1997 International Conference on Computer
Systems Technology for Industrial Applications--
Internet and Multimedia

CSIA '97
Proceedings

Intelligent Search of Mobile Agents §

Yao-Nan Lien*, Fuhan Liu*, Wen-Shyan Chen**, Chun-Wu Leng

* Department of Computer Science, National Chengchi University
** Institute of Computer Science, National Chung-Hsing University

Abstract

In a mobile computing environment that supports mobile agents, a client can send an agent to visit a sequence of servers in the network. Tracking the location of agents becomes a critical problem in managing a mobile agent service network. In this research, we studied the performances of a number of intelligent search strategies that make use of prior knowledge about the possible service time each server may take. Our study found that, under a variety of circumstances, the mean service time is a reliable index to predict the location of a mobile agent even if the service time is probabilistic.

1. Introduction

1.1 Agent and Agent Mobility

The goal of a ubiquitous information service network is to provide information to the users any time anywhere [7]. To accomplish this goal, a service network must be supported by a wireless communication network and be able to conveniently access to various information resources [9].

Due to the immaturity of distributed computing technology, clients have to access network resources in a prescriptive fashion by interacting with individual servers probe by probe to accomplish a complicated task. However, in most mobile computing environments, the nature of communications is intermittent and the battery energy is limited. Thus, it is very difficult to accomplish a complicated task that requires its client to interact with multiple servers intensively. A non-traditional computing paradigm, the *intelligent messaging*, which allows clients to interact with multiple servers in a dynamic fashion has been brought up to cope with this problem [1,2,3,6,8].

Simply speaking, an *intelligent message* is an electronic message that carries a computer program, whether procedural or declarative, which can be executed by the receiving servers on behalf of the originating client. The program in the message can also instruct a receiving server to forward automatically the message itself to another server, on which the program is executed continuously in a pipeline fashion. For simplicity, it is called a *mobile agent*, or *agent* in the rest of this paper. Good examples can be found in [3].

Since an agent may be traveling in a service network, the originating client may not be able to trace or control its operation directly. A service network must provide some mechanisms allowing its clients to trace and control these messages. This problem is referred to as the *agent mobility management*.

1.2 Mobile Agent Service Networks

1.2.1 Open service network architecture

Traditional telecommunication networks such as PSTN and 800 Toll-Free service used to take considerable resources and long deployment duration to establish. One major resource drain in such networks is the OA&M (Operation, Administration, and Maintenance). It will be impractical to demand the comparable resources to support the OA&M functionalities in many prospective information services. Thus, the computing community have to develop and deploy the demanded functionalities by themselves. All infrastructures and solutions must not require any change to the existing telecommunication network. To achieve this, we would employ the open service network architecture proposed in [3], which separates service networks from transport networks. It also allows services of any scale and any quality to be introduced into the network easily. Readers are referred to [3] for details.

On the top of the open architecture mentioned above, Lien proposed a hybrid operation mode in [2] that allows a service provider to offer both centralized and distributed operation modes to its subscribers. Subscribers can choose to use their own Internet facility, called *Home Base Node* (HBN), to share the OA&M functionalities. (Typical OA&M functionalities are billing, client location register, agent tracking and control, agent status holder, etc.) At their own expense, subscribers can also designate some OA&M functionalities to the centralized facility managed by the service providers. In this paper, we assume that a service network that supports mobile agents is established based on the proposed open architecture and managed using the proposed operation infrastructure.

§ This work is partially supported by ITRI/CCL grant under MOEA Distributed Information Processing Program (86-EC-2-A-17-0204).

1.2.2 Searching a mobile agent

After an agent is submitted into a service network, the client or the network manager may need to know its current location in order to inquire its status, or to control its execution, etc. A simple way is to send another agent, called *search agent*, to search the original agent along the original path, or to broadcast a message to every server where the agent might have visited. There are some problems associated with these straightforward solutions:

1. Sending many messages over a wireless network may be too expensive.
2. The path that an agent traveled along may be non-deterministic such that it is difficult to trace.
3. A sequential search may take too much time.

Therefore, better ways to locate an agent are in need. In [4], Lien and Leng proposed and analyzed several search strategies, which will be described in next section. Among the proposed strategies, the intelligent search is the most promising one if the service time of each individual task can be estimated using prior statistics. To reduce the computation time that is required to estimate the best possible location of a target agent, we studied various ways based on the aggregated statistical properties to improve the computational performance of the intelligent search strategy (IBS) proposed in [4]. The rest of this paper is organized as follows: Section 2 will review the search strategies proposed in [4]; Section 3 will discuss the intelligent searches using the aggregated service time statistics to reduce the computation time. An adaptive search strategy will be presented in Section 4. Finally, Section 5 will discuss some complicated issues and future research directions.

2. Previous Work

The strategies proposed in [4] can be categorized into two types: blind and intelligent searches. The intelligent search strategies make use of prior knowledge about the execution of all tasks, while blind search strategies don't.

The following notations will be used in the following sections:

- $S = \{S_1, S_2, \ldots, S_n\}$: the set of distinct servers visited by an agent.
- T: the elapsed time since the target agent was originated.
- T_k: the time duration that the target agent stayed at server S_k. It is called the *service time* at server S_k.

In this paper, we assume that the target agent visits $\{S_1, S_2, \ldots, S_n\}$ sequentially and nonrecursively. Further, the time for the target agent to move from one server to another is considered nominal and is neglected.

2.1 Chase-from-holder Algorithms

In the hybrid operational infrastructure proposed by lien [2], users are encouraged to use their own HBNs to participate in the management of their agents. One possible usage of HBNs is to store the current status of agents, including agents' locations. Thus, the current reported location of an agent can be obtained right in the status holder (i.g. HBN) of its originating user. If the accurate current location is needed, the search agent can visit the reported location first and proceed with a sequential search right from there if the target agent had passed that server. This type of algorithms is called the *Chase-from-holder algorithms*.

Obviously, one major problem with the Chase-from-holder algorithms is the extra expense to update status periodically. The update cost has at least two major components: the induced network traffic and the resource consumption in the status holder. The system resources could be very expensive if the status holder is designated to the network management center. Thus, the trade-off between update cost and the status availability must be carefully balanced. Depending on its status inquiry frequency, a client may choose to command an agent report its status either completely or selectively. The following search strategies are useful when the current location of an agent is not available in its status holder.

2.2 Binary Search Algorithms

The *Basic Binary Search* algorithm (BBS) is similar to the binary search in searching a data object in a sorted list. The search agent probes the middle server in the search list and excludes half of the servers out of search list at a time. The search is performed recursively until the target agent is found or the list is exhausted. In average, the number of probes required to find the target agent is in the order of $log(n)$, where n is the number of servers in the search list. BBS might be a very good search strategy for blind search. However, BBS may fail to find the target agent if the target agent continues to move during the search. During the course of search, some unvisited servers may be exclude out of the search list after a server is probed. This may cause a *slip through* problem, which means the target agent slips through the search window so that the servers it visits afterward are not included in the search list. As a result, the search agent fails to find the target agent.

The *Extended Binary Search* algorithm corrects this problem by not excluding any unvisited server at the cost of demanding more search probes. Fortunately, the average number of probes required to find the target agent is still in the order of $log(n)$, only with a larger coefficiency.

Without a prior knowledge about the status of the target agent and servers, by intuition, the binary search is probably the best way (or close to the best) to search an agent. However, when there is a prior knowledge, other algorithms that take this advantage will be better than binary search algorithms.

2.3 Intelligent Search

If a client has a good estimation on the service time in each probe, he/she may have a good guess on the current location of an agent. By using this information in a search, it will be able to reduce the search time significantly. The term *intelligent search* here reflects the fact that these algorithms make use of service time statistics and thus, is non-blind.

We assume that an agent visits a set of servers, $S_1, S_2, \ldots, S_n$, in sequence and it stays at each server, says S_k, for a time duration of T_k. At any elapsed time T, the current location of the agent, S_c, can be determined by the following formula:

$$\sum_{i=1}^{c-1} T_i \le T < \sum_{i=1}^{c} T_i.$$

However, it is impractical to know in advance exactly how long the target agent will stay at each server. The service time in each server is most likely probabilistic and can be pre-estimated through either sample collection or experiments. As a result, the location of the agent is also probabilistic. To minimize the number of search probes, it is essential to calculate the location of the target agent with the highest probability, next highest, etc., so that a search agent can locate the target agent with the minimum number of probes. Assuming that the service time of each task is uniformly distributed, Lien and Leng derived a formula to calculate for each server the probability that a target agent might reside [4]. For simplicity, this probability is called the *residing probability*. In their proposed Intelligent Binary Search algorithm (IBS), the search list is presorted according to the calculated probabilities. Then, the search agent probes the servers in the search list sequentially. During a search, the search list is maintained in the following way:

1. If the target agent had visited and left the currently probed server, all servers that precede the current server in the traveling sequence are removed from the search list. (Traveling sequence is the visiting sequence the target agent is commanded to take.) The original search order remains unchanged. The search agent continues to search for the target agent according to the shrunken search list.
2. If the target agent has not visited the currently probed server yet, the search list will not be changed. However, the search agent will probe the server that precedes the current server (in the traveling sequence) and has the highest residing probability.

2.4 Comparison of Blind Search and Intelligent Search

Intelligent search algorithms are much better than blind search algorithms because they make use of prior knowledge about the service time of each task in each server. However, it requires the service time to be predictable. In other words, the algorithm assumes the target agent and the service network work normally without any exception. Thus, these algorithms have better be used under healthy operation conditions. They may not be appropriate to be used in handling exceptional cases. For example, if the target agent is found far behind the schedule, the client may send out a search agent to check the status of the first agent. Apparently, the intelligent search is not appropriate to deal with this situation since the prior knowledge is definitely useless. In this case, blind search strategies may be a better choice. It requires further study to determine the appropriate strategies under various conditions.

Another problem associated with the calculation of residing probability is its quadratic computation time for each individual residing probability. Even if the computation overhead can be compensated by using faster processors, the collection of the entire service time distribution from every server will still consume significant system and network resources. Furthermore, the distribution of service time may not be well formed as we expected. Therefore, paying extra computation overhead to gain a specious accuracy may not be worthwhile. In this paper, we propose to use aggregated statistical properties to predict the location of a mobile agent. After a careful study, we found that the result is encouraging.

3. Aggregated Statistics Based Prediction

In this section, we will investigate the performance of intelligent search using aggregated statistical properties over several different distributions, namely, normal, uniform, and skewed distributions. We assume that T_i is the service time of an agent on server i and $P(T_i)$ is the probability density function of T_i. The residing probability $P_r(k,T)$ is defined as the probability that the target agent is at server S_k on the elapsed time T. In other words, $P_r(k,T) = P(\sum_{i=1}^{k-1} T_i \le T < \sum_{i=1}^{k} T_i)$. (In order to make the summation of the residing probability of all servers equals to one, we can add a pseudo server to the end of server sequence with an infinite service time.) The server that has the highest residing probability with respect to a elapsed time T is called the *most probable server (MPS) of time T*. Without loss of clarity, it may be called the MPS for simplicity.

3.1 Normal Distribution

The first type of service time distribution we studied is normal distribution. Assuming that the service time of server S_i is normally distributed as $N(\mu_i, \sigma_i)$, where μ_i is the mean service time and σ_i is the standard deviation. According to the property of normal distribution, the accumulated service time $\sum_{i=1}^{k} T_i$ is also normally distributed, with a mean of $\sum_{i=1}^{k} \mu_i$ and a variance of $\sum_{i=1}^{k} \sigma_i^2$.

3.1.1 Calculating Residing Probability

The residing probability $P_r(k,T)$ can be calculated as follows:

$$P_r(k,T) = P\left\{\sum_{i=1}^{k-1} T_i \le T < \sum_{i=1}^{k} T_i\right\}$$
$$= P\left\{T < \sum_{i=1}^{k} T_i < \infty\right\} - P\left\{T < \sum_{i=1}^{k-1} T_i < \infty\right\}.$$

Let $Z_k = \dfrac{\sum_{i=1}^{k}(T_i - \mu_i)}{\sqrt{\sum_{i=1}^{k}\sigma_i^2}}$, we can transform $P_r(k,T)$ into Eq. (1) as follows:

$$P\left\{\frac{T - \sum_{i=1}^{k}\mu_i}{\sqrt{\sum_{i=1}^{k}\sigma_i^2}} < Z_k < \infty\right\} - P\left\{\frac{T - \sum_{i=1}^{k-1}\mu_i}{\sqrt{\sum_{i=1}^{k-1}\sigma_i^2}} < Z_{k-1} < \infty\right\}$$

The distribution of Z is then a standard normal distribution and these probabilities can be easily obtained from tables. For a given elapse time T, we can calculate the residing probability for all servers and select the MPS, the next MPS, and so forth in $O(n \log n)$ time.

3.1.2 Experiments

In order to verify this analysis, we proceeded with a simulation experiment using 20 servers. The service time for the target agent in each server is normally distributed with its mean and standard deviation shown in Table 1.

The elapsed time is set at 30. The MPS is then at S_{15} based on the calculation given by Eq. (1). Based on this statistic, compute the number of probes required by BBS (suming that the agent is not moving during search), EBS, and the IBS. As we can see in Table 2 and 3, the IBS is much better than BBS and EBS. Further, it only requires about 2 probes in average to reach the target agent.

From the experiment shown above we can draw the following conclusions when the service time of all servers are normally distributed:

Table 1. Execution time distributions of servers.

Server	S_1	S_2	S_3	S_4	S_5
Mean	2.10	1.50	3.20	1.40	2.50
Std. Dev.	0.312	0.215	0.705	0.414	0.430
Server	S_6	S_7	S_8	S_9	S_{10}
Mean	1.24	1.45	2.36	2.11	3.21
Std. Dev.	0.411	0.426	0.315	0.271	0.75
Servers	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}
Mean	2.25	1.21	2.36	1.22	1.20
Std. Dev.	0.412	0.317	0.375	0.425	0.425
Servers	S_{16}	S_{17}	S_{18}	S_{19}	S_{20}
Mean	1.50	3.70	2.80	3.14	2.21
Std. Dev.	0.503	0.302	0.210	0.310	0.220

Table 2. Simulation results based on normal distribution.

	S11	S12	S13	S14
P_r (calculated)%	0.05	0.36	10.21	19.45
Residing Freq.	1	169	848	2185
P_r (simulated)%	0.01	1.69	8.48	21.85
search probes (IBS)	5	4	3	2
Search probes (BBS)	4	3	4	5
Search probes (EBS)	4	3	6	5
	S15	S16	S17	S18
P_r (calculated)%	33.35	21.52	10.48	0.01
Residing Freq.	3452	2315	1054	3
P_r (simulated)%	34.52	23.15	10.54	0.03
search probes (IBS)	1	2	3	4
Search probes (BBS)	2	4	3	4
Search probes (EBS)	2	4	3	4

* The residing frequency of a server is the number of times the agent is located at that server in the simulation.
* The search probes of a server on a particular search algorithm means that the number of probes required to reach the target agent if the agent is at that server and that algorithm is used.

Table 3. Average number of probes required by BBS EBS, and IBS over 10000 samples.

Search Algo.	# of Probes
avg. probes by IBS (Calculated)	1.80
avg. probes by IBS (Simulated)	1.82
avg. probes by BBS (Simulated)	3.54
avg. probes by EBS (Simulated)	3.71

1 The theoretical calculation of residing probability based on the Eq. (1) is very accurate when compared to the simulated experiments.

2. The number of probes required to locate a mobile agent using the IBS is about half the number of probes required by using either BBS or EBS.

3.2 Uniform Distributions

When the service time of all servers are uniformly distributed, we can still use Eq. (1) to calculate the residing probability for all servers. By Central Limit Theory, for infinite number of servers, the total service time is still normally distributed if each individual service time is uniformly distributed. Although it is impossible to have a infinite number of servers, it is still applicable for a small number of servers within a tolerable error. Assume the accumulated service time $\sum_{i=1}^{k} T_i$ is normally distributed. According to the property of uniform distribution, its mean service time is $\sum_{i=1}^{k} \mu_i$ and its variance is $\sum_{i=1}^{k} \sigma_i^2$. The residing probability $P_r(k,T)$ can then be calculated by Eq. (1).

3.3 Skewed Distributions

Intelligent search algorithms are not only good for normal-like distributions, but may also be good for other distributions as well. It is a challenge to release unnecessary constraints. In this section, we will investigate the possibility of using IBS algorithm for other types of service time distribution. According to Central Limit Theory, the total service time is normally distributed regardless of the distribution of each individual service time if there are infinite number of servers. Again, we cannot expect an infinite number of servers in the real world. When the number of servers is small, although we could just assume it is normally distributed similar to what we proposed for uniform distribution, the estimation error may be too large to be useful. Therefore, we propose to use the following search strategy.

Assuming that the distribution of residing probability with respect to the server sequence is a normal distribution, (or convex shape), the execution of IBS will be as follows:

1. visiting the MPS, which is the mode value of residing probability distribution;
2. sequentially visiting the server next to the previously visited server forward or backward.

It is impossible to precisely calculate the MPS. However a blind guess may be better than nothing. We have proceeded with some experiments to study the performance of two different estimation methods in choosing a MFS:

1. method N_1: $MFS = S_k$ if $\sum_{i=1}^{k-1} \mu_i \le T < \sum_{i=1}^{k} \mu_i$ is satisfied;
2. method N_2: $MFS = S_k$ if $\sum_{i=1}^{k-1} \lambda_i \le T < \sum_{i=1}^{k} \lambda_i$ is satisfied;

where λ_i is the mode value of $P(T_i)$.

3.3.1 Experiments

We have proceeded with six experiments in this study. The service time of all servers in each experiment are i.i.d.. The service time distribution of experiment 4 is shown in Table 4. The mean service time is 2.39 time units and the mode value is 3 time units. The simulation results are summarized in Table 5. The results show that the performance of the first estimation method is surprisingly good. In two or three probes, IBS can locate the target agent.

Table 4 Distribution of service time in experiment 4.

[illegible]	0.25	0.5	0.75	1.0
[illegible]	[illegible]	1.5	2.25	3
time	1.25	1.5	1.75	2.0
Prob.(%)	2.25	3	4	4.75
time	2.25	2.5	2.75	3.0
Prob.(%)	6.25	8	10.25	25.5
time	3.25	3.5	3.75	4.0
[illegible]	[illegible]	10	3.75	1

Table 5 Simulation results for skewed distribution.

Experiment	T	probe # by N_1	probe # by N_2
1	5	1.34	2.46
2	10	1.56	2.55
3	15	1.65	2.56
4	25	1.84	3.62
5	50	2.10	4.78
6	75	2.30	6.47

4. Adaptive Binary Search

The search strategies mentioned are all static, which means that the search strategy is fixed when the search agent is launched to the network. For a particular search strategy, the search route is completely determined by the path taken by the target agent. They do not make use of other information, such as the departure time when the target agent left a server, to adjust the search sequence. One way to improve the proposed search strategies is to recalculate the residing probability, based on the departure time when the target agent left, each time after a search agent probes a server. For instance, if an IBS search agent finds out that the target agent has already visited and left the currently probed server, it will know that the estimation of the MPS is not accurate. The search agent can use this departure time information to recalculate the MPS. This may be better than blindly searching forward along the original planed path.

This dynamic technique may improve the performance of blind search strategies by a significant margin. This will be discussed in the rest of this section. (The performance of IBS has been already very good, it could hardly be benefited by this technique.)

We assume that every server maintains an execution log that stores various execution status including the departure time when the target agent moved to next server. By making use of this information, a search agent can choose the best search strategy dynamically. One possible adaptation is to have the search agent switch from EBS to the sequential search if the target agent "just" left the currently probed server not long ago. The algorithm is shown in the followings:

```
Adaptive Binary Search (ABS) Algorithm
 (1) SrhSet =  {S_1,S_2, . . . ,S_n}
 (2) if (SrhSet = empty)
 (3) then return (NOT_FOUND)
 (4) S_c = Middle Server in the SrhSet
 (5) if (target in S_c)  return (S_c);
 (6) if (target had visited S_c)
 (7) then {
 (8)      SrhSet =  {S_c+1, . . . ,S_n}
 (9)      if ( target left recently )
 (10)     then {
               while ( SrhSet is not empty) {
 (11)             S_c = head of SrhSet
 (12)             if (target in S_c) return(FOUND)
 (13)             SrhSet -= S_c
             } } }
 (14) else  SrhSet = 1st half of SrhSet
 (15) return ( ExBinSrh(target, SrhSet))
```

There are two issues yet to be addressed. First, what is the proper condition to have the search agent switch from a search strategy to another? For instance, the time since the target agent left the current probed server is shorter than a certain limit. This is a parameter requiring intensive experiments. The second question is how to modify the EBS to take advantage of this log information. One possible way to improve EBS or BBS is to adjust the range of search probes to increase of the possibility of probing a visited node. (For instance, the forward probes can be shorter than half of the search list and the backward probes can be longer than half of the search list.) In a dynamic search, probing a visited server is better than probing a unvisited server. This is because the search agent can make use of the log information to obtain a more accurate prediction if the target agent had visited the currently probed server. Both issues require intensive experiments to address. It is beyond the scope of this paper. They will be studied in the future.

5. Issues and Future Research

5.1 Exact Search vs. Approximate Search

Because the execution of an agent continues to progress, the exact current location of an agent might have already changed when the client receives the acknowledge. To make the location of an agent remain unchanged after it is found, a freeze agent has to be submitted together with the search agent as explained in [2]. Otherwise, only an approximate location will be obtained.

5.2 Non-deterministic Execution

The search algorithms presented so far all assume that the agent to be searched traveled along a predetermined path. This may not be the case if the path a target agent traveled along is non-deterministic. In this case, the chase-from-holder algorithms are more appropriate. It is yet to be researched if the status holder is not available.

5.3 Lost Agent

An agent might get lost due to a failure in the agent itself, the server it resides, or the network it travels so that no search agent can locate it. Sometimes it has no harm to ignore the lost agent. However, it may be essential to recover the lost agent in some critical situations such as business transactions. This problem becomes more complicated when a concurrent execution is allowed. Therefore, further researches on this issue are needed.

5.4 Concurrent Search

Sequential search may take a much longer time than what a client can tolerate. A client may submit more than one search agent into a network to reduce the search time. A search agent may also create child search agents by itself to cover all possible paths in a non-deterministic situation such as an if-then-else decision. (The actual route an agent took may depend on real time conditions or the information it obtained in the course of its execution.)

We are expecting many open issues under this situation. The typical issues are:

- How to converge forked agents?
- What to do if some child agents or even the parent search agent itself gets lost?
- How to terminate all child agents?

6. Summary

In a mobile computing environment that supports mobile agents, a client can send an agent to visit a sequence of servers in the network. To track the location of agents becomes a critical problem in managing a mobile agent service network. This paper describes the performances of a number of intelligent search strategies that make use of prior knowledge about the possible service time each server may take. Our study found that, under a variety of circumstances, the mean service time is a reliable index to predict the location of a mobile agent even if the service time is irregularly probabilistic. By using this discovery in the proposed intelligent search, a search agent can easily locate the target agent in two or three probes without an intensive computation.

References

1. T. Imielinski and B. R. Badrinath, "Mobile Wireless Computing: Challenges in Data Management," *Communication of ACM*, August 1994.
2. Yao-Nan Lien, "Client and Agent Mobility Management," *Proc. of the Second Workshop on Mobile Computing*, Hsing-Chu, Taiwan, March 1996, pp. 141-152.
3. Yao-Nan Lien, "An Open Intelligent Messaging Network Infrastructure for Ubiquitous Information Service," *Proc. of the First Workshop on Mobile Computing*, Hsing-Chu, Taiwan, April 1995, pp. 2-9.
4. Yao-Nan Lien and Chun-Wu Leng, "On the Search of Mobile Agents," *Proc. of the IEEE Personal, Indoor, and Mobile Radio Conference*, Taiwan, Oct. 1996, pp. 703-707.
5. Yao-Nan Lien, et. al., "FlyingCloud: A Mobile Agent Service Network", *Proceedings of the International Conference on Distributed Systems, Software Engineering, and Database Systems*, Dec. 1996, pp. 177-183.
6. P. Maes, "Agents that reduce work and information overload", *CACM*, July 1994, pp. 30-41.
7. M. Weiser, "The computer for the 21st century", *Scientific America*, 1992, pp. 94-104.
8. James E. White, "Telescript Technology: The Foundation for the Electronic Marketplace", General Magic, Inc.
9. TIA/EIA IS-41, "Cellular Radio Telecommunications Intersystem Operations", *Telecommunications Industry Association*, Dec. 1991.