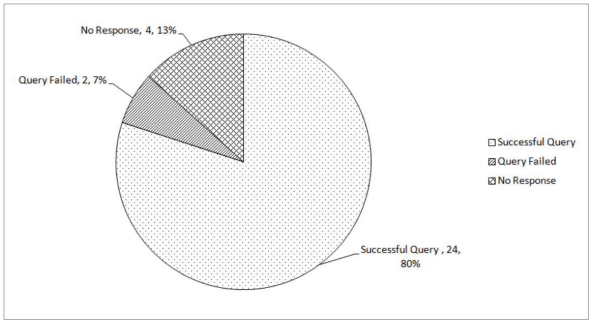

Fig. 7. Query successful vs. unsuccessful - Affinity method

Furthermore, it was realised that NaN is due to explicit specification of *minimum support* parameter as 0.0. The nodes in similarity with 0.0 did not host the content that was required by user. In case, of flooding, 26 queries were passed through various nodes. 32% of queries had query hit and 68% of queries failed (See Figure 8).

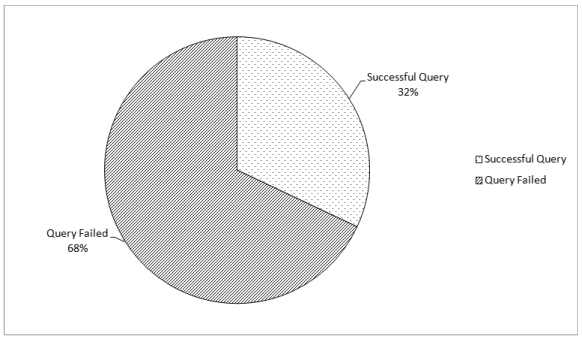

Fig. 8. Query successful vs. unsuccessful - Flooding method

4.2.3. *Observations*

Ref. 39 and Ref. 40 has confirmed that no matter how many peers or resources are there on an overlay network, the flooding technique generates a consistent number of messages on an overlay. We have observed in flooding that the amount of traffic or messages on an overlay network or even response time increase or decrease is at-

tributed to mainly the PING and PONG activity. This continuous stream of messages is produced by the peers to check existence and current status of other peers. We used Wireshark to monitor the Gnutella packets.[52] A total of 14001 Gnutella packets were analysed when overlay network was subjected to 2 queries. It is clear from the pie chart (Figure 9) that 84.7% of traffic is related to PONG descriptors, 12.8% to PING, 2.22% to QUERY, and 0.07% to re-transmission errors. Gnutella connections are relatively unstable which lead the nodes in iterative effort for discovering other nodes on overlay network as opposed to nodes joining and leaving network autonomously.

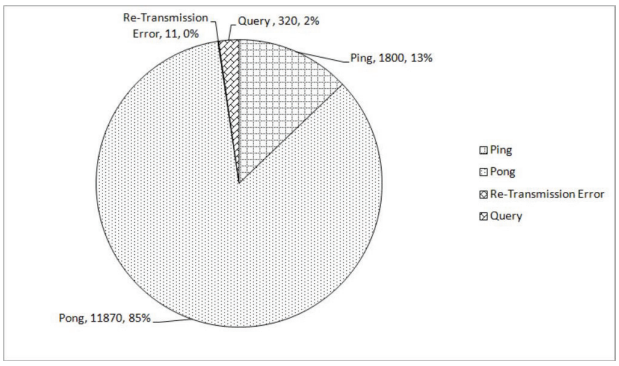

Fig. 9. Division of packets for Gnutella

It is also observed from the graph in Ref. 40 that no matter what is the number of resources shared, as long as number of peers is constant the number of messages (bytes) will stay constant.

4.2.4. *Critical Analysis*

However, this raises another issue of why there is a decrease is number of messages also claimed by Dasgupta et al and Kambayashi et al.[39,40] We observed and evaluated that the decrease in number of messages in these multi-agent systems is due to decrease in number of hops to locate a resource. In Ref. 40 Kambayshi et al, claims that number of messages on overlay network will decrease with increase in number of resources. This is because the overlay network has become more resourceful and hence almost all peers have links to other peers, which means that when the SA enquires from directory services on NA about peer to migrate to, it is capable of informing SA about the highest possible logical distance value because of its resourcefulness. This is observed in proposed method too and the author agrees with Ref. 39 and Ref. 40 . It is evaluated in Section 4.2.1, that number of inter cluster links are on average higher than case where, logical distance value was used to create semantic links between nodes. More number of links makes the overlay network more resourceful thus reducing number of hops and reducing number of messages on network. Furthermore, it is evaluated through precision-recall results where the author defines precision as the ratio of number of relevant resources/nodes found during search to number of search results and recall as the ratio of relevant resources/nodes found to total number of relevant resources/nodes in corpus. Though, number of inter-cluster links are higher that may result in compromise of precision, however, we achieve higher recall making degree of relevance higher (See Figure 10) due to efficient indexing technique. Together, with results from reduced response time, higher recall and greater number of successful queries it can be concluded, that lesser number of messages exist on network.

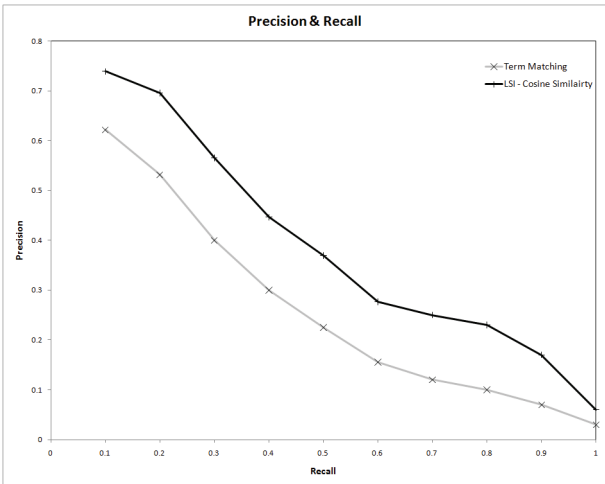

Fig. 10. Precision and recall results comparing LSI to TF-IDF indexing model

In test 2 - Section 4.3, the author investigates the effectiveness of their techniques/algorithms to reduce number of messages and compare them to proposed method.

4.3. *Test 2 - Comparison to Other Routing Techniques/Algorithms*

The aim of this test is to investigate the effectiveness of the routing mechanism employed by Ref. 40 that cal-

culates the logical distance between the nodes based on the resources shared by that node as compared to LSI based clustering of nodes and routing based on calculation of cosine similarity between search query and the lexicons shared by nodes. This experiment also indirectly studies the effect on amount of message on overlay network. Replicating exact environment as used by Ref. 40, has been a tedious process as they are using Overlay Weaver and Agent Space both tools developed by them and changed to accommodate messaging between agents through Overlay Weaver.

Though, the author evaluated their results and observed that the decrease in number of messages is due to increased similarity score (Jaccard Similarity) between shared lexicons. In this context, the author designed another experiment that would compare their indexing and routing algorithm to the proposed method by comparing its effectiveness on third party data provided by Cornell University.[51] The effectiveness was evaluated in different experiments.

4.3.1. Experiment 1 - Pair-Wise Document Similarity And Evaluation

In experiment 1, pair-wise document similarity is investigated by comparing Jaccard similarity (subset used by Ref. 40) and Cosine similarity (used by proposed method). In case of Kambayashi et al Ref. 40, the test required normalising the term-document matrix using term-frequency and inverse document frequency indexing (TF-IDF) for measuring Jaccard similarity. In our case, the test required creating the normalised latent-semantic indexed matrix for measuring Cosine similarity as described in Section 3.3.1. The effectiveness in experiment 1 is studied by finding out number of documents that match where the minimum threshold is > 0.1. The result of number of document will indicate the resourcefulness of overlay network as that is used to cluster the nodes. In other words, more is the number of matched documents, larger is the cluster, and more are the chance for mobile agent to locate a resource which would mean lesser number of migrations for mobile agent and hence less number of messages on overlay network. We evaluated from the following graph (Figure 11) that using LSI and cosine similarity clearly has larger number of pair-wise matches between documents and hence provide larger cluster and links between clusters.

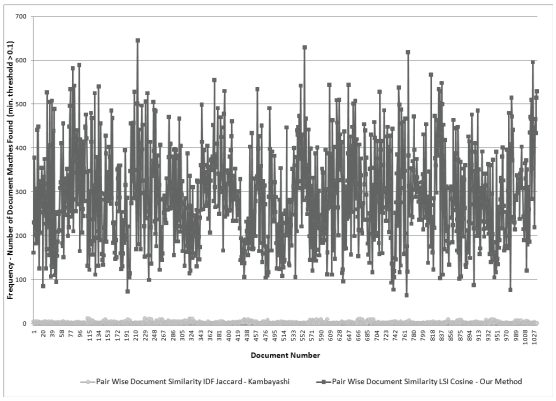

Fig. 11. Pair-wise document similarity TF-IDF Jaccard vs. LSI Cosine

It is evaluated that larger is a set of similar documents, more resourceful is the overlay network, hence lesser number of hops are require by RA to locate a resource. The pair-wise documents similarity is large in case LSI technique used in our method, hence number of messages required by RA to locate a resource will be lesser and in this case much lesser than flooding as observed in Ref. 2 and Ref. 3 and logical distance method observed in Ref. 39 and Ref. 40 making our method for routing RA through overlay network more efficient in terms of time and bandwidth consumption.

4.3.2. Experiment 2 - Effectiveness of Search Technique And Evaluation

In experiment 2, the aim was to investigate number of documents that found to be similar in to search query. Larger number of documents effectively indicate:

1. large number of nodes for the RA to migrate to for locating resources

2. better inter-cluster link for routing the RA through overlay network.

It is highly important that mobile agent can traverse through overlay network for locating the resource.

If routing links cannot be established between clusters - it would indicate:

1. mobile agent cannot locate a resource because of its incapability to migrate to different clusters or

2. mobile agent will provide results that are less precise.

It must be noted that larger number of matches also mean large number of nodes to be visited by the RA hence more number of message on overlay network which in effect means higher bandwidth consumption. This however is controlled in our case by introduction of factor called *minimum support* as mentioned in Section 3.3.1, that is set by user to reduce the number of selected nodes for the RA to visit. We conducted similarity test on corpus of 1033 documents by subjecting them to 30 different queries.[51] The following graph (Figure 12) was obtained as a result of this experiment, informing number times a matched documents is found for 30 queries where *minimum support* is > 0.002.

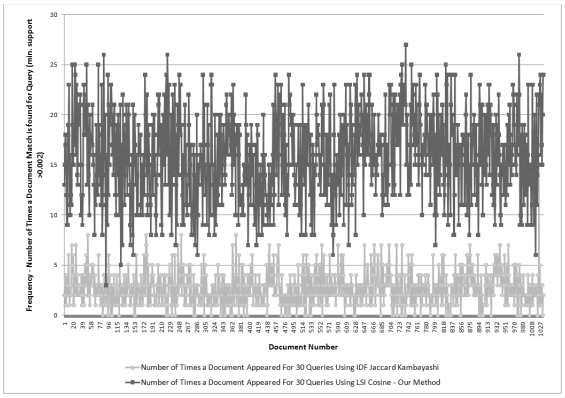

Fig. 12. Number of times a document appears for 30 queries Jaccard similarity vs. Cosine similarity

It is observed that our method is resulting in larger inter cluster links and also large number of nodes where the RA can potentially visit as compared to logical distance method used by Kambayashi et al.[40]

4.3.3. *Experiment 3 - Effectiveness to Locate Resources and Evaluation*

In experiment 3, the aim was to investigate effectiveness of our method to locate the resource. Keeping that in context, in general terms it means - number documents found per query using our method as compared to the logical distance method. Similar to experiment 2, for achieving the aims of this test, the document corpus was subjected to 30 queries and number of documents found per query were obtained for minimum support > 0. This number was compared for LSI based Cosine similarity and TF-IDF based Jaccard similarity. The graph (Figure 13) shows that number of documents using LSI Cosine method used in our method is higher than TF-IDF Jaccard method. It is further evaluated, that a larger number of documents associated with a query means 1. higher cluster links 2. larger set of relevant documents found as part of resource discovery. Of course, as mentioned in experiment 2, larger set of documents can also indicate irrelevant information, but this can be capped using parameter *minimum support* as mentioned in Section 3.3.1.

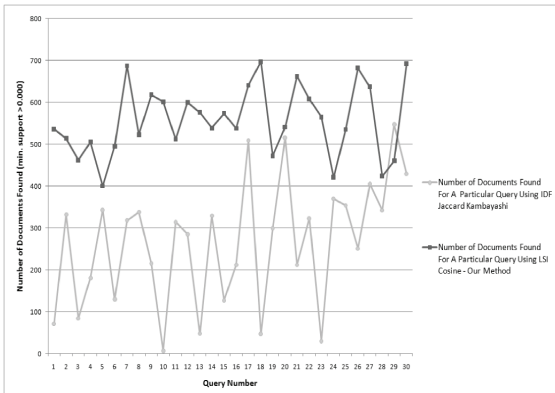

Fig. 13. Number of documents found for 30 separate queries on corpus of documents

4.3.4. *Experiment 4 - Degree of Relevance of Results and Evaluation*

In experiment 4, the aim was to investigate the degree of relevance of results obtained during search process by the RA. Again, similar to experiment 2, the corpus was subjected to 30 queries to find out about similarity scores. The highest similarity score obtained is assumed as resource that is best match to a given query. The objective was to collate the highest similarity scores and find their frequency distribution. This process would achieve

1. offer insights into relevance of results
2. inform which method is capable of extracting best match documents.

Perhaps, if the same document is found a result of search, using both methods, if logical distance is low, it may safely be assumed that mobile agent may take more time or even more number of hops to reach the node.

It is observed from the graph (Figure 14), that using our method the similarity scores tend to be on higher end of frequency distribution as opposed to other research works. This indicates that it is of utmost importance the similarity scores are high which would effectively mean less messages on overlay network and better response time.

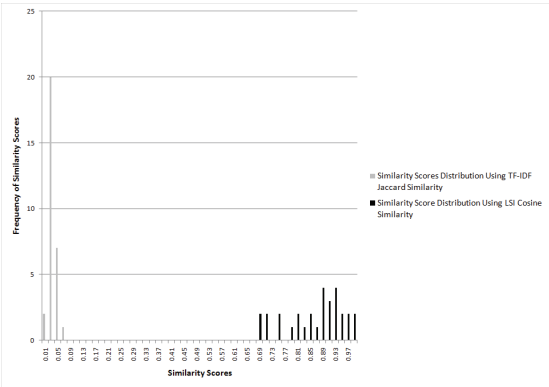

Fig. 14. Similarity score distribution TF-IDF Jaccard vs. LSI Cosine

5. Discussions - Related Work

The works done by Zhu et al ESS Ref. 23, Dasgupta et al Ref. 39 and Ref. 49, Kambayashi et al Ref. 40, and Crespo et al Ref. 34 are related to our work for development of P2P system for resource discovery and the first sections of chapter provides related discussions.

5.1. Analysis of Research Works from Crespo et al.

Crespo initially in Ref. 18 presented the idea of routing indices for controlling the amount of flooding and saturation of overlay network. The concept however suffered from maintenance of distributed-index on various nodes that itself generated it own large amount of traffic.

Later in Ref. 34, Crespo et al. introduced the idea of semantic overlay networks (though not in a P2P context) where the nodes can be clustered to form an overlay network. Crespo use explicit term semantics to building routing indices. They assign documents with terms indicating related realms, and maintain in each peer a statistic table containing term-based routing indices, which indicates how many documents would be found, if probes the query of that term to a neighbour peer.[46]

We understand that Crespo et al brought improvement to searching but as most latent semantics analysis proved, only terms-based statistics cannot fully capture resource characteristics as terms also have underlying correlations and semantics.[47,46] We have been inspired from the idea to form relationship between nodes but in our system we use these relationships for coordination of resources that are managed by nodes and further use it for informed routing of the RA.

5.2. Analysis of Research Works from Zhu et al.

Zhu *et al.* in Ref. 23 presented the use of information retrieval from unstructured and structured P2P system by use of semantic links between the nodes. The query flooding on P2P network is controlled using routing based on Jaccard similarity technique. However, as described in our tests the results obtained from normalised LSI based cosine similarity technique are far superior on terms of number of document matches and higher similarity scores. Furthermore, their system is not a mobile agent based resource discovery system which as mentioned in literature greatly improves upon the classical unstructured and structured P2P system. Our work contributes towards the dynamic organisation overlay network based on resources published by nodes. The relationship between nodes and resources for guidance of agent (direction) on overlay network is central and crucial.

5.3. Analysis of Research Works from Babaoglu et al. *and Dasgupta* et al.

Dasgupta *et al.* in Ref. 39 and Ref. 49 research work is greatly inspired from Babaoglu et al work on Anthill in Ref. 41 and Ref. 42. We here present analysis of Dasgupta's research work as they have used MAS.

Dasgupta *et al.* in Ref. 39 and Ref. 49 introduced the used of mobile agents for P2P resource discovery. Their system is based on referrals made by search agents. Clearly, in their system the behaviour of search agents evolve and gets better based on the trails established by searches done before. In contrast to our work, they do not use the routing tables for guiding the search agent through the overlay network as done in our work using directory facility made during initial registration of peer on bootstrap server. Furthermore, they did not introduce the use of peer-keyword semantics to form clusters of semantically similar peers. Clearly, they are using the classical technique of flooding to discovery resources that improves over time based on the search trails left by previous searches.

5.4. Analysis of Research Works from Dasgupta et al. *and Kambayashi* et al.

Kambayashi *et al.* in Ref. 40 has provided method of resource discovery by using mobile agent and DHT. Like

our work, there work also overcomes use of flooding for finding resources using node management table on each node (similar to directory service on InfA). However, the node management table is constructed by calculating logical similarity of keywords on peers based on primitive form of Jaccard similarity function as opposed to using latent semantics of keywords and finding cosine similarity in our case. Inspired from Crespo *et al.* in Ref. 34 , Kambayashi *et al.* also used the terms to capture the realm of resources shared. However, as mentioned before, matching only terms to cannot capture resource characteristics which is where we introduced the idea of using latent semantic analysis. In our case, we have introduced the use of minimum support for peer discovery and latent semantic indexing between peers to direct the RA towards resource.

Inspiring from Dasgupta's works in Ref. 39 and Ref. 49 and Babaoglu *et al.* works in Ref. 41 and Ref. 42, Kambayashi *et al.* in Ref. 40 introduced the use of pheromone value (AntHill) (that is calculated taking parameters such as number of resources shared by peer and clustering value (logical distance between peers). This feature is expected to guide search agents towards nodes with high correlation by reducing free-riders. We believe both, the techniques are equally credible, however the work from Kambayashi et al in Ref. 40 is discriminating free-rider which may hold a resource that is useful. Our aim in our work has been to create harmony between nodes and relevance of resources to user's query. We believe that if resource is available it should be locatable. Finally, they also used DHT - Chord structured P2P system for resource discovery, which we believe is interesting but opposes the original aim that DHTs cannot handle queries that are multi keyword or text based and is also only viable when keyword for finding resource is known exactly.

Kambayashi *et al.* in Ref. 40 technique i.e. guiding search agents using pheromone values and DHT for resource discovery may be leaving "ill-effect". In former case, the credible peer by removing a free-riders from list of peers that may be holding a resource and in latter case to direct the search agent towards exactly known resource keyword. They are undermining the level of ambiguity and introducing too much certainty into searches which is not the case in our system, where user can increase or decrease the search ambiguity/certainty by changing value of *minimum support* thus providing bigger/smaller "canopy" for movement of RAs.

6. Conclusions and Future Work

We have proposed a novel resource discovery system that uses mobile agent (RA) for discovery resources on an overlay network that is realised based on semantic similarity of keywords that are shared by peers. We further proposed a flexible multi-agent based approach to P2P network organisation that is based on the similarity of content shared by peers. We claim that the use of semantic similarity between content shared by peers i.e. clustering effect can be effective technique to route the RA to peers that host content that is similar to a user query and finally, that LSI based resource search by RA to find resources hosted by peers that are best match for a user query (where the user query can be text based or an approximate query) is very effective whether the query is contains text that is certain or ambiguous.

We further demonstrated that our approach for resource discovery is better than flooding and further more that an informed search technique used to guide RAs on a overlay network is better than controlled flooding. The results have demonstrated that the using flooding increases the quantity of messages on a network and it can be reduced by use of our technique.

In our previous experiments in Ref. 9 , we used flooding technique to find resource on the network i.e. the RA migrated from one peer to another in hope of finding the resource. The author has realised the shortcoming of last technique and introduced the use of guidance directory on each peer for providing the RA with better chance of finding a resource.

We also realise that in current technique only the resource name, the peer GUID, and the similarity value are results that are sent by the RA to IntA. We have not introduced the technique for downloading the resource to the peer which can be easily accomodated. Finally, we also realise that initially as resources are scarce, some clusters may not overlap, resulting into cases where resource cannot be located, but we do understand that as the peers become more resourceful, the clusters will start overlapping to higher degree, hence resulting into better search results.

As mentioned before in Section 3.3.1, *keyword-peer* and *keyword-resource* matrices can be large sparse ma-

trices. Holding these large matrices consumes memory which is not always abundant on systems that are continuously publishing or are dynamic. Dimensionality reduction used in proposed work offers a solution to some extent i.e. reduction in matrix size of an order of around 40%, but that can still be a large matrix. Some research works have been done in this field but are out of scope for this work. Further work can be done in this project to accommodate for this characteristic. We beleive that system architecture presented is generic and can be further refined in order to support distributed LSI as discussed in Ref. 53, where by the indexing could be decentralised and global search can be conducted for relevant resources on pure P2P overlay network. The problem to generate globally-consistent LSI structure is very challenging as the number of nodes presented by their content is large, dynamic and distributed.

7. References

1. Napster, "Napster," 2003.
2. K. Aberer, M. Punceva, M. Hauswirth, and R. Schmidt, "Peer-to-peer systems," in *In Practical Handbook of Internet Computing*, CRC press, 2004.
3. Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making gnutella-like p2p systems scalable," 2003.
4. G. D. Forum, "Gnutella protocol specification v0.4," 2002.
5. S. Ratnasamy, P. Francis, S. Shenker, R. Karp, and M. Handley, "A scalable content-addressable network," in *In Proceedings of ACM SIGCOMM*, pp. 161–172, 2001.
6. A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," 2001.
7. I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for internet applications," in *ACM SIGCOMM*, pp. 149–160, 2001.
8. B. Y. Zhao, J. Kubiatowicz, A. D. Joseph, B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," tech. rep., 2001.
9. M. Singh, X. Cheng, and X. He, *Multimedia Resource Discovery using Mobile Agent*. New York,: IGI Global, 2009.
10. F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE*. Wiley, 2007.
11. M. Karnstedt, K. Hose, and K. uwe Sattler, "Query routing and processing in schema-based p2p systems," in *In Proceedings of DEXA workshops*, pp. 544–548, IEEE Computer Society, 2004.
12. B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 41–53, 2004.
13. J. Killmeyer, *Information security architecture : an integrated approach to security in the organization*. 2nd ed. ed., 2006.
14. Y. Zhu and Y. Hu, "Efficient, proximity-aware load balancing for dht-based p2p systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 4, pp. 349–361, 2005.
15. T. C. Project, "Chord faq," 2010.
16. I. Guvnec and J. J. Urdaneta, "Peer-to-peer file sharing: A survey," 2010.
17. Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in *Proceedings of the 16th annual ACM International Conference on supercomputing (ICS) 2002*, 2002.
18. A. Crespo and H. Garcia-Molina, "Routing indices for peer-to-peer systems," *Distributed Computing Systems, International Conference on*, vol. 0, p. 23, 2002.
19. M. Bawa, G. S. Manku, and P. Raghavan, "Sets: search enhanced by topic segmentation," in *SIGIR*, pp. 306–313, 2003.
20. Y. Zhu and Y. Hu, *Handbook of Theoretical and Algorithimic Aspects of Ad Hoc, Sensor, and Peer-to-Peer Networks*, ch. Semantic Search in Peer-to-Peer Systems, pp. 634–664. Auerbach Publications, 2006.
21. A. L. Lopes and L. M. Botelho, "Improving multi-agent based resource coordination in peer-to-peer networks," *Journal of Networks*, vol. 3, no. 2, pp. 38–47, 2008.
22. Y. Zhu and Y. Hu, "Ess: Efficient semantic search on gnutella-like p2p systems," tech. rep., Department of ECECS, University of Cincinnati, 2004.
23. Y. Zhu and Y. Hu, "Efficient semantic search on dht overlays," *Journal of Parallel and Distributed Computing*, vol. 67, no. 5, pp. 604 – 616, 2007.
24. C. Tang and S. Dwarkadas, "Hybrid global-local indexing for efficient peer-to-peer information retrieval," in *NSDI*, pp. 211–224, 2004.
25. P. T. D. Talai, P. Fragopoulou, M. Mordacchini, M. Pennanen, K. Popov, and V. V. S. Haridi, "Peer-to-peer models for resource discovery on grids," tech. rep., Institution of System Architecture, 2006.
26. C. Mastroianni, D. Talia, and O. Verta, "A super-peer model for building resource discovery services in grids: Design and simulation analysis," in *EGC*, pp. 132–143, 2005.
27. J. Li, B. T. Loo, J. M. Hellerstein, M. F. Kaashoek, D. R. Karger, and R. Morris, "On the feasibility of peer-to-peer web indexing and search," in *IPTPS*, pp. 207–215, 2003.
28. P. Reynolds and A. Vahdat, "Efficient peer-to-peer keyword searching," in *Middleware '03: Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*, (New York, NY, USA), pp. 21–40, Springer-Verlag New York, Inc., 2003.
29. J. Casey and W. Zhou, "Reducing the bandwidth requirements of p2p keyword indexing," *International Journal*

of High Performance Computing and Networking, vol. 6, pp. 119–129, 2009.
30. M. Zaharia and S. Keshav, "Gossip-based search selection in hybrid peer-to-peer networks: Research articles," *Concurr. Comput. : Pract. Exper.*, vol. 20, no. 2, pp. 139–153, 2008.
31. H. Chen, H. Jin, Y. Liu, and L. M. Ni, "Difficulty-aware hybrid search in peer-to-peer networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, pp. 71–82, 2008.
32. Y. Sun, L. Sun, X. Huang, and Y. Lin, "Resource discovery in locality-aware group-based semantic overlay of peer-to-peer networks," in *InfoScale '06: Proceedings of the 1st international conference on Scalable information systems*, (New York, NY, USA), p. 40, ACM, 2006.
33. S. Kang, Y. Lee, D. Lee, and H. Y. Youn, "A landmark-based scalable semantic resource discovery scheme," *IEICE - Trans. Inf. Syst.*, vol. E90-D, no. 6, pp. 986–989, 2007.
34. A. Crespo and H. Garcia-Molina, "Semantic overlay networks for p2p systems," in *AP2PC*, pp. 1–13, 2004.
35. C. Tang, Z. Xu, and S. Dwarkadas, "Peer-to-peer information retrieval using self-organizing semantic overlay networks," in *Proceeding of ACM SIG-COMM*, pp. 178–186, 2003.
36. K. Arabshian, C. Dickmann, and H. Schulzrinne, *The Semantic Web: Research and Applications*, ch. Ontology-Based Service Discovery Front-End Interface for GloServ, pp. 684–696. Springer Berlin / Heidelberg, 2009.
37. Y. Zhu, H. Wang, and Y. Hu, "Intergrating semantics-based access mechanisms with p2p file systems," in *Proceedings of third international conference on Peer-to-Peer computing*, 2003.
38. C. R. Dunne, "Using mobile agents for network resource discovery in peer-to-peer networks," *ACM SIGecom Exchanges*, vol. 2, pp. 1–9, 2001.
39. P. Dasgupta, "Improving peer-to-peer resource discovery using mobile agent based referrals," in *AP2PC*, pp. 186–197, 2003.
40. Y. Kambayashi and Y. Harada, "A resource discovery method based on multiple mobile agents in p2p systems," in *Intelligent Agents in the Evolution of Web and Applications*, pp. 113–135, 2009.
41. O. Babaoglu, H. Meling, and A. Montresor, "Anthill: A framework for the development of agent-based peer-to-peer systems," in *IEEE Proceedings of 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, pp. 15–22, 2002.
42. O. Babaoglu and M. Jelasity, "Self-* properties through gossiping," *Philiosophical Transactions of the Royal Society A*, vol. 366, pp. 3747–3757, 2008.
43. K.-H. Yang, C.-J. Wu, and J.-M. Ho, "Antsearch: An ant search algorithm in unstructured peer-to-peer networks," *IEICE Transactions*, vol. 89-B, no. 9, pp. 2300–2308, 2007.
44. J. Gao and J. Zhang, "Clustered svd strategies in latent semantic indexing," *Information Processing and Management*, vol. 41, no. 5, pp. 1051–1063, 2005.
45. G. H. Golub and C. F. V. Loan, *Matrix Computations*. The John Hopkins University Press, 1996.
46. X. Liu, M. Chen, and G. Yang, "Latent semantic indexing in peer-to-peer networks," in *ARCS*, pp. 63–77, 2004.
47. S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, pp. 391–407, 1990.
48. M. Hasan and Y. Matsumoto, "Document clustering: before and after the singular value decomposition," tech. rep., Nara Institute of Science and Tecjnology. Technical Report TR-99. Processing Society of Japan, 1999.
49. P. Dasgupta, "A multiagent swarming system for distributed automatic target recognition using unmanned aerial vehicles," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 38, no. 3, pp. 549–563, 2008.
50. K. McCrary and B. Waters, "Jtella v0.7," October 2000.
51. C. University, "Cornell university: Medline text collection." Smart System, August 1999.
52. Wireshark, "Wireshark go deep." Online, 2010.
53. D. Bassu and C. Behrens, "Distributed lsi: scalable concept-based information retrieval with high semantic resolution," in *Proceedings of the 2003 Text Mining Workshop*, pp. 72–82, 2003.