行動代理人不確定路徑的二元搜尋法
Non-Deterministic Binary Search of Mobile Agents

張宏慶
Jang, Hung-Chin
連耀南
Lien, Yao-Nan
劉富翰
Liu, Fu-Han

政治大學資訊科學系
Dept. of Computer Science
National Cheng-Chi University
Taipei, Taiwan, R.O.C.

黃智賢
Huang, Jyh-Shyan

政治大學應用數學系
Dept. of Applied Mathematics
National Cheng-Chi University
Taipei, Taiwan, R.O.C.

摘要

在一支援行動代理人(mobile agent)的行動計算網路環境中，用戶可送出一行動代理人到網路依序拜訪網路上的伺服器。為了管理行動代理人在服務網路上的行動，其位置的追蹤方法成為一重要的研究主題。Lien[6]提出數個追蹤的方法且研究這些方法的執行效率，然而這些方法都需假設在被追蹤的代理人之行動路徑為事先預知的情況下，才能追蹤到代理人的位置。在本篇論文中，我們提出不確定路徑的二元搜尋法(Non-Deterministic Binary Search，NDBS)，NDBS 不受以上假設的限制。NDBS 的搜尋時間複雜度為 $O(m+\log(\frac{n}{m}))$，其中 m 、n 分別為行動代理人在其生命週期中所拜訪之不確定節點的總數及所有被拜訪之節點的總數。

關鍵字：行動代理人，智慧型搜尋法

Abstract

In a mobile computing environment that supports mobile agents, a client is able to send an agent to visit a sequence of servers in the network. Tracking the locations of agents becomes a critical problem in managing a mobile agent service network. In [6], Lien proposed several blind and intelligent search methods and studied their performances. Since all these search strategies are under an assumption that the agent to be searched moves along a pre-deterministic path. In this paper, we propose a search strategy called Non-Deterministic Binary Search (NDBS) which will release this assumption and locate an agent through a non-deterministic path. The time complexity of our NDBS is $O(m+\log(\frac{n}{m}))$*, where m and n are the numbers of visited non-deterministic nodes and all the nodes to be visited by the target agent during its lifetime, respectively.*

Keywords : mobile agent, intelligent search

1. Introduction

1.1 Agent and Agent Mobility

The goal of an ubiquitous information service network is to provide information to the users anytime and anywhere [8]. Thus, a service network must be provided with a wireless communication network and be able to easily access to various information resources [10].

Due to the immaturity of distributed computing technology, clients have to access network resources in a prescriptive fashion by interacting with individual servers. However, in most mobile computing environments, the nature of communications is intermittent and the battery energy is limited. Thus, it becomes more difficult to accomplish a complicated task that requires intensive interactions among its client and multiple servers. A non-traditional computing paradigm, intelligent messaging, allows clients to interact with multiple servers in a dynamic fashion has been brought up to cope with this problem [1,2,3,8,10].

This work is partly supported by the National Science Council of the Republic of China under the grant NSC-87-2213-E-004-009.

Simply speaking, an intelligent message is an electronic message carrying a computer program, either procedural or declarative, that 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 automatically forward the message to another server, upon which the program is executed continuously in a pipeline fashion. We would use the term, mobile agent, as a substitution for intelligent message through the paper. Good examples are referred to [3].

Since an agent moves along a service network, the originating client may not be able to track or control its operation directly. A service network must provide some mechanisms allowing its clients to track and control these messages. This problem is referred to as 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 OA&M (Operation, Administration, and Maintenance). It will be impractical to demand the comparable resources to support OA&M functionalities in many prospective information services. Thus, the computing community have to develop and deploy demanded functionalities by themselves. All infrastructures and solutions must not require any change to the existing telecommunication network. To achieve this, we employ an open service network architecture [3] to separate service networks from transport networks. Under this architecture, it allows services of varied scales and qualities to be introduced into the network easily. Readers are referred to [3] for details.

On top of this open architecture, Lien proposed a hybrid operation model [2] that allows a service provider to offer both centralized and distributed operation models to its subscribers. Subscribers can choose to use their own Internet facility, e.g., Home Base Node (HBN), to share the OA&M functionalities. At their own expense, subscribers have alternatives to 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 based on the proposed open architecture and managed using that operation infrastructure.

1.2.2 Search a Mobile Agent

After an agent is submitted into a service network, the client or the network manager may want to know its current location for either inquiring its status or controlling its execution, etc. An obvious solution is to send another agent, called search agent, to track the original agent along the original path, or to broadcast a message to all the servers that the agent may choose to go to. A couple of problems might arouse with this solution:

(1) The cost of sending many messages over a wireless network is high.
(2) The path that an agent moves along might be non-deterministic and this is hard to trace.
(3) A sequential search consumes lots of time.

Lien [4] proposed several search strategies to cope with these problems. However, all these strategies assume that the path of the target agent is deterministic. In this paper, we introduce a Non-Deterministic Binary Search (NDBS) method to release this assumption and make it non-deterministic. The rest of the paper is organized as follows:
In section 2, we review those search strategies given in [4,6]. The proposed Non-Deterministic Binary Search method is included in section 3. At least, we have conclusion and future research in section 4 and 5, respectively.

2. Previous Work

The searching strategies proposed in [4,6] can be classified into two groups: blind searches and intelligent searches. The intelligent search strategy makes use of prior knowledge about the execution of all tasks, while blind search strategy doesn't.

The following denotations will be used in the rest of the paper :

$S = \{ S_1, S_2, ..., S_n \}$: the set of distinct servers visited by an agent.

T : the elapsed time since the target agent was originated.

T_i : the service time at server S_i, it is the time duration that the target agent stayed at server S_i.

We assume that the target agent visits $\{ S_1, S_2, ..., S_n \}$ sequentially and non-recursively ; and the time for the target agent moving from one server to another is considered nominal and is thus ignored.

2.1 Chase-From-Holder Algorithm

In the hybrid operational infrastructure proposed by Lien [2], users are encouraged to use their own Home Base Node (HBN) to participate in the management of their agents. One possible usage of HBNs is to store the current status of agents including agents' locations.

The path of a target agent may be either deterministic or non-deterministic. In the case of deterministic, the agent can be found by using BBS, EBS, IBS and ABS. If every step of the agent is deterministic, the sequence of servers to be visited by the agent can be determined beforehand. In case of non-deterministic, some steps of an agent are non-deterministic. It means that when an agent goes to one server, say server A, there are multiple alternatives to be its next stop. The next stop won't be clear only if the agent completes the task in server A.

For example, if someone plans to go form Peking to Taipei. He needs his agent to confirm the flight time and book a room for him. So, the agent goes to the airport to confirm the flight time and communicates with the information service agent in the airport to get information about the hotels in Taipei. After interacting with all hotels, it chooses one of the hotels and books a room. Here, the path of the agent is non-deterministic since all hotels in Taipei are possible candidates. This path is thus no more a sequential list of servers but a spanning tree of all alternative servers. An example is shown in Fig 1.

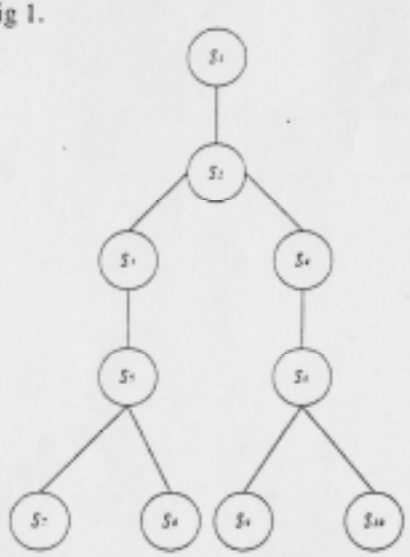

Fig. 1 A non-deterministic path represented by a spanning tree.

Definition 1 : A node S_i is called a deterministic node if and only if the next stop of target agent is deterministic when it stays at S_i.

Definition 2 : A node S_i is called a non-deterministic node if and only if the next stop of target agent is non-deterministic when it stays at S_i.

Definition 3 : A node S_i is called a first non-deterministic node (FN node) if and only if it is the first non-deterministic node to be visited by target agent.

Simply speaking, a node in the spanning tree that has more than one child is a non-deterministic node, eg. node S_2 in Fig. 1. Otherwise, it is a deterministic node, eg. node S_1. Both non-deterministic and deterministic nodes have exactly one parent node ; and deterministic is a special case of non-deterministic. By Definition 1, in the case of deterministic, every step of the target agent is deterministic and every node in the spanning tree is a deterministic node. The tree looks like a linear list of nodes. Therefore, the spanning tree spanned as a linear list of nodes if and only if the components of the tree are all deterministic nodes.

Theorem 1 : The spanning tree spanned as a linear list of nodes if and only if the components of the tree are all deterministic nodes.

By Definition 3, we conclude that every node between the FN node and the root is deterministic node.

Theorem 2 : Every node between the FN node and the root is a deterministic node.

3.2 NDBS Algorithm

Here we propose a Non-Deterministic Binary Search (NDBS) to search the target agent when the path is non-deterministic. It takes two steps to complete this search.

Step1: Search agent traces the first non-deterministic node (FN node) from the root of the spanning tree and comes to the following two alternatives:

Case 1 : If the target agent has visited the FN node, we will be able to know which child node is the next to be visited from the log information recorded in server. Then we prune off all the ancestor nodes and all those child nodes that won't be visited. Go to step1.

Case2 : If the target agent hasn't visited the FN node then we can make sure that the agent must stay in one of the nodes that between the root and the FN node. We then prune off all child nodes of the FN node. Since all the nodes between the root and the FN node are deterministic, the pruned tree becomes a list of nodes. An example of the reduced spanning tree is referred to Fig. 2. Go to step2.

Step2 : We use intelligent binary search, IBS, to find the target agent. In order to prevent target agent from passing the end node of the list before the search agent catches it, we leave the search agent at the end node waiting for target agent and fork

Thus, the reported current location of an agent can be obtained in the status holder (i.e. HBN) of its originating user. If the exactly current location at this moment is needed, the search agent will visit the reported location first and proceed with a sequential search from there if the target agent had passed that server. This agent search algorithm is called chase-from-holder algorithm.

One major problem with chase-from-holder algorithm is that it requires extra cost to update status periodically. This cost consists of that induced by network traffic and resource consumed 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 reporting 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 Basic Binary Search (BBS) and Extend Binary Search (EBS)

The Basic Binary Search (BBS) algorithm 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. On 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 it continues to move during the search. During the course of search, some unvisited servers may be excluded 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 to be visited are not included in the search list. As a result, the search agent fails to find the target agent.

The Extended Binary Search algorithm resolves 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 remains in the order of $log(n)$ with a larger coefficiency.

Without prior knowledge about the status of the target agent and servers, the binary search might be the optimal solution to search an agent. When the prior knowledge is considered, other algorithms would have better performance over binary search algorithm.

2.3 Intelligent Binary Search (IBS)

If a client has a better estimation on the service time in each probe, he may have a more precise prediction on the current location of an agent. With this information, we will be able to reduce the search time significantly. The intelligent search algorithms that make use of service time statistics distinct themselves from blind search algorithms.

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

$$\sum_{i=1}^{r-1} T_i \le T \le \sum_{i=1}^{r} T_i$$

In real world, it is hard to predict 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 samples 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, less high probability, 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 [4] derived a formula to calculate the (residing) probability for each server that a target agent might reside. In their Intelligent Binary Search algorithm (IBS), the search list is presorted according to the calculated probabilities. The search agent probes the servers in the list sequentially. During a search, the list is maintained in the following way:

(1) If a target agent has visited and left the currently probed server, all servers ahead of it are removed from the search list. The original search order remains unchanged and the search agent continues the search following the shrunken list.

(2) If a target agent has not visited the currently probed server yet, the search list won't be changed. The search agent will probe the server that precedes the current server with the highest residing probability.

3. Non-Deterministic Binary Search (NDBS)

The search agent algorithms presented so far all assuming that the agent to be searched is along a pre-deterministic path. We would like to further release this assumption by using a Non-Deterministic Binary Search (NDBS) which is applicable to the case when the path of the target agent is non-deterministic.

3.1 Deterministic Path vs. Non-Deterministic Path

a child agent using IBS method to continue the search. This induces two alternatives:

Case 1 : The child agent found the target agent and reported the location of the target agent to the search agent and terminates the search. The search agent then reports the result and terminates the whole searching process.

Case 2 : The search agent found the target agent. It means that the target agent arrives at the end node before the child agent finds it. The search agent then reports the location of target agent. Neither the search agent nor child agent will terminate until the search agent receives the search result of the child agent.

Fig. 2 A deterministic path represented by a list of nodes.

3.3 Verification of NDBS Algorithm

The original spanning tree is reduced in step1 and it won't stop until the tree becomes a linear list of nodes. The head of the linear list is the last non-deterministic node that the target agent has visited and the end is the next non-deterministic node to be visited. In this section, we use the two theorems in section 3.1 to verify our NDBS algorithm.

Let S_{last} be the last non-deterministic node the target agent has visited and S_{next} the next non-deterministic node to be visited. There is no other non-deterministic nodes in between. Let there be n nodes $S_1, S_2, \ldots, S_n$ to be visited sequentially ahead of S_{last}.

In step1, we send out a search agent to locate the target agent. In the beginning, S_1 is the first non-deterministic node, i.e. FN node, to be visited by target agent. The search agent comes to visit S_1 first and finds that the target agent has visited S_1. S_2 is the next non-deterministic node to be visited. The search agent then prunes off both the path between S_1 and its ancestor and all the paths between other child nodes that won't be visited.

After the pruning, S_1 and S_2 becomes the root and FN node of the tree, respectively. Repeating this step until S_n becomes the root of the tree. Since S_{last} is the next non-deterministic node to be visited after S_n, at the time when S_n becomes the root of the tree, S_{last} becomes the FN node of the tree. Continuing with this process until search agent visits S_{last} and finds that S_{last} has been visited. Then it prunes off the path between S_{last} and other nodes except the one between S_{last} and S_{next}. At this time, S_{last} becomes the root of the tree. Since there is no other non-deterministic nodes between S_{last} and S_{next}. According to Theorem 2, S_{next} is the FN node of the tree. The search agent goes to visit S_{next} and finds that S_{next} has not been visited yet. It prunes off the path between S_{next} and its children.

At this time, S_{last} becomes the root of the tree and S_{next} is the bottom node of the tree. Both S_{last} and S_{next} are deterministic nodes and there is no other non-deterministic nodes in between. Now, all nodes are deterministic nodes and the tree is reduced to a linear list of nodes. The head of the list is S_{last} and the end is S_{next}.

After step1, the original spanning tree is reduced to a linear list of nodes and any of BBS, EBS or IBS is applicable to locate the target agent in step2.

3.4 Analysis of Time Complexity

Let m and n be the numbers of non-deterministic nodes and nodes to be visited by the target agent during its lifetime, respectively. The worst case of step1 is that the target agent has visited the *mth* non-deterministic node while the search agent is still searching for it. The search agent won't take step2 after it traced these m non-deterministic nodes. So, the time complexity of step1 in the worst case is $O(m)$.

After step1, the original spanning tree becomes a linear list of nodes. Besides, the head and the end of the linear list of nodes are non-deterministic nodes. Since, there are n nodes to be visited and m nodes of them are non-deterministic. The average length of nodes between two non-deterministic nodes is $\frac{n}{m}$. In other words, the average length of the linear list of nodes is $\frac{n}{m}$. Therefore, if we use BBS to locate the target agent in step2. The time complexity of step2 is $O(\log(\frac{n}{m}))$.

Summing up the time complexity of step1 and step2, the time complexity of our NDBS is $O(m+\log(\frac{n}{m}))$. Similarly, if we adopt any one of EBS and IBS to locate the target agent in step2, we will have the same time complexity.

4. Conclusion

In a mobile computing environment that supports mobile agent, a client is able to send an agent to visit a sequence of servers in a network. To track the locations of agents becomes a critical issue in managing a mobile agent service network.

Lien proposed a number of intelligent search strategies in [6]. However, all those strategies base on

an assumption that the agent to be searched moving along a pre-deterministic path. In this paper, we also propose a search strategy called NDBS to release this assumption and the time complexity of NDBS is $O(m+log(\frac{n}{m}))$.

5. Future Research

We enumerate a number of other issues to be covered in this research.

5.1 Exact Search vs. Approximate Search

Due to the execution of an target agent is kept going, the exact current location of an agent might be changed when the client receives the acknowledge sent back by search agent. To make sure the location unchanged after it is found, a freeze agent is required to be sent together with search agent as explained in [2]. Otherwise, what we get is only an approximate location rather than a exact location.

5.2 Recursive Execution

The search algorithms presented so far all assume that the target agent to be searched moving along a non-recursive path, i.e., none of the servers will be revisited by the target agent. Our search algorithms exclude those servers once they are visited, it will cause problem when a server is to be visited a second time. We have to release this assumption in next version of our algorithms.

5.3 Lost Agent

An agent might be lost because of failures. The reasons could be the agent itself, the server it resides, or the network it moves along. Agent lost might be significant or not. In a real time control system or a business transaction system, agent lost may cause catastrophe. It becomes more complicated when concurrent executions are allowed.

5.4 Concurrent Search

Sequential search is simple but consumes time. If a client is allowed to submit more than one search agent to a network, search time may be saved to some degree. We also may consider to have a search agent create child search agents to cover all possible paths in a non-deterministic situation such as in making an if-then-else decision. The derived issues are as follows:

(1) How to converge forked agents?
(2) What to do if either child agents or parent search agent get lost?
(3) How to terminate all child agents?

References

[1] 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] Yao-Nan Lien, Fuhan Liu, Chun-Wu Leng and Wen-Shyan Chen, "Intelligent Search of Mobile Agents", 1997 International Conference on Computer System Technology for Industrial Applications, April, 1997, pp. 110-116.

[7] Yao-Nan Lien, Fuhan Liu, Wen-Shyen Chen and Chun-Wu Leng, "Asymmetric Binary Search of Mobile Agents ", Submitted to the 1997 International Symposium on Multimedia Information Processing.

[8] Maes, "Agents that reduce work and information overload", CACM, July 1994, pp. 30-41.

[9] Weiser, "The computer for the 21st century", Scientific America, 1992, pp. 94-104.

[10] TIA/EIA IS-41, "Cellular Radio Telecommunications Intersystem Operations", Telecommunications Industry Association, Dec. 1991.