A Comparative Evaluation of Agent Location Mechanisms in Large Scale MAS

David Ben-Ami
Technion - Israel Institute of Technology
Haifa 32000, ISRAEL
davb@tx.technion.ac.il

Onn Shehory
IBM Haifa Research Lab
Haifa 31905, ISRAEL
onn@il.ibm.com

ABSTRACT
Agents in open multi-agent systems (MAS) need means for locating other agents with which they may collaborate. To address this need, several agent location mechanisms were suggested. Two major approaches dominate agent location mechanisms: a centralized approach using middle agents, and a distributed, peer-to-peer approach. Agent designers, when designing agents to be part of open MAS, should consider these approaches, to provide the agents with appropriate agent location capabilities. However, selecting an agent location approach, let alone a specific solution, is a nontrivial task. In this study we address this difficulty. We perform a systematic comparative evaluation of agent location approaches. We measure the performance of these approaches subject to various MAS configurations. We draw conclusions regarding the conditions in which each approach is preferable. Prior evaluations fall short in addressing realistic MAS settings. In particular, our evaluation is the first to examine scalability of agent location mechanisms in terms of both system size (thousands of agents) and network distribution (over multiple hosts). We present advantages and shortcomings of the examined approaches.

Categories and Subject Descriptors
I.2.1 [**Distributed Artificial Intelligence**]: *Multiagent systems*

General Terms
Algorithms, Management, Measurement, Performance, Design.

Keywords
Multi Agent Systems; Location Mechanisms; Peer to Peer; Distributed; Evaluation; Simulation.

AAMAS'05, July 25-29, 2005, Utrecht, Netherlands.

1. INTRODUCTION
Agents in open MAS may need to perform tasks for which they do not have the required capabilities or capacities. They may however be able to delegate their tasks to other agents to be performed by them. To do that, agents need to either hold contact information of the other agents, or be able to acquire such information. In open MAS, holding up-to-date contact information of other agents is impractical. Hence, a major problem in open MAS is the agent location problem.

Solutions to the agent location problem follow two major approaches: a centralized approach using middle agents, and a distributed, peer-to-peer approach. Agent designers, when designing agents to be part of open MAS, should consider these approaches, to provide the agents with appropriate agent location capabilities. However, selecting an agent location approach, let alone a specific solution, is a nontrivial task.

The majority of agent location mechanisms (e.g., [4,6,11,19]) adopt the centralized approach. However, centralized mechanisms in large-scale, distributed MAS introduce problems. Most prominently, central middle agents are potential single points of failure, which can compromise the overall fault-tolerance of the system. Some solutions attempt to overcome those problems via distribution. In [9], a distributed matchmaking solution is presented, where multiple matchmakers with partial information each are coordinated using a special protocol. Another approach [13] takes advantage of the fact that each agent already knows about its own capabilities and uses a peer-to-peer (recursive) search for locating an agent with the needed capability. The distributed solutions introduce communication and coordination overheads.

Given the limitations each solution approach introduces, one needs means to decide which approach is more appropriate for specific MAS. An attempt to evaluate these approaches was introduced in [2]. That work however does not address well realistic MAS settings. In particular, the size of MAS examined is small to medium, and distribution over a network is not examined. In contrast, in this work we evaluate and compare centralized and distributed location mechanisms in large scale MAS (consisting of thousands of agents) distributed over a real multi-host network.

We derive system characteristics in which each mechanism performs better. We show that although both location approaches provide agents with references they seek, they differ in speed and quality of the results provided. We further prove that the centralized mechanism works better for smaller, lightly loaded systems, and that the distributed mechanism performs better for larger, highly loaded systems.

Our results were arrived at via experiments performed on a MAS deployed on a multi-host network. We have implemented both centralized and distributed agent location mechanisms, evaluated them, and compared their performance and robustness under various configurations. Our methodology and results are presented in this paper.

2. AGENT LOCATION APPROACHES

The most common location mechanisms are central directory services in the form of middle agents of different sorts (e.g. facilitators, matchmaker and registry agents). The problems with the centralized solutions are the potential limitations on scalability and fault tolerance of the system. In a large-scale system with heterogeneous agent capabilities the workload on the middle-agent that handles the location requests for the whole system can increase rapidly and degrade the overall performance of the system and of individual agents. Moreover, the middle agent that supplies the directory services becomes one of the system failure points and/or communication bottlenecks. Failures in the middle agent can be critical and paralyze the entire system. These potential performance and fault-tolerance drawbacks call for alternative location mechanisms in open MAS that offer scalability and fault-tolerance while preserving the inherent accuracy of the centralized mechanisms.

Different approaches were suggested to overcome the above mentioned problems. These solutions distribute the information about agents' capabilities among other system elements. Distributed matchmaking [8] suggests coordination among multiple matchmakers, each holding only part of the contact information of the MAS. A variation of this approach might take advantage of deployment of agents on different hosts or network segments. The system designer assigns each host or segment of the system with a local matchmaker that provides service to agents in its vicinity (its segment). The local matchmaker can consult its peers or a central matchmaker whenever it cannot provide an answer to a local query. This type of solution reduces communication traffic and confines it to network segments (in which communication is fast). It further reduces message queue sizes, thus improving scalability and fault tolerance. This approach is applicable mainly in systems that have a hierarchical topology, in which information sharing can be confined to local segments. In systems with very large segments the problems of scalability are only marginally relieved by this approach (because the large segments become overloaded systems which have local bottlenecks). Another case in which this approach is not useful is systems with many cross-links between segments, in which case the overhead of coordinating among local matchmakers might be greater than the benefit from their distribution.

A peer-to-peer approach [10] takes advantage of the fact that each agent already knows about its own capabilities and those of a few peers, and uses peer-to-peer (recursive) search for locating agents with the needed capability. An agent broadcasts a query for reference to its neighbors, and an agent that receives such a request either offers its services to the original caller or broadcasts the request to its own neighbors. This approach relies on each agent indeed holding a neighbor list of its peers and adhering to the location protocol. The communication among agents in this approach is essential and the overall communication traffic overhead may be large. It also requires a connection model among the agents that will ensure a high number of correct answers (good hit ratio for queries) and in the same time control communication overhead so that the network is not overwhelmed by the messages of the location protocol.

3. SIMULATION DESCRIPTION AND RESULTS

3.1 Simulation Testbed Description

To examine location mechanisms, we developed a simulation testbed that enables the definition of multiple different MAS configurations and the deployment of agents on multiple hosts. The testbed was developed in Java 2, using RMI for inter-agent communication. Using the simulator, we have thoroughly tested both the distributed and the centralized agent location mechanisms in a wide set of configurations. We have specifically measured the following metrics:

RefTime – The average time for an agent to find another agent with the sought capabilities (that is – to receive a reference to a relevant other agent). We also measure the average time for an agent's delegated tasks to complete (denoted as *DoneTime*).
HitRate – The average rate of task accomplishment, the ratio between the number of tasks that were successfully processed by an agent and the total number of tasks assigned to it.
MsgCount – The average number of both outgoing and incoming messages (generated as a result of the use of the location mechanism) per agent.

In our simulation, a MAS is implemented as a group of agent threads. A set of capabilities, referred to as *system capabilities*, is allocated to agents in the system. Each agent is allocated a private, randomly generated, subset of *system capabilities* referred to as *local capabilities*, and a set of randomly generated tasks (each task requires one capability for its execution). A central location mechanism is implemented as an independent *matchmaker* thread.

The agents are identical in their logic but they differ in their capabilities and tasks. When seeking other agents to perform a task it cannot perform by itself, an agent may use one of the two location mechanisms: centralized or distributed, to locate a reference for a relevant agent.

The distributed location mechanism relies upon an agent connection graph. This graph is determined at system setup according to a connection model. Note that a connection between two agents *A, B* does not refer to the existence of a communication line between the two. Rather, it refers to the fact that *A* knows of *B* or its address a-priori. Thus, the so-called connection graph is fully expressed by the neighbor lists held by the individual agents. In our tests of the distributed location mechanism, we used 2 principal connection models:

Grid connection - connecting each agent to its neighbors in a grid of agents. This model is typical to designed, close

systems, where agents are known in advance to the designer. This model is very convenient for analysis.

Random connection – connecting each agent to *k* randomly selected agents in the system. This model is typical to emergent, open systems, where agents build their contact list upon interaction with others.

The important parameters in this respect are the size of the neighbor list of each agent (out-rank of the connection graph) and the search horizon parameter. The latter is determined by the *Max_TTL* parameter. *TTL*, (Time To Live) determines the number of hops a query travels from its originator. The time complexity of the distributed discovery algorithm in ordered graphs is O(log*N*) (using the emergent small world actual connectivity graph among agents [20]).

In the distributed mechanism, agents attempt to execute their tasks, while attending to incoming messages from their peers. A uniform delay between tasks (which may also be set to 0) enables us to control the average workload on the system

3.2 Simulation Structure and Algorithms

This section provides a description of the structures, activities and protocols of simulation entities.

3.2.1 The algorithm of the agent thread.

Initially, an agent is allocated a set of capabilities and a set of tasks randomly. The agent then successively tries to find a reference to service provider agents that can perform a task it cannot perform by itself. Upon receiving references to other relevant agents, the agent selects one of the references – a service provider agent – and delegates the task to it. A task is considered done after the requester agent receives an acknowledgement from the service provider agent.

An agent that received a request for a reference either offers its services to the original requester, if it can perform the task, or broadcasts this request to its neighbors (provided that the request's *TTL* has not expired). An agent that is requested to perform a task on behalf of another agent does so and, upon completion, sends an acknowledgement to the requesting agent.

3.2.2 The algorithm of the matchmaker thread.

In the case of a central location mechanism, each agent sends its list of local capabilities to the matchmaker thread. Using these inputs, the matchmaker constructs and holds, for each system *capability*, a list of agents that provide it. To reduce complexity, these lists are not exhaustive. Instead of including all the agents that provide a specific capability in a capability provision list, only $N^{1/2}$ service providers – *N* being the number of agents in the system – are included in each list. Upon receiving a request for reference, the matchmaker selects (based on a predefined randomized selection policy) one service provider from the relevant capability provision list and sends a reference to the requester. The combination of square-root list sizes with random selection from the lists has experimentally shown to minimize the workload of execution on the service provider agents.

The size of the capability provision list was determined heuristically based on experiments which showed that long lists (close to N) proved to overload the matchmaker setup phase, and constant sizes have affected agent performance negatively. Yet we have no proof of optimality for square-root list sizes.

3.2.3 The algorithm of the monitor thread.

The monitor thread periodically queries all of the agents for their progress rate; the agents return their counter of done tasks. When all agents have finished their tasks, or the system is no longer progressing, or a global timeout was reached, the monitor thread stops all agents and closes log files.

3.2.4 Controlling system workload.

The rate at which agents request references depends on the time it takes them to complete tasks; since in a simulated system this time is negligible (there is no execution logic of the tasks), we introduced artificial delays to prevent unrealistic overloads. A constant delay between tasks, denoted by *InterTaskDelay*, was added to the agent's execution algorithm. This delay reduces the request rate and thus the workload on the location mechanism and on the service provider agents. By varying the value of *InterTaskDelay*, we controlled the system's workload.

3.3 Experimental Results

Several sets of experiments were conducted on randomly generated simulated MAS. Each set of experiments was repeated 5 times and results are the average of these 5 iterations. Each agent was allocated 50 tasks. Averaging over tasks was performed as well.

The first phase of experiments was conducted on a medium size (up to 200 agents) single host system (all the threads competed for resources on a single host). The second phase, conducted on a large scale (up to 1500 agents) multiple host system (consisting up to 15 hosts, each running a segment of the system), will be described later in this paper.

The parameters that were set in our experiments, their notations and units are described below:

1. System scale, denoted by *N* or *NumAgents*, is the number of the agents in the system.
2. System workload is inversely proportional to the delay between tasks. Therefore, we express it via this delay, denoted by *InterTaskDealy*, measured in milliseconds.
3. Connection model, may be either an ordered *grid* (with 2 or 4 neighbors) or *random* (2, 3 or 4 neighbors).
4. Search horizon, denoted by *Max_TTL*, is the maximal number of hops per message in the distributed location mechanism.
5. *Capability distribution* (*CD*) is the percentage of local capabilities out of the system capabilities allocated to each agent.

In the first phase of the simulation we were specifically interested in examining the effects of system workload, system scale, capability distribution, connection model and search horizon. We have thus conducted 4 sets of experiments in which

these parameters were examined. The experiments were performed such that, when testing (and varying the value of) a specific parameter, other parameter values were kept constant. The specific settings of these experiments' sets and their results are described below.

3.3.1 Effect of system scale (set 1).

For this set of experiments, we have constructed configurations with 9, 16, 25, 49, 64, 100, 121, 144 and 196 agents (grids were square, for convenience). The other parameters were set as follows: the workload was set to high (0 delay between tasks); the connection model was a grid connection with 2 neighbors and *Max_TTL* = 5; the *CD* was set to 30%. In these experiments we measured the response time (*RefTime*) for the distributed location mechanism, denoted by *DrefTime*, and for the centralized location mechanism denoted by *CRefTime*. The results are presented in Figure 1.

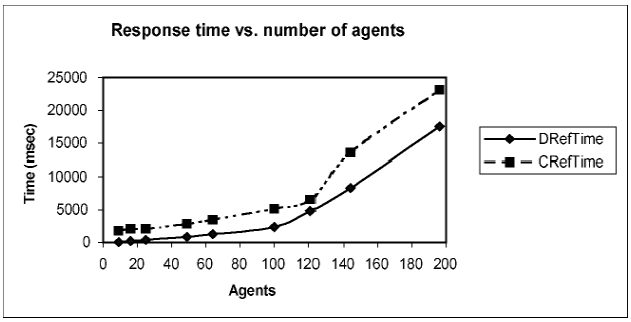

Figure 1. The response times of the distributed location mechanism are shorter than the response times of the central location mechanism.

We have further measured *MsgCount*, the average number of messages passing through an agent. The results appear in Figure 2, where *DMsgs* refers to the average message count for the distributed location mechanism, and *CMsgs* refers to the average message count for the centralized location mechanism.

As appears in Figure 1, the response times of the centralized location mechanism are significantly higher than those of the distributed location mechanism. Note however that the variance of the response time (not presented in the figure) is greater for the distributed location mechanism than it is for the centralized mechanism. The message count (Figure 2) is significantly higher for the distributed mechanism, however it is almost independent of the system scale. These results suggest that, in terms of responsiveness, the distributed location mechanism is more efficient than the centralized mechanism in systems with high workload, regardless of system size. Although this advantage of the distributed mechanism results in a communication overhead, this overhead is independent on system scale, therefore it does not affect scalability.

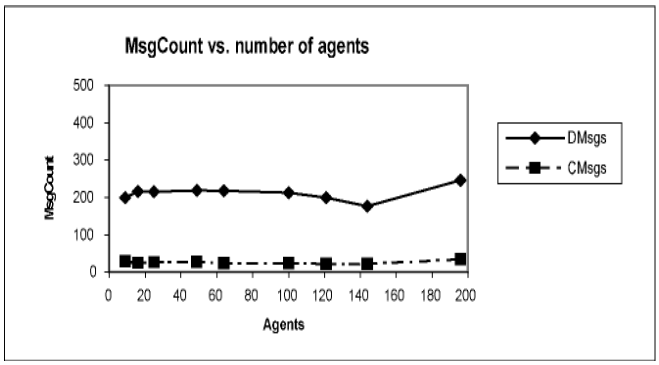

Figure 2. The average message count per agent is higher for the distributed location mechanism than it is for the centralized location mechanism.

3.3.2 Effect of system workload (set 2).

For this set of experiments, we varied workloads (by varying *interTaskDelay*), keeping other parameters constant. Recall that lower delays mean higher workloads. The delays checked were 0, 100, 200, 300, 400, 500, 750 and 1000 milliseconds. Other parameters were set as follows: system scale was set to two sizes, 49 and 100 agents; the connection model was a grid with 2 neighbors and *Max_TTL*=5; the *CD* was set to 30%. We measured *DRefTime* and *CRefTime*. The results are presented in Figure 3.

Note that for both system sizes, 49 and 100, at some point – we denote this point as the *phase transition point* – the response time of the centralized location mechanism becomes better (i.e. shorter) than the response time of the distributed mechanism. Phase transitions occur when the system's workload goes below some threshold.

Figure 3 shows that, as workload decreases, the response time of the centralized mechanism improves. Below some workload threshold, the centralized mechanism is better than the distributed mechanism. This workload threshold is lower as system scale increases. On the other hand, for high workloads – the distributed mechanism is significantly better than the centralized one, and this advantage is more prominent for larger systems.

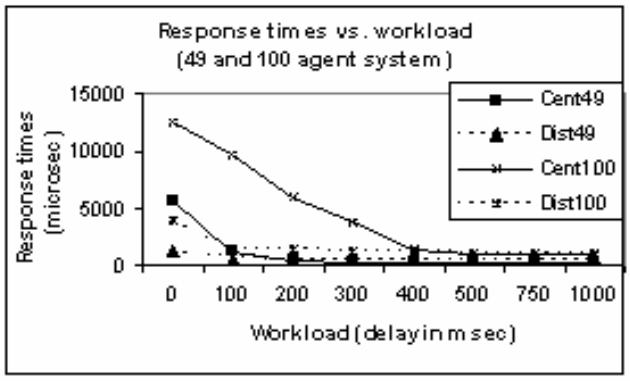

Figure 3. The response time improves as workload decreases (and delay increases). The improvement in the centralized case is sharper than it is in the distributed case. Phase transition occurs at ~150 ms and ~400 ms, 49 and 100 agents, respectively.

3.3.3 Effect of capability distribution (set 3).

For the third set of experiments, we have constructed configurations with *CD*s of 10%, 20%, 30%, 40%, 50%, 60% and 70%, keeping other parameters constant. Recall that *CD* refers to the ratio between the number of local capabilities and system capabilities. The other parameters were set as follows: scale was set to 49 agents; workload was set to high (setting delay to 0); the connection model was a grid with 2 neighbors and *Max_TTL*=4. We measured the *HitRate* (i.e. the average rate of accomplished tasks). The results appear in Figure 4, where *DHitRate* and *CHitRate* denote the hit rates of the distributed and the centralized mechanisms, respectively.

For brevity, response time and message count measurements for this set are not presented, however they were not surprising. Higher *CD* values result in shorter response times and lower message counts. A high hit rate for a centralized mechanism is not a surprise either. However, interestingly, the distributed mechanism achieved close to 100% hit rate for *CD*s above 20%; even for a 10% distribution it achieved a 75-80% hit rate (Figure 4). For *CD*s of 20% or less, the results of the centralized mechanism are better than those of the centralized one. This however depends on the connection model and does not scale up in larger systems and random connection models (Figure 5).

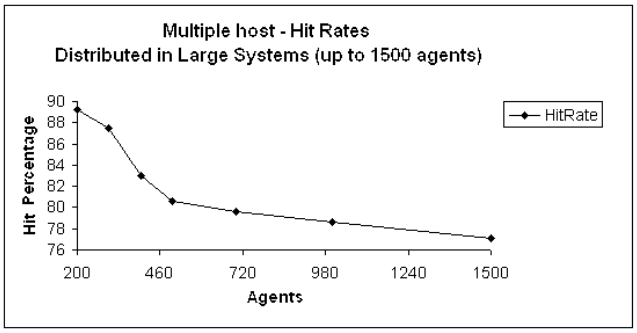

Figure 4. For small systems, hit rates are very high for both mechanisms.

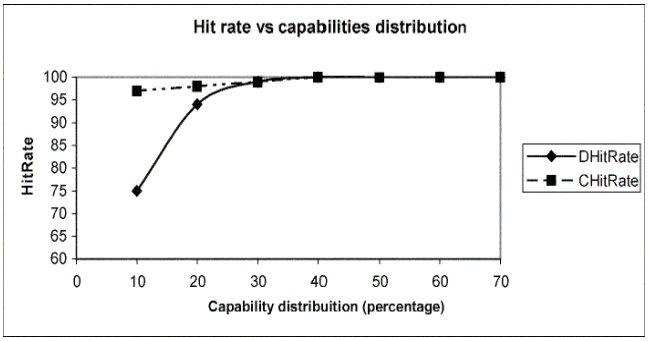

Figure 5. Hit rates with the distributed location mechanism deteriorate in large-scale systems.

A set of experiments with large-scale systems (with a random connection model) measured the average *HitRate*. As appears in Figure 5, there is a clear degradation in the hit rate as system scale increases. From this we can conclude that for larger system scales, one might consider changes in the location search strategy such as increasing search horizon or adding more contacts to the connection model. These changes will trade off communication overload for service quality.

3.3.4 Effect of the connection model (set 4).

For the fourth set of experiments, we have constructed configurations with 5 different connection models and search horizon (Max_TTL) values, keeping other parameters constant. The connection models we checked were grid2, grid4, random2, random3 and random4. Here, the number attached to the name of the model refers to the size of the agent's contact list or number of neighbors for each agent. The value of Max_TTL was varied from 2 to 6. Other parameters were set as follows: system scale was set to 49 agents; workload was set to high; the CD was set to 30%.

In these experiments we measured the response times only for the distributed location mechanism, as the connection model is irrelevant in the case of a centralized location mechanism. The results of these measurements are presented in Figure 6.

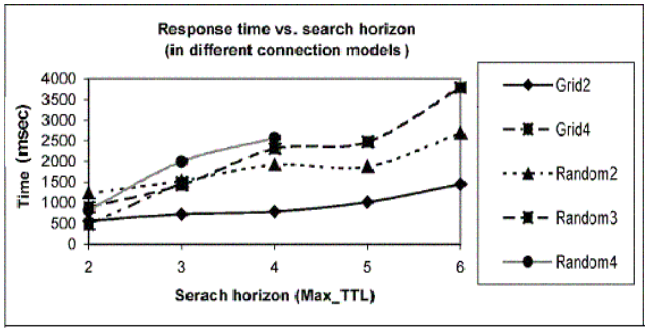

Figure 6. Response times of the random connection model are consistently higher than those of the grid models.

The outrank of the connection graph is the most important factor in affecting the growth rate of the message count. An outrank of 2 neighbors provides satisfactory results for most *TTL* values, while maintaining low overhead of messages.

The results of this experiment set suggest that the distributed location mechanism performs more efficiently with an ordered connection model than it does with a random model, although the communication overhead is higher in this former case. From multiple experiments we conducted, we learned that one should use small sizes of contact lists (2-3) to prevent exponential growth in the communication overhead in the distributed model. Even a contact list of size 4 must have a limited search horizon to prevent high communication overhead.

3.4 Multiple Hosts Experiments

In this section we describe the second phase of experiments, conducted on a large scale (up to 1500 agents) multi-host system, consisting of 10-15 hosts (some of the hosts were located in different LAN segments). This phase of experiments examines the results obtained in the previous phase in a larger, more realistic environment. We balanced the number of agents on each host by grouping them in system segments, although their contacts were deliberately chosen to be from other system segments.

3.4.1 Effect of system scale in multiple host MAS.

In this set of experiments, we varied system scale, keeping other parameters constant. The scales we checked were 200, 300, 400, 500, 700, 1000 and 1500. The agents were organized in segments of 100 agents, each segment deployed on a different host. The other parameters were set as follows: system workload was set to high; the connection model was a random model with 2 neighbors from another segment and Max_TTL = 4; the CD was set to 30%.

In these experiments we measured the response time (RefTime) for the distributed location mechanism, denoted by DrefTime, and for the centralized location mechanism denoted by CRefTime. The results of these measurements are presented in Figure 7. Note that beyond 500 agents, the message queues of the centralized location mechanism were extremely overloaded, and the simulation stagnated. Under the same conditions, the distributed mechanism functioned properly with 1500 agents, at reasonable response times. Larger scales were not examined because of limited memory resources: we could not run more than a 100 agents on each host, and we had only 15 hosts available.

As appears in Figure 7, the response times of the centralized location mechanism are significantly higher than those of the distributed location mechanism, and the centralized mechanism stagnates for medium to large systems. Even for smaller systems, the centralized mechanism exhibits a steep increase in response time. This indicates that this mechanism is very sensitive to scale. The system's scale affects the distributed location mechanism too, however this effect is moderate, which implies that the mechanism is scalable beyond the sizes examined here.

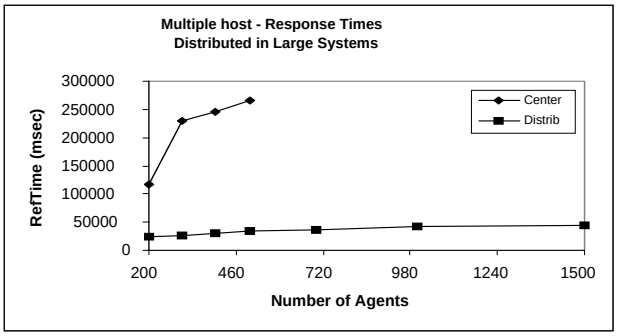

Figure 7. The central location mechanism exhibits consistently higher response times than the distributed location mechanism does in multiple host systems.

3.4.2 Effect of connection model in multiple host MAS.

In this set of experiments we also studied the behavior of response times and message count vs. different connection models and search horizons in a large-scale system (1000 agents deployed on 10 hosts). We used a random connection model (with 3 variations: 2, 3 and 4 contacts per agent) and allowed a range of search horizons. Note that implementing a grid connection model in a large system distributed across multiple hosts is cumbersome, and therefore we do not examine such a configuration. The results of the response times is not presented here for brevity however the message count results appear in Figure 8.

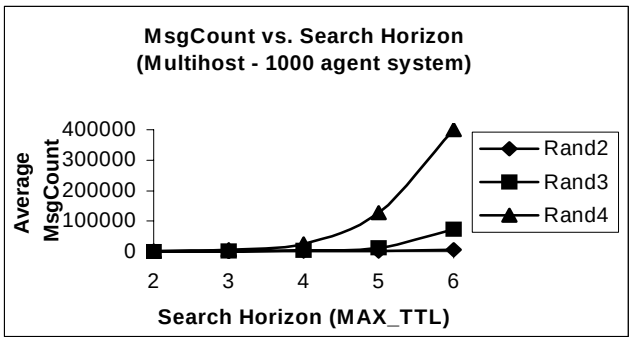

Figure 8. Random connection models achieve exponential message counts in large-scale multihost systems (The search horizon and outrank cause steep increase in communication overhead).

As seen in Figure 8, increasing the search horizon results in a exponential increase in message counts; this phenomenon occurs in all of the variations of the connection model checked. Thus, achieving better hit rates comes at the expense of a steep rise in communication overload. The effect of the outrank of the connection model on the response time is relatively minor, whereas it has a prominent effect on communication traffic.

In summary, the experiments presented in this paper systematically compare agent location mechanisms. The results provide valuable insights on performance and accuracy tradeoffs when such mechanisms are implemented in open large-scale MAS.

4. RELATED WORK

The MAS research community has performed several studies on agent location mechanisms. That research introduced mechanisms which are mostly centralized [4, 6, 11, 19]. While Decker et al. [4] introduce middle agents for web-based MAS and Wong and Sycara [19] describe different types of middle agents in MAS, Kuokka and Harada [11] suggest the use of a matchmaker mechanism and discuss its structure and merits, while recognizing that it might not scale well. Jha et al. [9] introduced a distribution of matchmakers and proposed a formal analysis of an algorithm for coordination among them. A fully distributed location mechanism was described in [13] and was analyzed theoretically. In [2], the authors evaluate location mechanisms, however they do so for fairly small systems, and without network distribution.

Similar ideas of distributed resource discovery in the Internet suggested, and new network models such as small-world [18] and scale-free networks [1], in line with our results, suggest that scalability of agent location mechanisms can be improved by using proper connection models among the agents and by eliminating central bottlenecks.

In a wider context, the theme of service location (or service discovery in different terminology) in distributed systems and networks was widely studied and many different mechanisms and protocols were suggested in that context. Many of the distributed models were inspired by ideas and successes of new communication models such as the emerging peer-to-peer [3, 16] and GRID [8,14,15] technologies.

These studies try to bring distribution of location and discovery mechanisms into GRID and agent systems and show the potential merits of these methods. Our study adds an experimental evidence to the potential benefits of these mechanisms in large-scale systems. We have seen relatively few studies ([8,17]) that emphasize simulation with similar conclusions, however these have focused on a specific framework and did not compare systematically the performance of distributed and centralized location mechanisms for large-scale MAS as we do.

Iamnitchi et al. [8] survey the resource discovery alternatives in very large scale systems (GRID) and recommend (without providing results for centralized mechanisms) introducing peer-to-peer discovery methods over centralized ones for improved scalability and fault-tolerance. Smithson and Moreau [14] and Srinivasan *et al.* [15] argue for a similar move in MAS for obtaining similar objectives. Vitaglione et al. [17] studied the effect of using agent-to-agent methods in the JADE agent framework. Gibbins and Hall [7] studied query routing services in MAS and arrived at conclusions similar to ours regarding the possibilities of fully distributed discovery, although they analyzed the problem theoretically and did not backup it up with experimental data.

Koubarakis [10] surveys the potential advantages of synergizing the benefits of peer-to-peer methods and ideas into MAS research and suggests a solution that involves distributed caches (called also DHT – Distributed Hash Tables) and super-peer solutions adopted in MAS context. A similar work is described in more detail by Dimakopoulos and Pitoura [5]. Their solution arrives at results better than our approach provides, however it requires a more complicated protocol to be embedded in each agent. Our work tried to keep the evaluated mechanisms as simple as possible, to isolate the effects of distribution on scalability. In addition to showing the advantages of distributed mechanisms, the results reported in [5] demonstrate the tradeoff between the complexity of the mechanism its performance.

Another complementary work of interest is that of Ogston and Vassiliadis [12], which suggests a sophisticated mechanism for querying neighbors and forming clusters of agents to replace the matchmaker function. Their simulation focused on the hit rate of the agent queries and the emergent behavior of the overlay network, while our work is focused on response time and message count measurements.

5. CONCLUSION

We have evaluated and compared distributed and centralized agent location mechanisms. For this we have developed a large scale MAS simulation testbed and performed a series of experiments in which both types of location mechanisms were examined and matched. Our major conclusions are as follows. Firstly, the response time of a distributed location mechanism is significantly better than the response time of a centralized one, in particular for large scale MAS. This result does not hold, however, in capability-deprived MAS, where a centralized mechanism will perform better. Secondly, it is evident that a centralized location mechanism is very sensitive to workloads. At a medium to high load, in particular in large MAS, the centralized mechanism will perform poorly, whereas the distributed one will hardly be affected.

In summary, it appears that a distributed agent location mechanism is the appropriate solution for large scale MAS with high workloads. Yet, one should recall that the advantages of the distributed solution do come at the cost of a communication overhead. This overhead may pose a severe problem. However, as our results suggest, a careful design of the connection model can reduce this risk.

Our study is a first step towards a better understanding of the advantages and drawbacks of agent location mechanisms. In particular, in future work we intend to examine mixed mechanisms in which agents can decide whether to use peers or to use middle agents to find other agents. We believe that such mechanisms have the potential to perform better than both the pure-centralized and pure-distributed solutions.

6. REFERENCES

[1] Barabasi, A.L., Albert, R., “Emergence of Scaling in Random Networks”. *Science*, page 286(509), 1999.

[2] Ben-Ami, D., Shehory, O., “Evaluation of Distributed and Centralized Agent Location Mechanisms”. LNAI, vol. 2446, pages 264-278, Springer, 2002.

[3] Clarke I., Sandberg O., Wiley B., Hong T.W., "Freenet: A Distributed Anonymous Information Storage and Retrieval System”. Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, 2000.

[4] Decker K., Sycara K., Williamson M. “Middle-Agents for the Internet” .Proceedings of IJCAI-97, pages 578-583, Nagoya Japan 1997.

[5] Dimakopoulos V.V., Pitoura E., "A Peer-to-Peer Approach to Resource Discovery in Multi-agent Systems". Proceedings of. CIA 2003: pages 62-77.

[6] Genesereth M., Ketchpel S., “Software Agents”. Communications of the ACM **37(7):** 48-53, July 1994.

[7] Gibbins, N. and Hall, W. “Scalability Issues for Query Routing Service Discovery”. Proceedings of the Second Workshop on Infrastructure for Agents, MAS and Scalable MAS (2001), pages 209-217.

[8] Iamnitchi, A., Foster, I., Nurmi, D., “A peer-to-peer approach to resource discovery in grid environments”. High Performance Distributed Computing, IEEE, Edinbourgh, UK, July 2002.

[9] Jha S., Chalasani P., Shehory O., Sycara K. “A formal treatment of distributed matchmaking”. Proceedings of Agents-98, pages 458-457 , Minneapolis, Minnesota, 1998.

[10] Koubarakis M., "Multi-agent Systems and Peer-to-Peer Computing: Methods, Systems, and Challenges". Proceedings of. CIA 2003 pages 46-61.
[11] Kuokka D., Harada L., “Matchmaking for information agents”. Proceedings of IJCAI-95, pages 672-679, 1995.
[12] Ogston E., Vassiliadis S., "Matchmaking Among Minimal Agents Without a Facilitator". Proceedings of the 5th International Conference on Autonomous Agents, pages 608-615, 2001.
[13] Shehory O., “A scalable agent location mechanism”. LNAI Vol. 1757 (ATAL-99), pages 162-172, Springer, 2000.
[14] Smithson A., Moreau L., “Engineering an Agent-Based Peer-To-Peer Resource Discovery System”. In Gianluca Moro and Manolis Koubarakis, editors, First International Workshop on Agents and Peer-to-Peer Computing, pages 69-80, Bologna, Italy, July 2002.
[15] Srinivasan N. et al., ”Enabling Peer-to-Peer Resource Discovery in Agent Environment”. Proceedings of Challenges in Open Agent Systems (AAMAS 2002), July 2002.
[16] Stoica I., Morris R., Karger D., Kasshoek M.F., Balakrishnan H., “Chord: A scalable peer-to-peer lookup service for Internet Applications”. Technical Report TR-819, MIT, March 2001.
[17] Vitaglione G., Quarta F. and Cortese E.. “Scalability and Performance of JADE Message Transport System”. Proceedings of the AAMAS Workshop on AgentCities, Bologna, 2002.
[18] Watts, D.J., Strogatz, S.H, “Collective Dynamics of ‘Small World’ Networks”. Nature, 393: pages 440-442, 1998.
[19] Wong H. C., Sycara K., “A taxonomy of middle-agents for the Internet”. Proceedings of ICMAS-00, pages 465-466, July 2000.
[20] Yolum P., Singh M.P., “An Agent-Based Approach for Trustworthy Service Location”. Proceedings of the 1st International Workshop on Agents and Peer-to-Peer Computing, Bologna, Italy 2002.
[21] Yu B., Singh M.P., “A Social Mechanism of Reputation Management in Electronic Communities”. Proceedings of CIA-00, pages 154-165, 2000.