Search of Mobile Agents

Yao-Nan Lien

Department of Computer Science

National Chengchi University

lien@cherry.cs.nccu.edu.tw

Abstract

To make information ubiquitously available to the people in the world requires not only the Information Superhighway; it also requires a new computing paradigm to overcome the intermittent connection problem inherent in mobile environments. The users of a mobile agent service network can submit a message containing an executable script to the network performing tasks on behalf of the user [1,2]. Our researches focus on the issues regarding to the operation, administration, and maintenance (OA&M) which are critical components of a service network. We have been developing a prototype of such a service network, *FlyingCloud*, in NCCU. The FlyingCloud will facilitate further study on various OA&M issues that are special to this new computing paradigm [4].

The agent in FlyingCloud is autonomous and self-contained. After being submitted to the network, it can be executed by a sequence of servers one-by-one. A server can execute the agent and forward it to the next one according to the script carried by the agent. Different from the RPC (Remote Procedure Call) approach, the execution of an agent in each server is atomic so that the control of an agent in a server is completely released after the agent moves to another server [4].

In order to control mobile agents and manage the service network, it is essential to be able to track the location of agents. One class of strategies is to have the target agent report its locations periodically so that it can be located immediately. To avoid the unnecessary messages for location reporting, another class of strategies searches the target agent "on-demand", at the cost of paying extra searching time. In our research, we have developed several search strategies corresponding to various situations. The target agent may or may not travel along a deterministic path. There are some applications in which the traveling sequence of an agent is predetermined. A typical class of applications is that whose component tasks has precedence dependency. For example, an agent that makes traveling plans for a client user may have to make flight reservation first, then rental-car reservation, finally hotel reservation. The search strategies reported in this paper assume that the target agent travels along a deterministic path.

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 the list each 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 locate the target agent is in the order of $\log(n)$, where n is the number of servers in the search list [3].

BBS might be a very good search strategy intuitively. However, BBS may fail to locate the target agent if the target agent 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. Unfortunately, the target agent may *slip through* the search window so that the servers it visits afterward are not included in the search list. As a result, the search agent will fail to locate the target agent. Although BBS is naive and faulty, it provides a basis to mutate into other search strategies as well as a baseline for performance comparison.

The *Extended Binary Search* (EBS) algorithm corrects the slip-through problem by not excluding any unvisited server out of the search list at the cost of demanding more search probes [3]. In [6], we found a very surprising fact that in Extended Binary Search algorithm, the best pioint (server) to probe is not right in the middle of the serach list. This is because forward and backward probes are treated differently. Probing a visited server can exclude some servers out of the search list, while probing an unvisited server can't. In our simulation study, we found that the best probing range is between 0.25 and 0.4 of the search list starting from the head.

Above search strategies are classified as *blind search* because they do not use prior knowledge about the status of the target agent and servers.

Intelligent Search Algorithms

If a client has a good estimation on the service time in each server, he/she may be able to guess the approximate location of an agent. By using this information in a search, we 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.

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. Furthermore, for security and privacy reasons, it may be more practical to obtain statistics rather than detailed execution time records from servers. As a result, the location of the agent is probabilistic. We can calculate the location of the target agent with the highest probability, next highest, etc., and then search the target agent according to the order of probability. Assuming that the service time of each task is uniformly distributed, we derived a formula to calculate for each server the probability that a target agent might reside [3]. For simplicity, this probability is called the *residing probability*.

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.

Another problem associated with the intelligent search strategy is its quadratic computation time to calculate each individual residing probability. Even if faster processing speed can compensate the computation overhead, the collection of the entire service time distribution from every server will still consume significant system and network resources. To reduce computation time, we proposed to use aggregated statistical properties to predict the location of a mobile agent [5]. Our analysis and experiments showed that, under a variety of circumstances, the mean service time is a reliable index to predict the location of a mobile agent [5].

Adaptive Search Algorithms

The search strategies previously proposed are all static, which means that the search strategy is fixed when the search agent is dispatched 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 intelligent search agent finds out that the target agent has already visited and left the currently probed server, it will know that the estimation is not accurate. The search agent can use this departure time information to recalculate the residing probabilities. This may be better than blindly searching forward along the original planed path.

We proposed an adaptive search strategy that can change the basic algorithm during the search of a mobile agent [7]. This strategy makes use of the up-to-date information obtained from the servers that have been visited by the target agent. By knowing the departure time of the target agent from a server, the search agent can either recalculate the location prediction or simply switch from the original search strategy to the sequential search. When applying this strategy to the previously proposed blind and intelligent searches, we can obtain a significant improvement on the search efficiency.

A great challenge to design a good adaptive search strategy is to determine the proper condition to have the search agent switch from a search strategy to another. One simple approach is to estimate the current location of the target agent based on the elapsed time since the target agent left the currently probed server and make a gamble. Each search agent is parameterized with a gambling range. At each probe, it compares the gambling range with the estimated distance of the target agent, if the estimated distance of the target agent is within the gambling range, it will switch to another search strategy, which is simply a sequential search in our simulation study.

The gambling range can be either static or dynamic. *Static gambling strategy* uses a fixed gambling range regradless the length of the search list; while a *dynamic gambling strategy* changes its gambling range according to the current length of the search list.

Reference

1. 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.
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 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.
4. Yao-Nan Lien, et. al., ``FlyingCloud: A Mobile Agent Service Network'', *Proc. of the International Conference on Distributed Systems, Software Engineering, and Database Systems*, Dec. 1996, pp. 177-183.
5. Yao-Nan Lien, Fuhan Liu, Chun-Wu Leng and Wen-Shyen Chen, ``Intelligent Search of Mobile Agents'', *Proc. of the 1997 International Conference on Computer System Technology for Industrial Applications*, April, 1997, pp. 110-116.

6. Yao-Nan Lien, Fuhan Liu, Wen-Shyen Chen and Chun-Wu Leng, ``Asymmetric Binary Search of Mobile Agents '', *Proc. of the 1997 International Symposium on Multimedia Information Processing*, Dec. 1997.
7. Yao-Nan Lien, Fuhan Liu, Wen-Shyen Chen and Chun-Wu Leng, ``Adaptive Search of Mobile Agents '', *Proc. of the 1997 National Computer Symposium.*