On the Performance of Distributed Search by Mobile Agents

A. Mawlood-Yunis[1], Amiya Nayak[2], Doron Nussbaum[1], and Nicola Santoro[1]

[1] School of Computer Science, Carleton University, Ottawa, ON K1S 5B6, Canada
{armyunis,nussbaum,santoro}@scs.carleton.ca
[2] School of Information Technology & Engineering, University of Ottawa
Ottawa, ON K1N 6N5, Canada
anayak@site.uottawa.ca

Abstract. When using mobile agents in distributed search, the overall performance is influenced by several factors. In this paper, we study some of them and their impact. Several experiments were carried out to study the impact of network size, network topology, and the number of agents. These experiments were performed on two well-known mobile agent platforms: AGLET (a commercial environment), and TACOMA (an academic research environment). The results shed some light on the nature of the functional dependency of performance on these factors. Moreover, the experiments show that, except for the scale, the results are platform independent.

1 Introduction

The performance aspects of the mobile agents have been a subject of several studies. In [2] the scalability of the server, the performance factor that are fundamental to the mobile agent idea, and separately, the performance factor due to design differences of different mobile agent platforms have been addressed. The authors compared a traditional client/server approach with a mobile agent approach on four mobile agent's platforms in the context of a simple informational retrieval application. In [7], analysis of mobile agent performance in a network management and, its comparison with the client/server model used by the SNMP (Simple Network Management Protocol) have been performed. Performance evaluation of mobile agents for e-commerce application have been studied in [4]. In [3], an analytical model that examines the claimed performance benefits of mobile agents over client/server computing for a mobile information retrieval scenario was developed.

In this paper, we study performance behavior of mobile agents in a distributed search using single and multiple agents. In the case of multiple agent search, we suggest three different algorithms. The search has been conducted under variable network size, network topology and number of agents. Several experiments are carried out to measure the performance of mobile agents solution to distributed search problem. The results on two different mobile agent platforms, namely AGLET (a commercial one) [6] and TACOMA (an academic one) [5], have been compared.

2 Distributed Search with Single and Multiple Mobile Agents

2.1 Structure and Components

Since our objective is to compare the results using AGLET mobile agent with the results obtained by [8] using TACOMA mobile agent, we use similar algorithm to the one implemented in [8]. We start with the construction of a logical spanning tree on top of the physical network. Once a spanning tree constructed, the search problem is reduced to a tree traversal problem. The search time of mobile agent does not include the time to construct the spanning tree, as we assume that the spanning tree was constructed in pre-processing steps. To traverse the spanning tree, agent may start at any node in the network. It obtains its neighboring information and travels to the first host in the list. Once the agent obtains the neighboring information, it changes its internal queue or stack (itinerary). The process of adding neighboring information into the internal data storage in *Single agent* search leads to different spanning tree traversal such as *Breadth_First_Search* (BFS) once the the internal data structure is queue or *Depth_First_Search* (DFS) once internal data structure is stack. The search can be carried out with multiple agents; this we call *flooding.* Agent interacts with the local resources at each node; it opens a file at each site it visits, and carries along the contents of the file. Once all nodes in the network were visited, the agent returns to the initiator node with the search result (file contents).

In our study, we have used a framework that consists of the following components: *Blackboard*, *Whiteboard*, *Router*, and *Log.* The purpose of the *Blackboard* is to help terminate the flooding algorithm. The purpose of the *Whiteboard* is to avoid multiple visit to the same node by different agents. The purpose of the *Log* component is to terminate the flooding algorithm without using the *Blackboard* component. A *Log* component is associated with each node in the network. The *Router* is used with both single and multiple agent search. It acts as a global and local *Router* to enable agents to move around the network.

2.2 Search Using Single Agent

At each host, the agent gets the list of the next hosts to visit which becomes the agent's *Itinerary.* The agent checks to see if a parent of the current host or initiator host are in the list. If the list contains the parent or initiator node, the agent removes it to avoid returning to the host it just came from or returning to the the initiator node before completing the search. It adds the remaining hosts (if not null) to the internal container. For the BFS traversal, every time we get a set of new neighbors from the *Router* we add it to the end of the internal container, this results in BFS traversal. In case of DFS, we add a new set of neighbors to the beginning of the internal list resulting in DFS traversal.

2.3 Search Using Multiple Agents (Flooding)

Flooding is another way to traverse spanning tree and solve the problem of distributed search. The agent starts from a node and duplicates itself as many identical agents as the number of children specified in the local number list; all these copies travel simultaneously to every child in the neighboring list. For the purpose of flooding the network with agents we use three different approaches: *Local flooding*, *Remote flooding*, and *Regular flooding*

In *Remote flooding*, the agent and the *Blackboard* communicate remotely. Here the agent contacts *Blackboard* for removing its ID once it is at the end of the search path and just before moving back to the initiator node. In this way, the agent removes its ID while being host on remote node, and only after its ID has been removed from the *Blackboard* it returns back to the initiator node.

In *Local flooding* the agent and *Blackboard* communicate locally on the initiator node. This contact happens just before agent reports its partial result and after it returns back to the initiator node. The agent contacts the *Blackboard* to remove its ID. Once its ID is removed, it reports the partial result to the initiator node.

In *Regular flooding*, agents start moving from the initiator node all the way down to the end of their search path. Once they reach the end of the search path, each agent reports its partial result to its immediate parent. This continues until the final result reaches the initiator. Once the last agent returns back to initiator, the algorithm terminates. In this approach, we do not have any *Blackboard* component to facilitate termination, as it is the case with two previous algorithms. Instead, we use a *Log* (i.e. local *Blackboard*) at each node which holds information about that node.

3 Experimental Results

In the experiments, agents are injected from any node of the network. The launching application or initiator measures the time in milliseconds before injecting the first agent and after the last agent has arrived. The difference between these two time measurements is the total execution time for the entire search. The strategy followed for testing different schemes consists of a set of test cases aimed at evaluating the impact of certain variables on the overall performance. The objective was to obtain sufficient information to compare the performance between the AGLET mobile agent and the TACOMA mobile agent.

All the experiments were carried out in the absence of failure. The computers were used simultaneously by other users, and no special care was taken to guarantee exclusive access to computer resources or network during these experiments. All the experiments presented herein were developed in the Graduate Lab of the School of Computer Science on 1000 MHz Pentium with 500 MB RAM machines. The agent platform used was AGLET-2.0.1 compiled and run on the Sigma Network (Linux Mandrake 7.1 machines) and the Ultra Network (SunOs 5.7 Generic_106541-06 sun4u sparc SUNW, Ultra5-10). The *Blackboard* associated to each node was implemented as a multi-threaded application using the Java JDK1.2.2. *Router* and *Whiteboard* were implemented in Java JDK1.2.2 and used a RMI server. The size of information carried by the agent was same for all algorithms. Agent(s) performed a dummy search at each site consisting basically of opening a file and reading the content.

3.1 Impact of Network Size

We compared the search time of running single and multiple agents on binary trees of different sizes from 3 to 28 nodes. Each test was run thirty times to get the average search time. The following results were obtained:

1. The AGLET agent performed better than the TACOMA Agent. Searching with a single AGLET agent is almost ten times faster than searching with a single TACOMA agent. In multiple agent search, the AGLET agent performed almost seven times better than the TACOMA agent. This is due to the difference in their respective platforms.
2. For both the AGLET and TACOMA platforms, single agent performed better than multiple agents for small networks whereas multiple agents performed better than single agent in large networks. The only difference here is in the size of the network in which multiple agents outperformed the single agent. In the TACOMA platform, the single agent performed better than multiple agents for a network size up to eighteen nodes, while in the AGLET platform network size of only eight nodes were sufficient for multiple agents to outperform a single agent. This result suggests that we can run multiple AGLET agents on binary tree (i.e. take advantage of parallelism) on network with size as little as eight nodes, whereas we cannot do that with TACOMA agents. In order to take advantage of parallelism with TACOMA agents, we should have a network (i.e. binary tree) size of at least eighteen nodes.
3. The single agent search performance is better than the multiple agents search performance for networks of small size.

The above explanations lead us to conclude that for small networks, the advantages we gain from parallelism by using multiple agents is not enough to overcome the overhead associated with it. In large networks, the advantages of parallelism will overcome the above mentioned overhead, and as a result we get a better search performance with multiple agents than with single agent. Another aspect of the third point of the result is that in multiple agents search, the AGLET system performs better than TACOMA system in small networks. The execution time for a single and multiple travelling agent for both AGLET and TACOMA system are shown in Figure 1 and Figure 2 respectively.

3.2 Impact of Number of Neighbors

In this test, we compared the results of running single and multiple agents of both TACOMA and AGLET systems on various trees with constant size. The trees are differ from each other in the number of children of each node. We started with a binary tree and then increased the number of neighbors for each node at each step by two until we ended up with a star structure.

The result shows that in the case of a single agent search, the number of neighbors did not have significant impact on search time in both systems. This result is predicted, since by increasing the number of neighbors in single agent

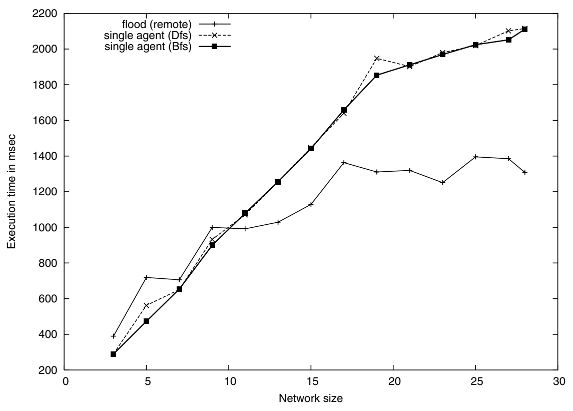

Fig. 1. Average Execution Time for Single and Multiple Agent Search (AGLET)

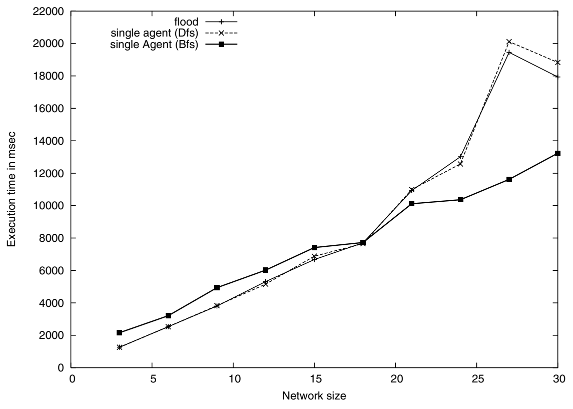

Fig. 2. Average Execution Time for Single and Multiple Agent Search (TACOMA)

search the change occurs only in the visiting sequence of hosts, and this does not have any effect on the search since we working on LAN. For both systems, multiple agents search performed better than single agent search when the number of neighbors was low. As the number of neighbors increased and the search became closer and closer to the sequential search, single agent performed better than multiple agent. This was due to the overhead associated with multiple agents search.

The only difference between AGLET and TACOMA agents here is at the turning point when single agent search starts to become more efficient than the multiple agents search. In the TACOMA system, multiple agents search performed better than single agent search only in the case of binary tree, whereas in AGLET system, multiple agents search performed better than the single agent search in a tree with each node having eight neighbors (k =8) and the best performance came in a tree with each node having four neighbors (k= 4).

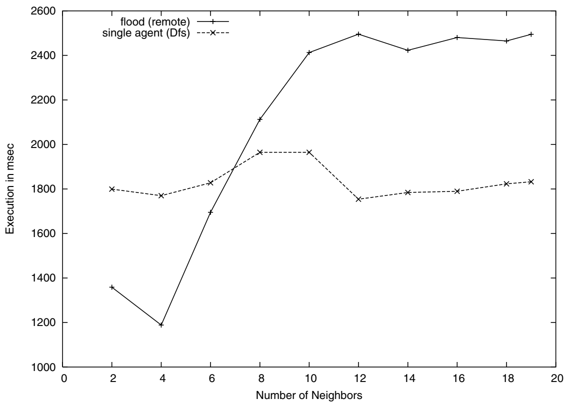

Fig. 3. Single and Multiple Mobile Agents with Various Number of Neighbors (AGLET)

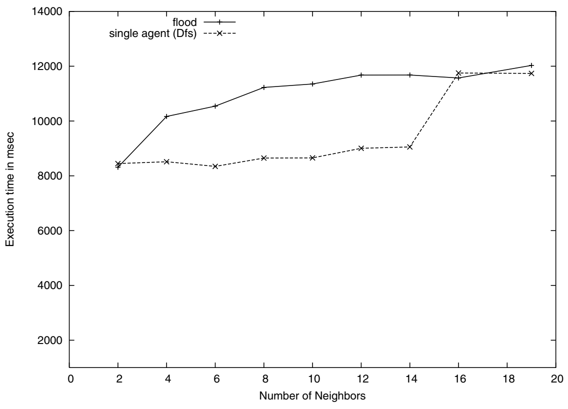

Fig. 4. Single and Multiple Mobile Agents with Various Number of Neighbors (TACOMA)

Figures 3 and 4 show the average execution time for a single and a multiple agent for both the AGLET and the TACOMA system when the network size is constant (N = 20) and the topology changes from a binary tree to a star (k =2, 4, 6, . . . , 19).

3.3 Impact of Network Topology

In this test, we compared the search times of running single and multiple AGLET and TACOMA agents in different network topologies of a fixed size. The network for this test consists of sixteen nodes. The topologies which been used for this test are: unidirectional ring, bidirectional ring, hypercube, and binary tree. Beside comparing the search times of AGLET and TACOMA, we also compared the search time of single and multiple agents for the AGLET system as well. We have the following observations:

Single Agent Traversal:

a. The best performance is seen in the bidirectional ring and the worst performance in the hypercube for both the AGLET and the TACOMA systems. Hence, the behavior of both systems is very much similar in different topologies when a single agent is used.
b. Even though we see the best and worst scenario, the performance in all topologies considered is almost the same for both systems.

Multiple Agent Traversal:

a. For deploying multiple agents, the best performance of the TACOMA agent is seen in the bidirectional ring, and the performance is similar in the case of a binary tree. On the other hand, the best performance of the AGLET agent is seen in the hypercube. From the first test, it is evident that the AGLET agents are more efficient than the TACOMA agents; hence, the AGLET agents are more capable of taking advantage of parallelism. The second test shows that the best performance of AGLET agents comes from a tree with four neighbors for each node or a hypercube.
b. The worst performance for the AGLET agent is seen in the unidirectional ring whereas the worst performance for the TACOMA agent is seen in the hypercube.

Single and Multiple AGLET Agent Performance:

The result shows that the AGLET performance is better in the flooding case than in the a single agent case for both binary tree and hypercube, but it is almost the same for bidirectional ring. In the case of bidirectional ring, the single agent performs slightly better than flooding. This is due to the overhead associated with flooding. If we ignore the extra overhead associated with flooding, the single agent search is better in the case of a bidirectional ring.

Figure 5 shows the performance comparison of AGLET and TACOMA mobile agent in single agent search for different network topologies. Similarly, Figure 6 compares the result in the case of multiple agents.

3.4 Three Flooding Algorithms

In this test, we compared the performance of *local flooding*, *remote flooding*, and *regular flooding* in binary trees of different sizes from 3 to 28 nodes. The following observations were made:

1. The difference between all three approaches is very little if the network size is small.
2. Once the network size exceeds nineteen nodes, *remote flooding* performs better the other two algorithms.

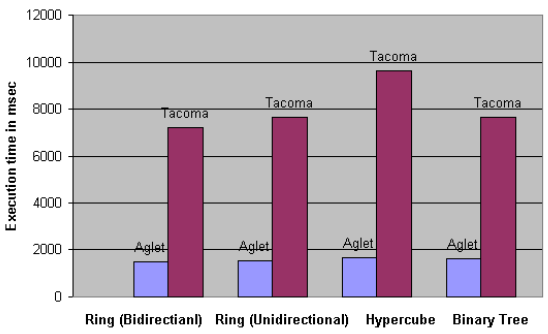

Fig. 5. Performance of AGLET and TACOMA in Different Network Topologies (Single Agent Case)

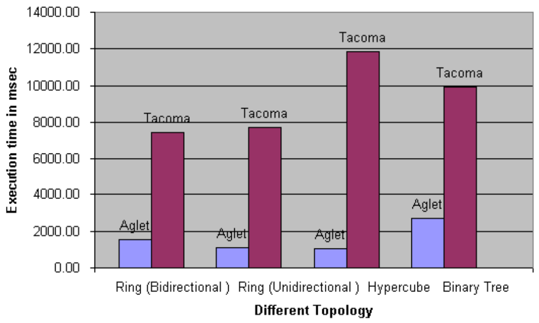

Fig. 6. Performance of AGLET and TACOMA in Different Network Topologies (Flooding Case)

3. The *remote flooding* performs better than *regular flooding*, even though we do not contact the *Blackboard* remotely nor do we have multiple agents running at the same place at one time.

Figure 7 shows the execution time for all three algorithms.

3.5 Cost of Moving AGLET and TACOMA Agents Between Nodes

In this test, we compared the time required for moving a single AGLET and TACOMA agent between any two nodes. The objective was to compare the time required in each case to deploy an agent to a node and to pull back this agent to the source. The result shows that even though the difference in the implementation language is in favor of TACOMA, the AGLET mobile agent is almost ten times faster than the TACOMA agent (162.32 msec in the case of AGLET compared to 1551.57 msec in the case of TACOMA). The difference in performance is due to the difference in their respective architectures. In the AGLET mobile agent case, we compile the agent code to byte code, and when we deploy the agent we serialize the byte code and send it over the socket to

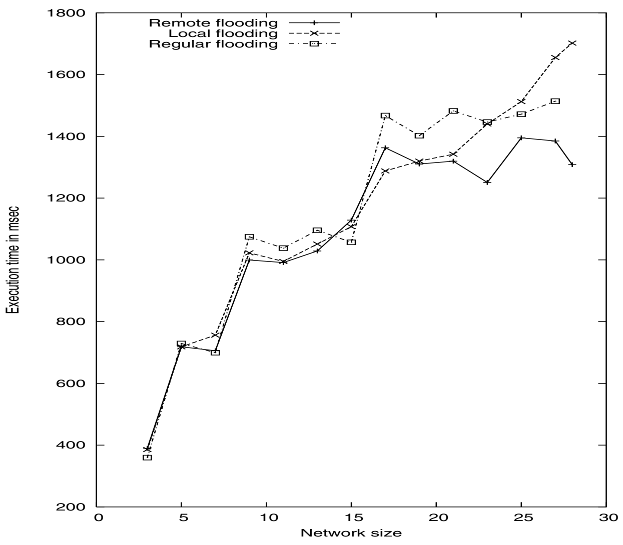

Fig. 7. Execution of remote, local and regular flooding for various network size

be executed at the remote host. In the TACOMA case, on the other hand, once we deploy the agent we send the code which should be compiled and linked for execution at the remote host. This requires more time than having byte code.

4 Conclusions

In all the tests we conducted, single agent performed better than multiple agents in networks of small size, whereas in large networks, multiple agents outperformed single agent. The same result was seen in the TACOMA platform; the only difference between the AGLET and the TACOMA is in the size of the network in which multiple agents outperformed single agent. We identified the crossover points for all our tests which could be used to make decision on the choice of mobile agent platform, search criteria, topology, etc. Comparing the results from both AGLET and TACOMA platforms, it appears that the overall behavior of a mobile agent is platform independent whereas its performance is platform dependent; this is the same conclusion that was found in [1].

References

1. J. Baek, J. Yeo, G. Kim, and H. Yeom. Cost Effective Mobile Agent Planning for Distributed Information Retrieval. *Proc. 21st Conference on Distributed Systems*, pp. 65-72, 2001.
2. R. S. Gray, D. Kotz, R. A. Peterson, J. Barton, D. Chacón, P. Gerken, M. Hofmann, J. Bradshaw, M. Breedy, R. Jeffers, and N. Suri. Mobile-Agent versus Client/Server Performance: Scalability in an Information-Retrieval Task. *Proc. 5th International Conference on Mobile Agents*, pp. 229-243 , 2002.

3. R. Jain, F. Anjum and A. Umar. A Comparison of mobile agent and client-server paradigms for information retrieval tasks in virtual enterprises. *IEEE Academia/Industry Working Conference on Research Challenges*, pp. 209-215, 2000.
4. R. Jha, and S. Iyer. Performance Evaluation of Mobile Agents for E-Commerce Application. *Proc. International Conference on High Performance Computing*, pp. 331-341, 2001.
5. D. Johansen, R. V. Renesse and F. Schneider. An Introduction to the TACOMA Distributed System Version 1.0. Technical Report 95-23, Department of Computer Science, University of Tromso, Norway, 1995.
6. D. B. Lange and M. Oshima. Programming and Deploying Java Mobile Agents with Aglets. Addison Wesley, Massachusetts, USA, 1998.
7. M.G. Rubinstein, O. Carlos, M. B. Duarte and G. Pujolle. Scalability of a Network Management Application Based on Mobile Agents *Proc. 2nd International IFIP-TC6 Networking Conference*, pp. 515-526, 2002.
8. E. Velazquez, N. Santoro, A. Nayak. A Mobile Agent Prototype for Distributed Search. *Proc. 3rd Int. Workshop on Mobile Agents for Telecommunications Applications*, pp. 245-254. 2001.