

a Study of Multi Agent Based Resource Search Algorithms

Abstract—Resource sharing in a networked environment has been an evolved field of research in both industry and academia. In a dynamic, resource constrained environment, resource discovery plays a vital role in the resource sharing process. A distributed multi agent model for a dynamic, heterogeneous network has been proposed in this paper. Furthermore, three resource search algorithms have been investigated, namely: Random Walk, Directed Search and directed Search with Self-organisation. Usage of the approximate matching mechanism between resources and tasks descriptions has been proposed as a solution for the trade-off problem between the accuracy and speed. As performing random search might lead to traverse the whole network and increase the failing ratio, we have introduced a heuristic directed search to overcome this problem. Based on the agents' closeness, the Agents re-organise themselves by creating an overlay network of agents which are holding semantically close resources and then direct the search message accordingly. The proposed solution has been implemented in a simulated environment and simulation results have shown a significant improvement in the search time (calculated as a number of hops).

I. INTRODUCTION

Peer-to-peer(P2P) [1] and service-oriented [2] technologies have emerged as the dominant means for realizing scalable and interoperable distributed applications [3]. At the same time, these technologies lack the autonomy and ability to form a system in which the nodes across the network co-operate and interact in order to solve problems in a distributed manner. Hence, Multi Agent Systems (MAS) [4] have arisen as a promising technology to provide the system with required characteristics. Multi agent and peer-to-peer systems have emerged as a way to represent dynamic scalable networks. As MAS can be seen as a loosely coupled network of problem solvers, they are well suited for distributed tasks and complex problems [5].

In the Grid environment where so many different implementations are available, the need for semantic matching has become increasingly important [6]. Depending on the task requirements, exact matches are not always required; reducing the match accuracy requirement will result in faster service with less network resources consumption. This motivates the need for semantic matching which has been introduced as a technique to overcome the heterogeneity problem across the network. In contrast to syntactic matching, semantic matching returns a degree of matching rather than an exact match or no match. For instance, if an agent requires a machine with the following requirements: CPU =Pentium 4 with 2 GHz speed, Memory = 1 Gbyte, a machine with CPU =AMD with speed 2 GHz and 1 Gbyte of memory would do the job although the two machine descriptions are not exactly the same.

Due to the increasing scale of such systems and the degree of network change, performing exhaustive search for resources becomes infeasible from a performance perspective. Given the trade-off between the accuracy of match and network utilisation, this paper aims to show how altering two aspects of the agent's decision affects the system's performance metrics. Specifically, we define the performance factors as, average hop counts, matching error and the percentage of search failure.

The rest of the paper has been organised as follows: section two has been devoted to the related literature, while the resource search model has been discussed in section three. System evaluation and experimental results are presented in section four. Finally, conclusions and suggestions for future development are discussed in section five.

II. RELATED WORK

Researchers in the industry and academia have developed many frameworks and tools for managing the resources description and discovery in distributed environments. The Globus toolkit [7] Monitoring and Discovery System (MDS4) includes a set of monitoring and discovery techniques for services and resources in a distributed environment. [8] have proposed a dynamic search algorithm in a peer-to-peer system which meant to overcome the traditional flooding and random search algorithms like searching the whole network aggressively and the long search time respectively. the proposed algorithm is a modified version of the traditional algorithms (random walk and flooding) by maintaining a probability function which provide some knowledge to the peer from the previous states so it could use this knowledge to predict the best move.

[9] have studied the performance of the flooding search algorithm in a resource discovery context. Using this approach the node/agent sends a request to all of its neighbours which in turn contact their own neighbours and so on. Flooding algorithms maintain no knowledge about the network or the available resources. Furthermore, the authors have added two variations to the flooding algorithm which are teeming and random walk [10] with the aim of reducing the exponential increase in the number of messages.

A dynamic network topology adaptation technique has been proposed by [11] with the aim of improving the search performance of the flooding algorithm by creating semantic communities based on query traffic. In contrast with our work, the authors have focused more on changing the topology based on statistical heuristics by collecting nodes with similar interests together in small communities. They assumed that

the file request could be fulfilled by nodes with in the same community.

Resource search in a multi agent environment has been addressed by [10], who proposed a mechanism in which each agent can maintain a list of k contacts and their available resources. Based on the information which has been stored locally, the agent could direct its search query to a subset of the agent population using traditional search methods (i.e flooding, teeming or random walk). The proposed method formed some sort of network of caches and did not impose any semantics or re-organisation capability in the agents.

In contrast to the literature which has been mentioned above, the work presented has made use of the resource and task descriptions along with semantic matching to support the search process. Re-organising the network's connections based on the actual available resources' information rather than using a probabilistic measurement would help forwarding resource search messages more efficiently and decrease the amount of messaging required to fulfill the request. Furthermore, this re-organisation helps in creating agents communities which hold semantically similar resources and forward resource search messages directly to a member of the community rather than all or random members which could improve network traffic and reduce the messaging time.

III. RESOURCE SHARING MODEL

A. Simulation Tools

In order to support our investigation into the micro-macro link between the selected search parameters, and their global consequences with respect to our system performance metrics; we have chosen to run multiple simulations of our resource discovery scenario across the available state space as defined by our parameters. A number of agent-based simulation tools exist such as Mason [12], NetLogo [13] and Swarm [14] in addition to specific agent languages [15] and platforms e.g. Jason [16]. From this myriad of tools we have selected to use PreSage [17] a Java based rapid prototyping and simulation tool developed to support the animation of agent societies. Similarly to the platforms and tools mentioned above PreSage allows us to script a series of simulation runs with varying parameters, and record results from a global perspective. We can also:

- conditionally trigger exogenous events such as changes to the network;
- quickly define and adapt our agent communication protocols such as our resource search in a flexible manner;
- allow individual agents to control the application layer network (from a localised perspective);
- simulate large numbers of heterogeneous agents, and is relatively easy to deploy to high throughput computing platforms (HTPC) such as Condor [18].

B. Network Formation

To test the proposed algorithms, a distribute multi agent model which represents the dynamic networked system has been designed. The system has been modeled as a population

of agents which communicate with each other and with the environment via messages of different types based on what kind of information the agent needs to convey. At the beginning of the simulation, the agents' population is created with each agent encapsulate certain internal information which are known by the owner agent only (autonomy). after that, the system starts activating the agents, one agent per simulation time cycle. While the simulation is running more agents will join the system till the saturation condition $t = n$ reached where t is the simulation cycle time and n is the number of agents. Upon activating new agent, the agent starts operating by connecting to the network via sending a *hello* message to a random active agent. The *hello* message has the following formula:

$$hello = \{To, From, TS, Resource\} \quad (1)$$

Where, *To* is the sender's ID; *From* is the destination's ID; *TS* is the sending time (simulation cycle) and *Resource* is the sender's resource description.

After connecting to the network, the agent would start executing tasks or donate resources and it does not need to use *hello* message again as the resource search message will work as *hello* in addition to resource discovery. The task creation process and resource description are done by using a random uniform distribution to simulate the fact that the tasks and resources are changing overtime (dynamism). If the agent needs some resources to execute its tasks and these resource are not available then it starts to send a resource search message of the form:

$$RSM = \{To, From, TS, TaskDescription, Hops\} \quad (2)$$

Where: *RSM* is the resource search message; *From* is the destination's ID; *TS* is the sending time (simulation cycle); *TaskDescription* is the task description as shown in equation 6; and *Hops* is the number of hops.

Other type of message is used to convey the match found message as follows:

$$RFM = \{To, From, MyResource, TS\} \quad (3)$$

Where: *RFM* is the Resource Found Message; *MyResource* is the match's owner resource description.

C. Resource and Task Description's and Matchmaking

In a distributed computing environment, many standard methods have been developed to describe resources and tasks extensively. For instance, WSDL (Web Service Description Language) [19], OWL (Web Ontology Language) [20] and RDF(Resource Description Framework) [21]. For the purpose of simulation and since the focus of this paper is on improving the network traffic performance, resources and tasks have been described using the following formulas:

$$RD = \langle r_1, r_2, r_3 \rangle \quad (4)$$

Where: RD is the resource descriptor as tuple of three elements.

$$Resource = \{RD, Av, Dn, T_s, T_e\} \quad (5)$$

Where: Av is the available amount; Dn is the donated amount; T_s is the donation start time and T_e is the donation end time.

$$Task = \{RD, Ra, T_s, T_e\} \quad (6)$$

Where: Ra is the required amount of resources; T_s is the task's start time and T_e is the task's end time.

Resource vs. request matchmaking process takes place using the Manhattan [22] distance measure between resources and tasks vectors as shown in Equation 7.

$$Dis(R^A, R^B) = \sum_{i=0}^2 |r_i^A - r_i^B| \quad (7)$$

where: Dis is the Manhattan distance between resource R^A and resource R^B . Resources and tasks have been referenced as R because the system uses the same way to describe them.

D. Resource Search Algorithms

The resource discovery problem has been considered as a search problem with the aim of finding the required resource across the network. As the network could scale too largely, the aim is to direct the search using the available resource information and by applying the matchmaking technique. Directing the search would decrease the number of hops required to discover the resources and increase the chances of doing that before the request message times out. Based on the ideas expressed in [23], three algorithms have been implemented and tested under simulation environment. The first algorithm operates as a general search function which handles the receipt of the resource search message. Based on the initiator identifier, the algorithm checks if the sender is already known to the receiver and act accordingly. The receiver has two options to do, the first option is to add the sender to the contact database if it has not contacted it before otherwise just update its record. Afterwards, the receiver checks if the required resource is available within the requested allowable error, if yes then reply with resource found message to the initiator. Otherwise the receiver will have to forward the search message based on algorithm 2 if and only if the search message has not timed out. If there is no more peer to forward the message to then the search message will have to be returned to the previous agent in the list of recipients. The search process fails in two cases which are: case one, if the search message times out ($tll = 0$) and case two if the search message returns to the initiator which means it has been traversed the whole network.

To utilise the network connectivity, two different methods have been used to add new contact to the agent's contact database upon receiving a resource search message as shown in algorithm 3. the first method is to add the newcomer based on First In first Out (FIFO) technique, using FIFO the

agent removes the older contact and add the newcomer. the second method is based on the degree of similarity between the newcomer's resources and receiver's resources. If the newcomer's resources are semantically closer than any contact in the contact database then the agent replaces the one with less similarity and add the newcomer.

Algorithm 1: General Search Function for handling receipt of a Resource Search Message.

```

input : The agent's contact database ContactDB. A
         resource search message rsm
         <to,from,initiator, reqResource, maxError,
         path, ttl> where rsm.initiator records the
         details of the agent requiring the resource.
         rsm.reqResource describes the resource the
         initiator needs. rsm.maxError defines the
         allowable distance between a matched resource
         and rsm.reqResource. rsm.path is an ordered
         (by time) list of agents who have received the
         search message and finally rsm.ttl logs the
         remaining time a message has to live before a
         search is considered to have failed.

if  $\neg$ ContactKnown(rsm.initiator, ContactDB) then
    AddPeerContact(rsm.initiator, time,
    ContactDB);
else
    UpdatePeerDetails(rsm.initiator, time,
    ContactDB);
end
if MatchError(rsm.reqResource, my.resource) <=
rsm.maxError then
    //MatchError is defined in equation
    7;
    send resource found msg to rsm.initiator;
else
    if rsm.ttl < 1 then
        send search failed message to initiator;
    else
        add my.name to rsm.path;
        decrement rsm.ttl;
        nextpeer  $\leftarrow$  SelectPeer(ContactDB,
        rsm.path, rsm.reqResource);
        if nextPeer! = null then
            forward rsm to nextpeer
        else
            return rsm to previous peer in rsm.path
        end
    end
end

```

IV. SYSTEM EVALUATION AND EXPERIMENTAL RESULTS

To assess the fidelity of the proposed algorithms we have designed experiments and run them under simulation environment with the following configuration:

Algorithm 2: SelectPeer Function, selects the next peer to receive a Resource Search Message

input : *reqResource*, *path*

switch the value of *Strategy* **do**

case *RandomSearch*

 return random contact from *ContactDB* excluding those already in the path

case *DirectedSearch*

 sort *ContactDB* by *MatchError(ContactDB.record, reqResource)* in ascending order;
 exclude those listed in path;
 return top match;

endsw

endsw

Algorithm 3: AddPeerContact Function, decides whether or not to add a new peer into an agent's *ContactDB*

input : The agent's contact database *ContactDB*, *k* the maximum size of the agent's contact database, *my.resource* the agents own resources and finally *peer* the details of the new peer

switch the value of *Strategy* **do**

case *Static*

if *size(ContactDB) < k* **then**
 add peer to database
 end

case *FirstInFirstOut*

 add peer to database;
 if *size(ContactDB) > k* **then**
 remove oldest record;
 end

case *HighestSimilarity*

 add peer to database;
 if *size(ContactDB) > k* **then**
 sort *ContactDB* by *MatchError(ContactDB.record, my.resource)* in descending order;
 remove top record;
 end

endsw

endsw

- Config-1: Random Search with First In First Out (FIFO) contacts.
- Config-2: Directed Search with First In First Out (FIFO) contacts.
- Config-3: Directed Search with Self-Organisation.

For simulation purposes four parameters have been set as follows:

- Number of Agents = 300.
- Maximum number of contacts for each agent = 10.

- Maximum number of hops each resource search message can do (ttl) = 100.
- Required Accuracy of matching between [0,12].

The aim of these experiments is to evaluate the effects of each configuration on the system performance which has been measured by the following parameters:

- System Mean Average Error (MAE).
- Average Hop Count to find the resource successfully.
- Percentage of Failures : The ratio of failed requests to the total number of requests.

The first experiment was to evaluate the each configuration against the MAE as shown in fig 1. As the aim is to compare between the allowable matching error and the actual matching error, figure 1 shows that with small allowable matching error all the configuration behave in the same way. The reason for this behaviour is, while the allowable matching error is too small there is no difference with the actual matching error specially with allowable matching error of 0 (exact match). In contrast, as the allowable matching error increases the directed search algorithms start to do better with directed search with self-organisation does slightly better than directed search.

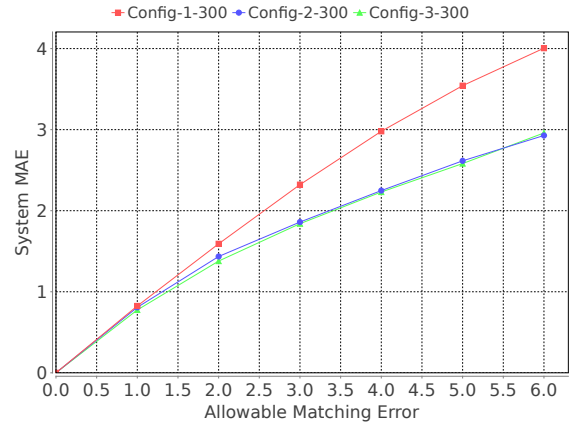


Fig. 1. A comparison between the allowable matching error the found matching error (system MAE).

The second experiment was to evaluate the average time required for a successful resource search request. As the experiment has been run under simulation environment it was not possible to compute the actual time so we have simulated the time using the number of hops (as each hop required one time cycle). The experiment has shown a noticeable difference between the three configuration with random walk algorithm did the worse case in compare to directed search algorithms. With small allowable error (i.e. better quality of matching) directed search with self-organisation has shown the best result. Furthermore, as the allowable matching error increases (less matching quality is acceptable), all the configurations have started to behave in the same way because with high allowable error the search could be done within a few hops as the agent accepts any available resources.

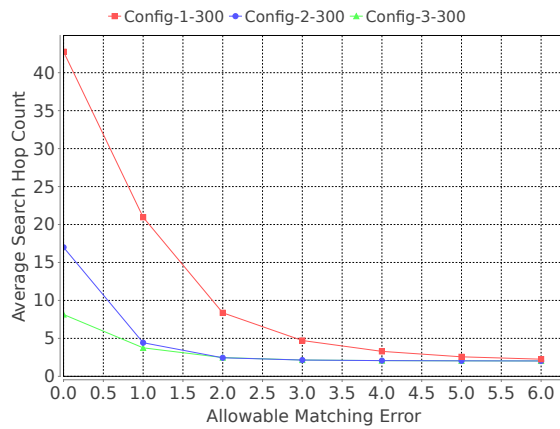


Fig. 2. Average number of hops a resource search message will make before successfully finding the requested resource.

V. CONCLUSION

A multi agent architecture for modeling a dynamic heterogeneous network environment has been introduced in this paper. Furthermore, a semantic inspired resource search algorithm has been proposed. As the semantic matching returns a degree of similarity rather than an exact matching, we have made use of this idea to adapt the conventional random walk algorithm to direct the search messages based on the semantic closeness. Decreasing the required matching degree could decrease the search time within an acceptable matching quality. In a dynamic time constrained environment re-organising the network by creating overlay networks of agents which are holding semantically close resources improves the system performance by forwarding the search messages heuristically. The experimental results have proven this idea correct as the directed search algorithms produced better matching quality and less number of hops.

Currently we are building on the idea of this paper by enabling the system to create virtual organisations of the agents which are semantically close. Furthermore, the virtual organisation could be treated as a single entity by forwarding messages to the organisation rather than a single agent. To do so, a member of the organisation should be selected as the organisation's head based on some criteria like performance or voting.

REFERENCES

- [1] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Communications Surveys and Tutorials*, vol. 7, pp. 72–93, 2005.
- [2] H. Demirkan, R. J. Kauffman, J. A. Vayghan, H.-G. Fill, D. Karagiannis, and P. P. Maglio, "Service-oriented technology and management: Perspectives on research and practice for the coming decade," *Electron. Commer. Rec. Appl.*, vol. 7, pp. 356–376, December 2008. [Online]. Available: <http://dx.doi.org/citation.cfm?id=1465743.1466108>
- [3] E. Garcia, A. Giret, and V. Botti, "Software engineering for service-oriented mas," in *Proceedings of the 12th international workshop on Cooperative Information Agents XII*, ser. CIA '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 86–100. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-85834-8_9
- [4] H. Aziz, "Multiagent systems: algorithmic, game-theoretic, and logical foundations by y. shoham and k. leytton-brown cambridge university press, 2008," *SIGACT News*, vol. 41, pp. 34–37, March 2010. [Online]. Available: <http://doi.acm.org/10.1145/1753171.1753181>
- [5] M. Fasli, *Agent Technology for e-Commerce*. West Sussex, England: John Wiley and Sons Ltd, 2007.
- [6] S. A. Ludwig and S. M. S. Reyhani, "Semantic approach to service discovery in a grid environment," *Journal of Web Semantics*, vol. 4, pp. 1–13, 2006.
- [7] J. M. Schopf, L. Pearlman, N. Miller, C. Kesselman, and A. Chervenak, "Monitoring the grid with the globus toolkit mds4," *Journal of Physics: Conference Series*, vol. 46, pp. 521–525, 2006.
- [8] L. Tsungnan, L. Pochiang, W. Hsinping, and C. Chiahung, "Dynamic search algorithm in unstructured peer-to-peer networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, pp. 654–666, 2009.
- [9] B. Yang and H. Garcia-Molina, "Improving search in peer-to-peer systems," Stanford InfoLab, Technical Report 2001-47, October 2001. [Online]. Available: <http://ilpubs.stanford.edu:8090/513/>
- [10] V. V. Dimakopoulos and E. Pitoura, "Performance analysis of distributed search in open agent systems," in *Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, ser. IPDPS '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 20.2–. [Online]. Available: <http://dl.acm.org/citation.cfm?id=838237.838604>
- [11] V. Cholvi, P. Felber, and E. Biersack, "Efficient search in unstructured peer-to-peer networks," in *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, ser. SPAA '04. New York, NY, USA: ACM, 2004, pp. 271–272. [Online]. Available: <http://doi.acm.org/10.1145/1007912.1007955>
- [12] N. Minar, R. Burkhart, C. Langton, and M. Askenazi, "The swarm simulation system, a toolkit for building multi-agent simulations," 1996. [Online]. Available: citeseer.ist.psu.edu/minar96swarm.html
- [13] S. Tisue and U. Wilensky, "Netlogo: A simple environment for modeling complexity," in *International Conference on Complex Systems*, 2004, pp. 16–21.
- [14] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan, "Mason: A multiagent simulation environment," *Simulation*, vol. 81, no. 7, pp. 517–527, 2005.
- [15] A. S. Rao, "Agentspeak(1): Bdi agents speak out in a logical computable language," in *Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world : agents breaking away: agents breaking away*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1996, pp. 42–55. [Online]. Available: <http://dl.acm.org/citation.cfm?id=237945.237953>
- [16] R. H. Bordini, M. Wooldridge, and J. F. Hübner, *Programming Multi-Agent Systems in AgentSpeak using Jason (Wiley Series in Agent Technology)*. John Wiley & Sons, 2007.
- [17] B. Neville and J. Pitt, "Presage: A programming environment for the simulation of agent societies," in *Programming Multi-Agent Systems*, ser. Lecture Notes in Computer Science, K. Hindriks, A. Pokahr, and S. Sardina, Eds. Springer Berlin / Heidelberg, 2009, vol. 5442, pp. 88–103, 10.1007/978-3-642-03278-3_6. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-03278-3_6
- [18] T. Tannenbaum, D. Wright, K. Miller, and MironLivny, "Condor – a distributed job scheduler," in *Beowulf Cluster Computing with Linux*, T. Sterling, Ed. MIT Press, October 2001.
- [19] S. J. Samuel and T. Sasipraba, "Article: Trends and issues in integrating enterprises and other associated systems using web services," *International Journal of Computer Applications*, vol. 1, no. 12, pp. 17–20, February 2010, published By Foundation of Computer Science.
- [20] G. K. Saha, "Web ontology language (owl) and semantic web," *Ubiquity*, vol. 2007, pp. 1:1–1:1, September 2007. [Online]. Available: <http://doi.acm.org/10.1145/1295289.1295290>
- [21] G. Klyne and J. J. Carroll, "Resource description framework (rdf): Concepts and abstract syntax," February 2004.
- [22] N. Ljubešić, D. Boras, N. Bakarić, and J. Njavro, "Comparing measures of semantic similarity," in *Proceedings of the 30th International Conference on Information Technology Interfaces*, V. Lužar-Stiffler, V. Hljuz Dobrić, and Z. Bekić, Eds. Zagreb2: SRCE University Computing Centre, 2008, pp. 675–682.
- [23] M. Al-Asfoor, B. Neville, and M. Fasli, "Heuristic resource search in a self-organised distributed multi agent system," in *Self-Organizing Systems*, ser. Lecture Notes in Computer Science, F. Kuipers and P. Heegaard, Eds. Springer Berlin / Heidelberg, 2012, vol. 7166, pp. 84–89, 10.1007/978-3-642-28583-7_8.