
ELSEVIER

Information Sciences 122 (2000) 227–240

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

A novel mobile agent search algorithm

Wen-Shyen E. Chen [a,*], Chun-Wu R. Leng [b], Yao-Nan Lien [b]

[a] *Institute of Computer Science, National Chung-Hsing University, Taichung 40227, Taiwan, ROC*
[b] *Department of Computer Science, National Chengchi University, Taipei, Taiwan, ROC*

Received 12 September 1997; received in revised form 23 January 1999; accepted 15 March 1999

Abstract

Intelligent agent has been shown to be a good approach to addressing the issues of limited capacity and unreliable wireless links in mobile computing. However, before the approach can be commercially viable, a set of management capabilities that support the controls of intelligent agents in a mobile environment need to be in place. Since controls can only be applied after the target agent is located, an effective agent search algorithm is an indispensable part of the management functions. In this paper, we propose a new algorithm, the Highest Probability First Algorithm, for locating the target agent. The approach makes use of the execution time information to reduce cost and network traffic. The execution time of the agent on a server is assumed to be binomial distributed and therefore is more realistic.

1. Introduction

Compared to the conventional computers with a fixed connection to wired networks, mobile computers have narrow, unreliable connectivity, limited processing power and battery capacity, and have to operate in a dynamic, heterogeneous environment [1,2]. As a result, to realize nomadicity – the ability to allow the user to move from one place to another and retains access to a rich set of information and communications services while moving, a new paradigm for information processing and communications is needed. Intelligent Agent

* Corresponding author. Fax: +886-4 285 3869.
E-mail addresses: echen@cs.nchu.edu.tw (W.-S.E. Chen), leng@cs.nccu.edu.tw (C.-W.R. Leng).

0020-0255/00/$ - see front matter
PII: S0020-0255(99)00122-X

[3–7] is shown to be promising in addressing the issues of limited capacity and unreliable links of mobile computers. In this approach, an electronic message that carries a computer program, whether procedural or declarative, which can be executed by the computer system of the recipient on behalf of the client, is submitted by the client and can navigate autonomously through heterogeneous networks. The message is capable of interacting with servers or other agents at where the services are provided, moving to another machine while carrying the intermediate results, and resuming execution when it reaches the destination. After the message is submitted, the client can be disconnected from the network. The client will be notified when the agent finishes its task or is aborted.

With this approach, the mobile clients and servers are decoupled in the sense that instead of getting intermediate results many times, the client interacts with the network only when it is submitting the agent and when the agent returns with results, as depicted in Fig. 1. The asynchronous interaction between the client and the server provides robustness against frequent disconnection suffered by mobile computers. In addition, since the agent can incorporate smarter query languages, less data needs to be transmitted in the network. The feasibility of this approach is evidenced by several prototypes that employ this new paradigm of computing [7,8].

Nevertheless, a complete functioning agent, whether simulated in software or implemented from a robot, needs an integrated collection of diverse but

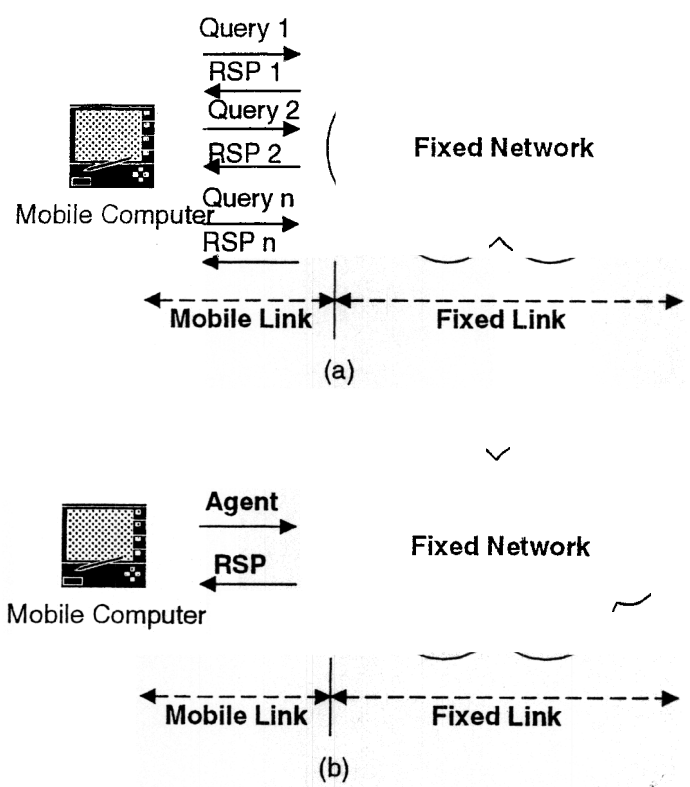

Fig. 1. The interactions between the mobile computer and the fixed network: (a) The conventional approach. (b) The intelligent agent approach.

interrelated capabilities, i.e., an architecture [7,8]. In addition, before an intelligent agent service can be accepted, a high quality and cost effective agent mobility operation, administration and maintenance (OA&M) system must be in place to guarantee a certain level of service quality. For any control to be applied to the target agent, it needs to be located first. Therefore, *agent location* is an indispensable part of the OA&M.

There are several approaches for agent location [9]: A straightforward approach, which is suitable for locating a target agent traversing the servers it will visit in a non-deterministic order, requires the dispatched agent to report its location and status to a *status holder* when it arrives at a server or when some specified events happen. The target agent can then be located by consulting the status holder. The other approach that does not require any status holder is to "blindly search" the target agent by broadcasting search agents to servers. These approaches will generate unwanted network traffic and further complicate management functions. However, there are applications that can determined the servers and the sequence of the servers the mobile agent will traverse before it starts. In [9], the author proposed to make use of the execution time information available to search the target agent more intelligently when the sequence of the servers to traversed can be determined beforehand. The result shows that the number of the probes can be effectively reduced. Nevertheless, the complexity of the algorithm is high and the assumption that the expected execution time on a server is *uniformly* distributed over a time period might not be realistic.

In this paper, we propose a new agent search algorithm, the highest-probability-first search (HPFS) algorithm, that makes use of the execution time information. In the HPFS algorithm, the execution time on a server is assumed to be *binomial distributed* [10], which is closer to reality. The derived probability function is shown to be much less complicated and can be adopted by a search agent when being sent to locate the target agent. Although the itinerary of the agent is assumed to be non-branching, which means that the agent will visit a set of servers in sequence, the proposed approach can be easily extended to cover the branching cases, as in the timed protocol specification and validation [11,12].

The rest of the paper is organized as follows. Section 2 illustrates an architecture that supports the mobility of intelligent agents. Section 3 describes the HPFS algorithm we propose. Simulation results are presented in Section 4. Concluding remarks and the future research topics will be given in Section 5.

2. An architecture to support agent mobility

In many cases, a mobile agent can be viewed as a delegate for a client. It travels in a service network, acting on behalf of the client to request services

from servers it visits. An architecture that can be used to support intelligent messaging in mobile computing is depicted in Fig. 2. Note that in the architecture, there are four types of agents: User Agent, Broker Agent, Service/Resource Agent, and Management Agent.

The user agent is submitted by the client to the fixed network. It carries the "goals" or requests for services from the client and will first arrive at the mobile supporting server. The mobile supporting server acts as a "proxy" for the mobile client and will be the home base for the agent being submitted. The user agent will first consult with the broker agent for the servers that can fulfill its goals or provide the requested services. It then travels to the resource/service providers and interacts with their service/resource agents at that location. The user agent might have to move in between several service/resource providers to have the requested work done. When the requested services are fulfilled or when the user agent is being called back, the user agent will return to the client. If the client is not currently available, the result will be temporarily stored in the access node or network management center (NMC) and the mobile client will be notified. The mobile client can then retrieve the stored result. When the user agent is in the service network, the client can also request management functions from the management agent to have the control over the target agent. Basic control functions include:

- *terminate* – terminate the execution of the target agent;
- *freeze* – postpone the forwarding of an agent until a resume message is received;
- *suspend* – suspend the execution of an agent until a resume message is received;

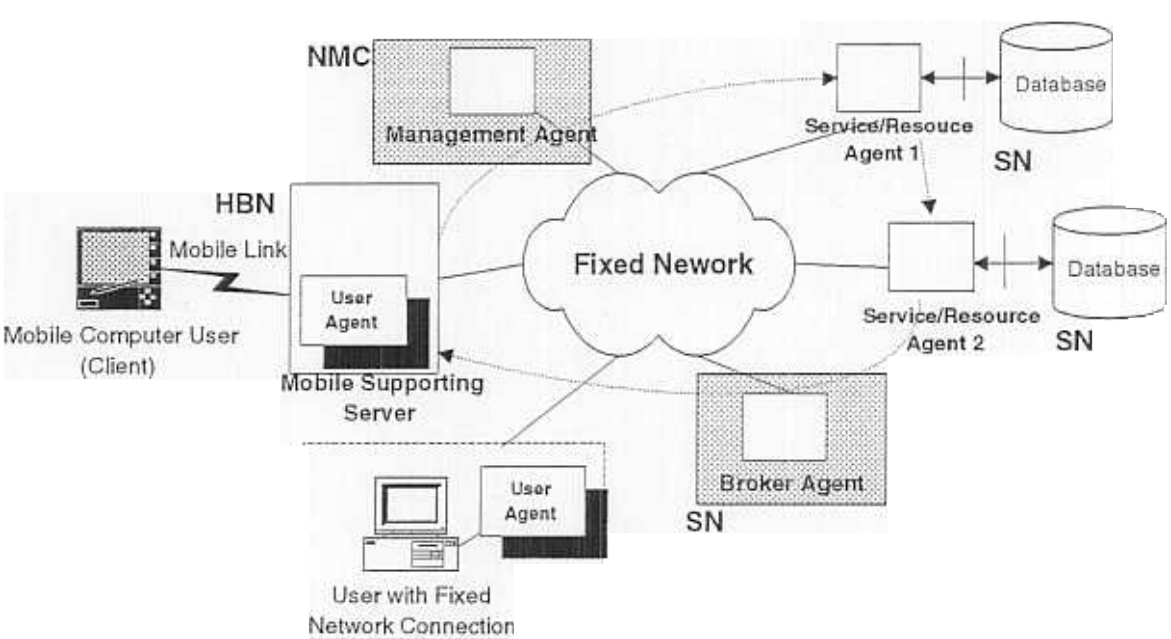

Fig. 2. An architecture to support intelligent agent in mobile environment.

- *resume* – resume the execution of a suspended or frozen agent.

The same architecture can be used by both mobile clients and clients with fixed network connections, as shown in Fig. 2.

3. The HPFS algorithm

3.1. Location estimation

According to the previous discussion, the performance of a search algorithm is determined by the time spent on locating the target agent, as well as the network overhead caused by the algorithm. Both evaluation criteria, in fact, are mainly resulted from the number of times that the search agent probes servers to locate the agent. Therefore, a strategy of querying to the server with the highest probability among those servers will consequently consume less search time and network overhead than blind search strategies. For instance, comparing with the binary search (BS) algorithm [9], the HPFS algorithm not only reduces the network traffic when searching the target agent, but also improves search efficiency.

The following notations are used in evaluating the HPFS.

1. $(S_1, S_2, \ldots, S_n)$: an ordered sequence of servers that the target agent will visit.
2. $[t'_{S_i}, t''_{S_i}]$: a service time range that the target agent stays in server S_i.
3. t_{S_i}: service time of the target agent completing its job at server S_i, i.e., $t''_{S_i} - t'_{S_i}$.
4. T_{S_i}: summation of the service time that the target agent stays in S_1, S_2 to S_i; i.e., $\sum_{\ell=1}^{i} t_{S_\ell}$.
5. T'_{S_i}: summation of all the minimum service time that the target agent stays in servers S_1, S_2 to S_i; i.e., $\sum_{\ell=1}^{i} t'_{S_\ell}$.
6. $\bar{\mathscr{F}}^t_{S_i}$: probability of the target agent still running at server S_i after t seconds since the agent is initially delivered to server S_i.
7. $P^T_{S_i}$: probability that the target agent is currently located at server S_i, T seconds after it was initiated at S_1.
8. $\mathscr{E}_T$: expecting number of probes that client may locate the agent, T seconds after it was initiated at S_1.
9. $\mathrm{Rank}_T(S_i)$: a function that returns the rank of server S_i by comparing with the value of $P^T_{S_i}$.

Instead of blindly searching for the target agent, the HPFS algorithm sends a probe to a server with the highest probability that the agent might currently stay. If the result of probing is negative, the server with the second highest probability will be the next target. This search strategy will continue until the agent is located. Obviously, HPFS requires less number of probes to reach the agent than other blind search algorithms, and thus reduces a certain amount of search time as well as network traffic overhead. However,

the essential part of the HPFS algorithm is the method to evaluate probabilities for the servers that the agent might currently stay. Conclusion of the following theorem provides an efficient way to determine the probability values.

Theorem 1. *Assume that the service time of an agent to complete its job on each server is binomial distributed over the time range* $[t'_{S_i}, t''_{S_i}]$. *After T seconds since the target agent is initialized in the first server* S_1, *the probability* $P^T_{S_i}$ *of the agent being located in server* S_i *is formulated as*:

$$P^T_{S_i} \quad \frac{1}{2^{T_{S_{i-1}}}} \sum_{j=0}^{t''_{S_i}} \binom{T}{T_{S'_{i-}}} \quad 1 \quad \frac{}{2^{t_{S_i}}} \sum_{\ell=0}^{t''_{S_i}} \binom{t_{S_i}}{t'_{S_i}}$$

To verify Theorem 1, without loss of generality, we make the assumption that the probability function, say $\mathscr{F}_{S_i}(x)$, of the service time of server S_i is a *normal-distribution-like* function over the execution time range from t'_{S_i} to t''_{S_i}. That is, the agent could spend an arbitrary length of time to finish its work in server S_i within the time range, but the highest probability of the length of time for the agent to complete its job should be around the mid-point between t'_{S_i} and t''_{S_i}. Obviously, the assumption of normal-distribution-like probability function seems to be more practical and reasonable than other distribution functions, such as the uniform distribution function. Consequently, the probability function $\mathscr{F}$ can be formulated to be a binomial distribution function, which is a *discrete function* with a shape similar to that of the curvature of a normal distribution function [10].

$$\mathscr{F}_{S_i}(x) \quad \frac{1}{2^{t_{S_i}}} \sum_{\ell=0}^{t_{S_i}} \binom{t_{S_i}}{\ell} x^{t'_{S_i}+\ell}$$

Note that $x^{t'_{S_i}}$ is the lower bound of the time interval that the agent will stay in server S_i. The coefficient of a term $x^{t'_{S_i}+\ell}$ in Eq. (2) represents the probability of agent to spend $t'_{S_i}+\ell$ seconds to accomplish the work in S_i. Those Δ term powers are out of the summation range will be considered to have zero probability. Consequently, Eq. (2) can be simplified as

$$\mathscr{F}_{S_i}(x) = \frac{}{2^{t_{S_i}}} x^{t'_{S_i}} \quad (1+x)^{t_{S_i}}$$

Equation (3) is mainly used to depict the length of time that agent requires to complete its work in a server. If the time to deliver the agent from the end of a server S_i to the start of the next server S_{i+1} is negligible (or the deliver time can

be treated as a part of the responsibility to server S_i), Eq. (3) is useful to analyze an agent search model which prioritizes the order of servers to be searched according to their corresponding $\mathscr{F}_{S_i}(x)$ values.

The $P_{S_i}^T$ for each server S_i not only can be used to determine which server has the highest probability that the target agent is currently located, but also to calculate the expecting number of probes to be sent to reach the target agent. Considering the probability that the target agent is in server S_i after T seconds from the client sending out the agent to the first server, the probability consists of several disjoint components. The first component is in conjuction with two probability values: the probability that the previous $i-1$ servers spend all the T seconds services time, and the probability that server S_i will not finish the job within zero second. According to the results of Eq. (3), probability function of the first $i-1$ servers is the production of all the $i-1$ server probability functions because all the first $i-1$ serves can be considered as one large system, and each individual server among them is just one step of the whole procedure. Therefore, the probability function of the first $i-1$ servers as a whole can be formulated as

$$\prod_{k=1}^{i-1} \mathscr{F}_{S_k}(x) = \frac{1}{2^{T_{S_{i-1}}}} x^{T'_{S_{i-1}}} \cdot (\ \ + x)^{T_{S_{i-1}}} \tag{4}$$

Then, the coefficient of the term x^T in Eq. (4) represents the probability that the target agent spends exactly T seconds in $i-1$ servers in total.

$$\text{coef. of } x^T = \frac{1}{2^{T_{S_{i-1}}}} \begin{pmatrix} T_{S_{i-1}} \\ T - T'_{S_{i-1}} \end{pmatrix} \tag{5}$$

Next, the probability that the target agent *will not* finish the work at server S_i in t seconds is abbreviated by a notation $\bar{\mathscr{F}}_{S_i}^t$. We can describe the function $\bar{\mathscr{F}}_{S_i}^t$ from a different point of view: the probability value will be one minus each probability value that the job will be done in less than t seconds.

$$= 1 - \frac{1}{2^{t_{S_i}}} \sum_{\ell=0}^{t} \begin{pmatrix} t_{S_i} \\ \ell - t'_{S_i} \end{pmatrix} \tag{6}$$

Concluding from the the discussion above, as well as Eqs. (5) and (6), Theorem 1 can be verified from the following formulation.

$$= \text{prob. of first } i-1 \text{ servers spend } \ T \text{ seconds}$$

$$* \text{ prob. of server } S_i \text{ spends over 0 seconds}$$

$$+ \text{prob. of first } i-1 \text{ servers spend } T - \ \text{ seconds}$$

$$
\begin{aligned}
&* \text{ prob. of server } S_i \text{ spends over 1 seconds} \\
&+ \text{ prob. of first } i-1 \text{ servers spend } T-2 \text{ seconds} \\
&* \text{ prob. of server } S_i \text{ spends over 2 seconds} \\
&+ \\
&+ \text{ prob. of first } i-1 \text{ servers spend } T-t''_{S_i} \text{ seconds} \\
&* \text{ prob. of server } S_i \text{ spends over } t''_S \text{ seconds}
\end{aligned}
$$

$$
\frac{1}{2^{T_{S_{i-1}}}}\begin{pmatrix} T_{S_{i-1}} \\ T-T'_{S_{i-1}} \end{pmatrix} \quad 1-\frac{1}{2^{t_{S_i}}}\sum_{\ell=0}^{0}\begin{pmatrix} t_{S_i} \\ \ell - t'_{S_i} \end{pmatrix}
$$

$$
+\frac{1}{2^{T_{S_{i-1}}}}\begin{pmatrix} T_{S_{i-1}} \\ T-1-T'_{S_{i-1}} \end{pmatrix} \quad 1-\frac{1}{2^{t_{S_i}}}\sum_{\ell=0}^{1}\begin{pmatrix} t_{S_i} \\ \ell - t'_{S_i} \end{pmatrix}
$$

$$
+
$$

$$
+\frac{1}{2^{T_{S_{i-1}}}}\begin{pmatrix} T_{S_{i-1}} \\ T-t''_{S_i}-T'_{S_{i-1}} \end{pmatrix} \quad \frac{1}{2^{t_{S_i}}}\sum_{\ell=0}^{t''_{S_i}}\begin{pmatrix} t_{S_i} \\ \ell - t'_{S_i} \end{pmatrix}
$$

Note that Eq. (7) is the same as Eq. (1).

The probability function in Theorem 1 describes the percentage of each server S_i that the agent is located after T seconds from the agent is initialized. This probability function is helpful to evaluate the performance of various agent search algorithms. According the the discussion above, a better search algorithm consumes less number of probes to reach the agent and thus requires network traffic overhead. Therefore, the expected number of probes $\mathscr{E}_T$ to find the agent is the key criterion to evaluate a search algorithm. For instance, if a sequential search algorithm to probe the $S_1, S_2, \ldots, S_n$ in order, its expected number of probes will be formulated as

$$
\mathscr{E}_T = \sum_{i=1}^{n} i \left\{ \frac{1}{2^{T_{S_{i-1}}}}\sum_{j=0}^{t''_{S_i}}\begin{pmatrix} T_{S_{i-1}} \\ T-T_{S'_{i-}}-j \end{pmatrix} \quad 1-\frac{1}{2^{t_{S_i}}}\sum_{\ell=0}^{t''_{S_i}}\begin{pmatrix} t_{S_i} \\ \ell - t'_{S_i} \end{pmatrix} \right\}
$$

Since the HPFS algorithm always first searches a server with the highest $P_{S_i}^T$ value among those servers to be searched, its expected number can be expressed as

$$\mathscr{E}_T = \sum_{i=1}^{n} \mathrm{Rank}_T(S_i) \left\{ \frac{1}{2^{T_{S_{i-1}}}} \sum_{j=0}^{t''_{S_i}} \binom{T_{S_{i-1}}}{T - T_{S'_{i-1}} - j} \right.$$

$$\left. \frac{1}{2^{t_{S_i}}} \sum_{\ell=0}^{t''_{S_i}} \binom{t_{S_i}}{\ell - t'_{S_i}} \right\}$$

Note that the function $\mathrm{Rank}_T(S_i)$ returns the rank of server S_i's probability value in comparison with all the other servers. Obviously, a higher $P_{S_i}^T$ value will be multiplied to a smaller rank, and hence produces a smaller $\mathscr{E}_T$ than other search algorithms.

3.2. The HPFS algorithm

With the results from the previous section, we propose to include the following HPFS algorithm in the search agent to locate the target agent.

HPFS algorithm

```
Main {
    ServerSet = {S1, S2, ..., Sn}
    PrioServerList = Sort(ServerSet)
    HPFS (target, ServerSet)
}
Procedure Sort(ServerSet){
    Sort ServerSet in decreasing order according to
    corresponding probability values
}
Procedure HPFS (target, ServerSet){
    if (ServerSet = ∅)
    then return NOT_FOUND
    St = First server in PrioServerList
    Move the Search Agent to St
    if (target found in St)
    then return (St)
    else if (St has been visited by the target agent)
    then PrioServerList = Sort (ServerSet - {S1, S2, ..., St})
    else PrioServerList = Sort (ServerSet - {St, St+1, ..., Sn})
    return (HPFS (target, PrioServerList))
}
```

Note that "ServerSet" and "PrioServerList" are global variables.

In this algorithm, if the search agent arrives at a server S_t and finds that the target agent has moved away from that server, then we should exclude all

the servers that proceed S_t from the search list. This is based on the assumption that the servers will be visited in sequence. On the other hand, if the target agent has not arrived at S_t, then all the servers following S_t in the original execution order will be excluded in the future search for the same reason. The search list will then be sorted according to the servers' corresponding probability values. Since the target agent is still mobile before being located, it is possible that the target agent might "slip through" the search, i.e., the target agent might move to servers excluded from the search list in previous rounds of search. A simple solution is to leave some information at the servers that the search agents have visited and ask the target agent to report its position and status when it arrives at those servers. As can been seen, the HPFS can effectively reduce the unnecessary probes in searching the target agent.

4. Simulation results

In this section, we present the simulation results and use "number of probes" needed to locate the target agent as a performance measure to compare the basic binary search and the HPFS.

In the simulations, we assume that the target agent will visit 20 servers, numbered in sequence form 1 to 20. The service time ranges for the servers are as shown in Table 1. The elapse times range from 1 to 200 in the simulations and the service time range for server 20 is chosen so that it can be a "sink", i.e., the target agent will not go beyond server 20. The values of the simulation results are obtained by taking the average of the results of 1000 runs.

Our goal is to predict with certain accuracy where the target agent is when the elapse time and execution time ranges are given. Therefore, it is of interest to know if the Eq. (1) can show the probability of where the agent is accurate and if the difference between the theoretical and simulations results vary with elapse time. Figs. 3–5 (with the elapse time equals to 15, 85, and

Table 1
Service time ranges for the servers

Server		2	3	4	5	6	7	8	9	10
t'_{S_i}	3	4	6	2	3	2	8	7	4	
t''_{S_i}	8	20	17	10	14	6	16	22	20	9
Server	11	12	13	14	15	16	17	18	19	20
t'_{S_i}	8	6	3	5	9	2	4	2	7	500
t''_{S_i}	17	20	18	14	13	11	17	21	28	600

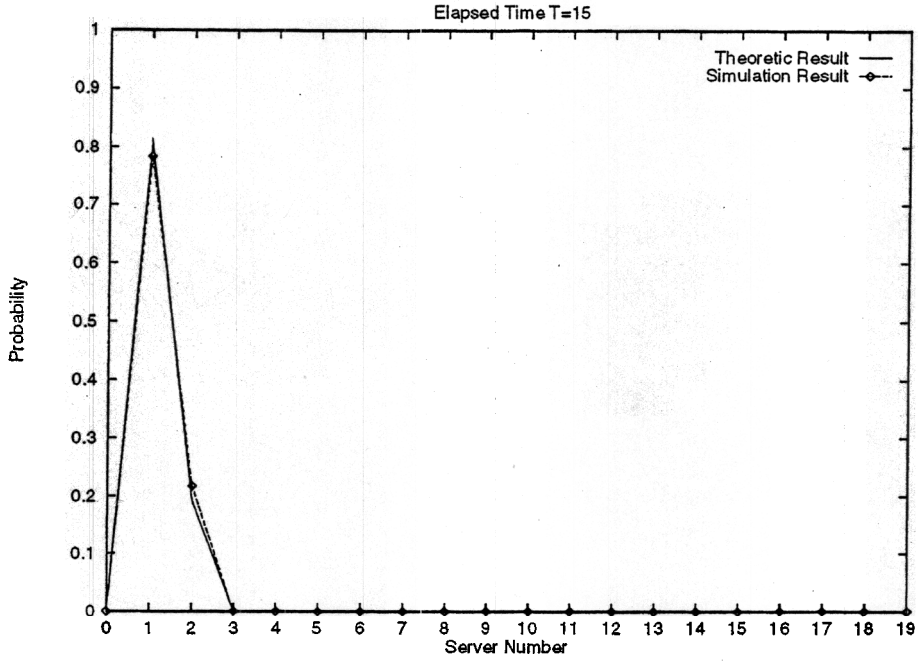

Fig. 3. Probability distribution of the location of the agent (Elapse time = 15).

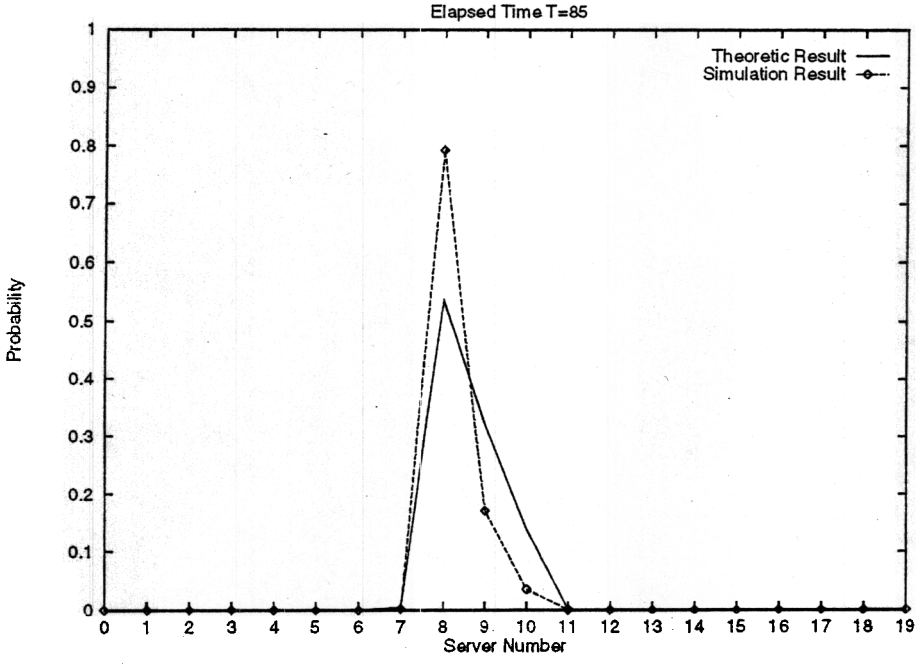

Fig. 4. Probability distribution of the location of the agent (Elapse time = 85).

145, respectively) show that the simulation results are very close to the theoretical results with different elapse times, with the highest probability all pointing to the same server.

Fig. 6 shows the comparison of the basic binary search and the HPFS algorithms. As illustrated in the figure, the basic binary search algorithm needs more probes to locate the target agent. In addition, the numbers of probes

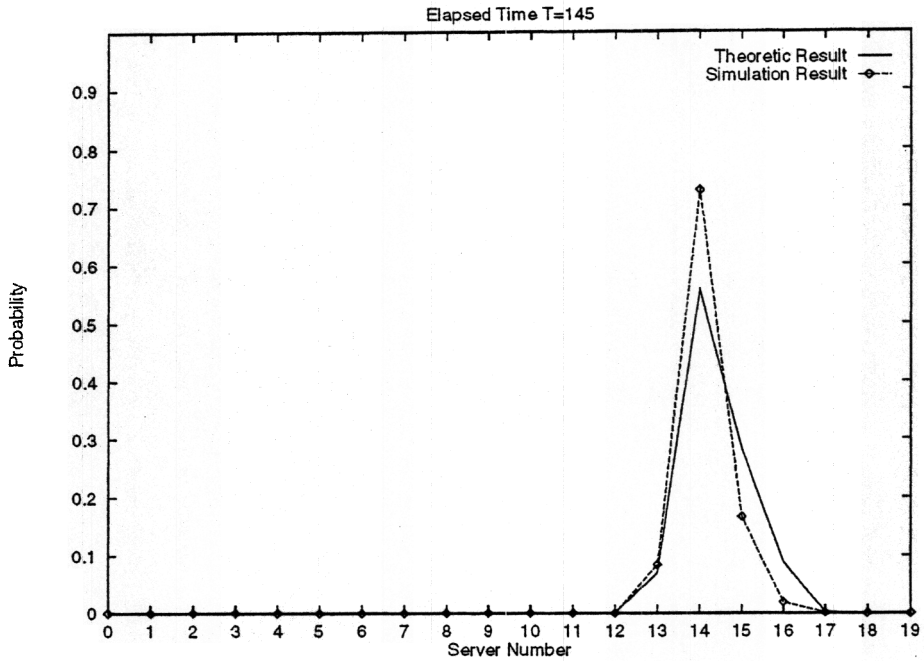

Fig. 5. Probability distribution of the location of the agent (Elapse time = 145).

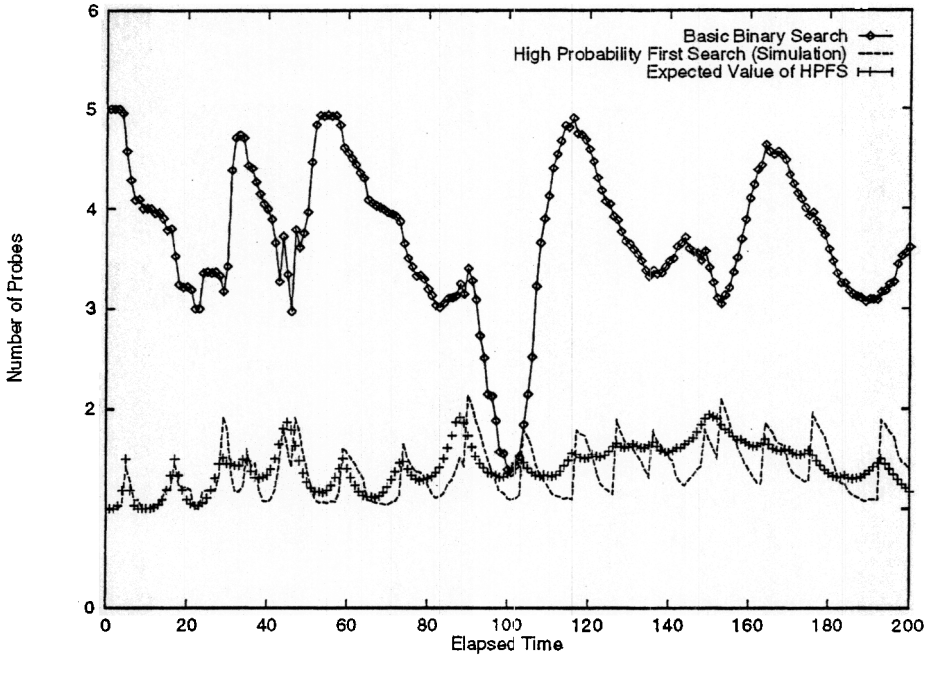

Fig. 6. Comparison of the basic binary search and the HPFS.

needed vary in a wide range with the elapse time. On the contrary, the expected values of the probes needed for the HPFS algorithm are lower than those for basic binary search and the variation is much smaller. The simulation results match the theoretical results quite well. Consequently, the validity of Theorem 1 can be verified.

5. Conclusion

Agent location is an indispensable part of the management functions needed to support the mobility of intelligent agents. Straightforward agent search algorithms often lead to excessive network traffic. In this paper, we have proposed a new agent search algorithm that makes use the execution time information available. The execution time is assumed to be binomial distributed, which is closer to reality. With the probability functions we came up with, the HPFS algorithm is then formulated. It is expected to generate less probes to locate the target agent. The simulation results match the theoretical results quite well.

We had made the assumption that the target agent traverses the servers in a predetermined order. However, the path the target agent takes might depends on the real time condition and could be non-deterministic. In addition, the relationship between agent location and agent control functions (how to apply the control function after the target agent is located?) needs to be clarified. We plan to resolve these problems in the future research.

Acknowledgements

We gratefully acknowledge the help of C.-Y. Lin, Y.-T. Liu, and C.-J. Chu for developing the simulation programs.

References

[1] G.H. Forman, J. Zahorjan, The challenges of mobile computing, IEEE Computer (1994) 38–47.
[2] T. Imielinski, B.R. Badrinat, Mobile wireless computing: Challenges in data management, Communications of the ACM, August 1994.
[3] M. Wooldridge, N.R. Jennings (Eds.), Intelligent Agents: Theories, Architectures, and Languages, Lecture notes in AI – vol. 890, Springer, 1995.
[4] M. Wooldridge, N.R. Jennings (Eds.), Intelligent Agents II: Theories, Architectures, and Languages, Lecture notes in AI – vol. 1037, Springer, 1996.
[5] N. Jennings, M. Wooldridge, Software agent, IEEE Personal Commun. Magazine (1996) 17–20.
[6] J.E. White, Telescript Technology: The Foundation of the Electronic Marketplace, White paper, General Magic, 1994.
[7] W.-S. E. Chen, Y.-N. Lien, Intelligent messaging for mobile computing over the world-wide web, in: Proceedings of the Second Workshop on Mobile Computing, April 1995.
[8] Y.-N. Lien, An open intelligent messaging network infrastructure for ubiquitous information service, in: Proceedings of the First Workshop on Mobile Computing, April 1995.
[9] Y.-N. Lien, C.-W. R. Leng, On the search of mobile agents, in: Proceedings of the Seventh IEEE Symposium of Personal, Indoor, and Radio Communications, October 1996.

[10] R. Jain, The Art of Computer System Performance Analysis, Wiley, New York, 1991.
[11] F.-J. Lin, P.M. Chu, M.T. Liu, Protocol verification using reachability analysis, ACM Comput. Commun. Rev. 17 (5) (1987) 126–135.
[12] G.J. Holzmann, Design and Validation of Computer Protocols, Prentice-Hall, Englewood Cliffs, NJ, 1991.