Third Workshop on Mobile Computing
第三屆行動計算研討會
中華民國八十六年三月二十五、二十六日

An Open Infrastructure for Mobile Agents in Mobile Computing[†]

Wen-Shyen E. Chen, H.-T. Shu*, S.-T. Su*, C.-H. Du* and Yao-Nan Lien***

*Institute of Computer Science
National Chung-Hsing University
Taichung, Taiwan, ROC
email: echen@cs.nchu.edu.tw

**Department of Computer Science
National Chengchi University
Taipei, Taiwan, ROC
email: lien@cherry.cs.nccu.edu.tw

Abstract

Mobile Agents have been shown to be a promising approach to addressing the issues in ubiquitous computing: they have advantages in managing vast amount of information available in the fixed network, relieving the problems of slow, unrelibale connectivity, and limited capacity inherited in mobile computers. However, before the approach can be commercially viable, an extensive study of the issues involved in the management support to provide better quality of services to the user is needed.

In this paper, we study related issues of mobile agents and propose an open infrastructure for mobile agents in mobile computing. In addition, a prototype is implemented to demonstrate its feasibility. An examaple mobile agent that searches files on behalf of a client is also presented to show how the a service can be provided with such an infrastructure.

Keywords: Mobile Computing, Mobile Agent, Management, Network Infrastructure, Java

1 Introduction

Compared to the conventional computers with a fixed connection to wired networks, mobile computers have narrow, unreliable connectivity, limited processing prower and battery capacity, and have to operate in a dynamic, heteregeneous environment. As a result, to support *mobiles computing* [1-4], a new paradigm for information processing and communications is needed.

Mobile Agents [5-11] are shown to be promising in addressing many issues in ubiquitous computing. In this approach, an agent 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 automously through hetegeneous networks. The agent is capable of interacting with servers or other agents locally, moving to another server while carrying the intermediate results, and resuming execution when it reaches the destination. After the agent 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 appraoch, the mobile clients are decoupled from the servers 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.

The technical advantages of mobile agents are many: high bandwidth, support for disconnected operation, support for weak clients, ease of distributing individual service clients, semantic routing, scalability, lower overhead for secure transactions, and robust remote interaction. Other than providing support for existing network services, mobile agents are also likely to offer possible new services. Nevertheless, to make a mobile agent service network commercially viable, a high quality and cost effective agent mobility management system must be in place to guarantee a certain level of quality of service.

In this paper, we examine the issues of mobile agent infrastructure for mobile computing. In addition, a prototype of the infrastructure is implemented to illustrate the feasibility of this approach and to provide some insights for the future full implementation.

The rest of the paper is organized as follows. Section 2 gives an overview of the wireless communication and mobile computing. Section 3 introduces the open infrastructure we propose to support mobile agents in mobile computing. Section 4 discusses the implementation issues of the infrastructure. Section 5 presents the implementation of an example agent that runs in the prototype to search files in different hosts. Concluding remarks and possible future research topics are given in Section 6.

[†] This research was sponsored by MOEA and supported by Institute for Information Industry, R.O.C.

2 Overview

2.1 Characteristics of Wireless Communication

Wireless communication is much more difficult to achieve than wired communication because the surrounding environment interacts with the signal, blocking signal paths and introducing noise and echoes [1, 2]. As a result, wireless connections are of a lower quality than wired connections: lower bandwidth, high bandwidth variability, higher error rate, and more frequent spurious disconnection. They are also much more expensive than their wired counterparts. These factors can in turn increase communication latency due to retransmissions, retransmission timeout delays, error control protocol processing, and short disconnection.

2.2 Mobile Computing

Wireless networking/communication greatly enhances the utility of carrying a computing device. It provides the mobile user with versatile communication to other people and expedient notification of important events, yet with much more flexibility than cellular phones or pagers. Continuous access to the services and resources of the wired network is also made possible. The combination of networking and mobility will create new applications and services. However, the impairments of the underlying wireless communication infrastructure, along with the physical constrains of the mobile computers, handicap the establishing of this paradigm of computing [1, 2].

With the characteristics of wireless communication discussed in Section 2.1, if traditional protocols, such as TCP/IP, is used without modification, they may suffer much performance degradation or even cease to function when used directly in such an environment. For a mobile computing service to be accepted, all these factors must be taken into consideration and the underlying protocols have to be efficient, robust, be able to cope with disconnection more gracefully and work around them whenever possible. In the next section, we will propose an infrastructure that takes these factors into account to support mobile computing.

3 The Infrastructure

As discussed in Section 2, we need a new computing paradigm to address the issues in mobile environment. In this section, we propose an open mobile agent infrastructure, with the focus on the network trasport, to support mobile computing.

The infrastructure is illustrated in the Figs. 2 and 3, for moving and stationary user agents, respectively, to support ubiquitous computing. Note that the components in the infrastructures we propose here are "LOGICAL" entities, i.e., their physical locations might be implementation-dependent.

3.1 Entities in the Infrastructure

The major entities in the infrastructure are:

1. **User Agent**: It is invoked by a user submitting a request for services. It can either be stationary or mobile, depending on which is more feasible for the requested task. The user agent will first query the Broker Agent (described later in this seciton) to obtain the locations of the services/resources agents where the requested services can be fulfilled. It can either interact with service/resource agents via an Agent-Agent protocol (as shown in Figure 3) or be an mobile agent that moves to interact with the service/resource agents locally (as shown in Figure 2.).

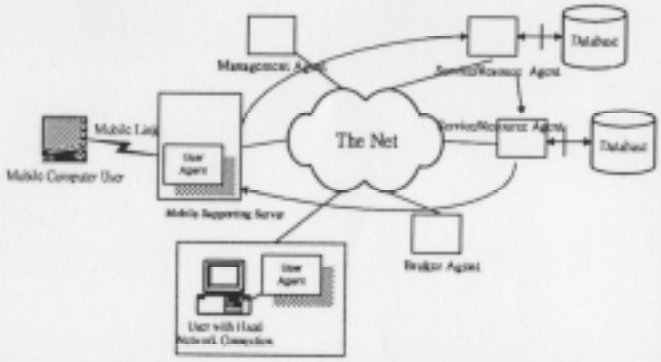

Figure 2. The Infrastructure with Moving User Agents.

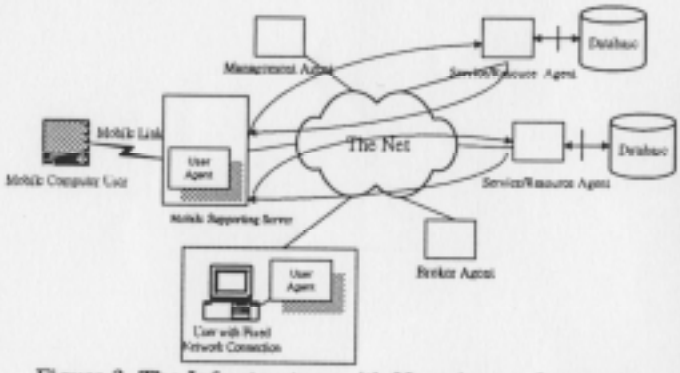

Figure 3. The Infrastructure with User Agents Interacting with Service Agents One by One.

2. **Mobile Supporting Server:** It is the server that compensates the limited capacity of the mobile computers and the unreliable, slow mobile link between the fixed network and the mobile user. The Mobile Supporting Server accepts requests from the mobile user and invokes an instance of the user agent on behalf of the mobile user to carry out the requests. The user agents in the Mobile Supporting Server hold the profiles of the mobile users. The Mobile Supporting Server also holds the final results and notifies the mobile user about their arrivals.

3. **Service/Resource Agent**: It is where the services or resources are provided for the user. When it first comes to service, or when there's a status change, the agent advertises the invocations/changes to the Broker Agent (described later in this seciton). For the infrastructure with the mobile user agent, it has to provide a transport mechanism for moving the agent to its next stop. It also needs to have a mechanism to deal with the heterogeneous databases it might have to provide.

4. **The Broker Agent**: It serves as a "bulletin board" where the service/resource agents advertise the services/resources they provide. The user agent will query the broker agent for the services it requests. The interactions between other entities and the Broker Agent are as shown in Figure 4.

5. **The Management Agent**: The management agent holds the status of the mobile agents and will respond to the queries from the user about the status of the agents he/she submitted. In addition, the user can instruct the Management Agent to carry out some action (such as suspend, terminate, etc.) against the mobile user agents. The interactions between other entities and the Management Agent are as shown in Figure 5.

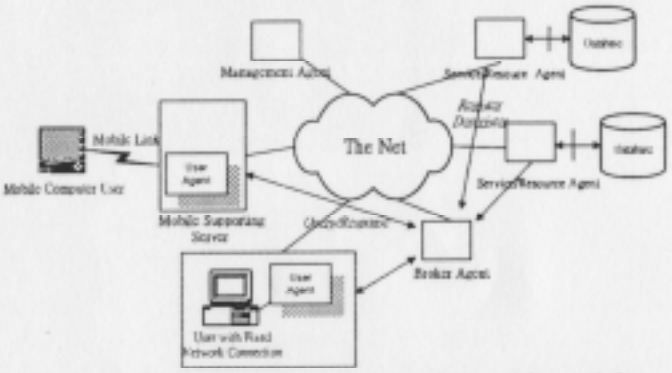

Figure 4. The interactions between other entities and the Broker Agent

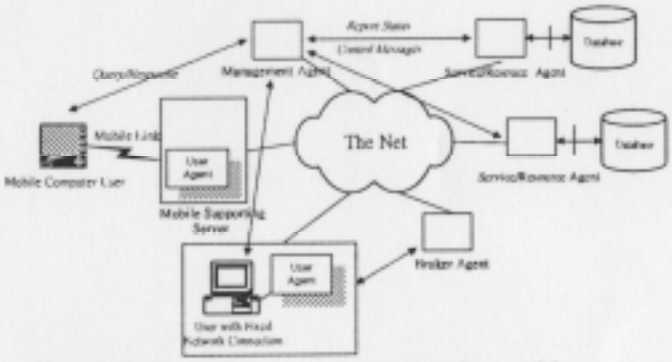

Figure 5. The interactions between other entities and the Management Agent

3.2 Related Issues

For the infrastructure to be feasible in providing the needed services to the users, some issues needs to be addressed. The issues and their possible solutions are summarized as follows.

- **Transport for Agents:** A transport mechanism is needed for the mobile user agent as moving the agent to interact with the service/resource agents locally will reduce the traffic in the network. The HTTP [12] protocol is not suitable as the client-server protocol does not meet the peer-to-peer interaction requirements for the agents interactions (a request might results in multiple responses). As a result, a new transport is needed. One possible solution is to adopt the Agent Transfer Protocol (ATP) [13] from IBM Tokyo Research Laboratory. (In our prototype implementation, we adopt the ATP with our own extensions.)

- **Service Agent Registration:** The interactions between the Service Agent and the Broker Agent need to be specified for the service agent to register the services/resources it provides.

- **Control and Management Functions:** The Management Agent needs to hold the status of the mobile agents and provide some control functions to the user agents. For this to be possible, the management agent need to have a way of locating the mobile agents and send control messages to it. To summarize, the functions it needs to provide are as follows.

  a. **Agent Location:** How to locate an agent in a service network?

  b. **Agent Status Report:** How to find out effectively if an agent is **running**, **hopping**, **suspended**, **frozen**, or **terminated**?

  c. **Agent Control:** How to communicate with an agent and control its execution in a service network?

  d. **Debugging:** How to trace the execution of an agent for debugging or auditing purposes?

  In addition, possible control actions that can be provided are:

  - **terminate:** Stop execution of release the resource held by an agent
  - **freeze:** stop the forwarding of an agent until a resume message is received.
  - **suspend:** suspends the execution of an agent until a resume message is received.
  - **resume:** resume the execution of a frozen or suspended agent.
  - **cancel:** cancel a previously submitted control message.

- **Information Abstraction and Filtering:** For the information gathered from the Service/Resource Agents, the user agent should have an information abstraction and filtering facility corresponding to the user profiles, especially for the mobile users where the mobile link cannot sustain high volume of traffic, to *distill* needed information from the results. In addition, a user profile will need to be consulted for the user agent invocation and information filtering functions.

- **Protocol between the mobile computer and the Mobile Supporting Server:** The conventional TCP/IP might not be feasible for the mobile environment as the frequent disruption in the mobile link resets the congestion control windows in the TCP/IP and results in low bandwidth utilization [15]. A new protocol for the wireless link is needed.

corresponding parameters and the intermediate results, and delivers the package to its destination.

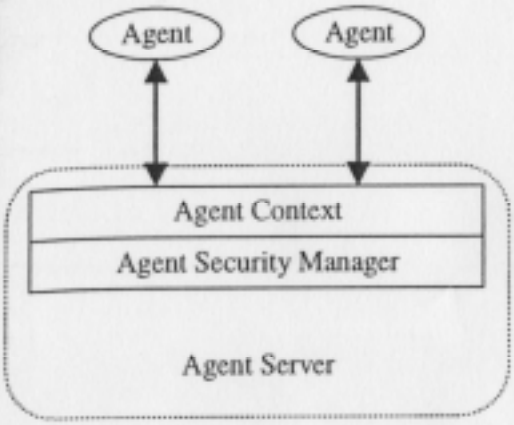

Figure 8. Mobility Enhancement to the Agent Server.

4.3 Implementation of the Agent Server

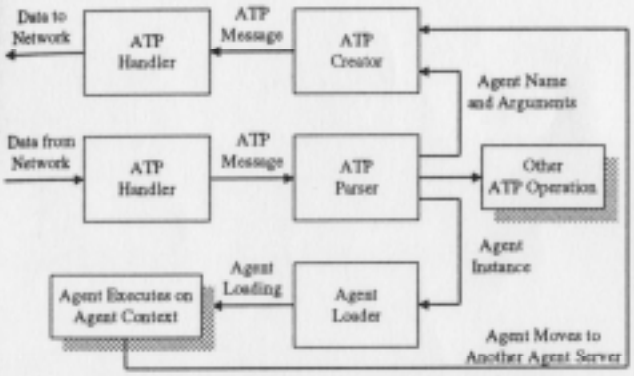

Figure 9. The structure of the Agent Server

The structure of the Agent Server is as shown in Figure 9. Whenever an Agent Server needs to dispatch an agent to another server, it will first passes the agent to the ATP Creator to package it into the ATP format; the package will be passed to the ATP Handler to be forwarded to its destination. On the other hand, an ATP message received from the network will first be passed to the ATP Handler, and be parsed by the ATP Parser to retrieve the agent code, parameters, and the intermediate results, if any. The agent is then instantiated, and be sent to the Agent Loader to set Agent Context and parameter values. It is then executed on the Agent Context.

For management purposes, each Agent Server has a Launch Table that stores information about the agents launched by the server. The format of the Launch Table is as shown in Figure 10. The Agent ID field records the unique identifier of the agent; the User ID field stores the ID of the user who submitted the agent; the Status field shows if the agent has been completed; and the Itinerary field records the "road map," the servers list, where the agent will visit.

Agent ID	User ID	Status	Itinerary
. . .	. . .	. . .	. . .

Figure 10: Launch Table of the Agent Server

Another table maintained by the Agent Server is Running Agent Table, as shown in Figure 11. Note that the Status field shows the status (as discussed in Section 3) being served by the corresponding Agent Server. The Thread Pointer field stores a pointer to the thread that the agent is executing on so that the Agent Server can apply some control, such as "suspension," over the agents. The fields Previous and Next are used to record the previous and next agent servers, respectively, in the corresponding agent's road map.

Agent ID	Status	Thread Pointer	Previous	Next
. . .	. . .	. . .	. . .	. . .

Figure 11: Running Agent Table.

4.4 Implementation of the Directory Server and Road_Map Server

The Directory Server and Road_Map Server together hide the locations of the Agent Servers that provide the requested services from the agents. As a result, an agent does not need to travel to all of the available Agent Servers to carry out the requested services.

The Directory Server provides register, query, modification, and de-register services to the Agent Servers. In our architecture, the structure of the Directory Server is as shown in Figure 12.

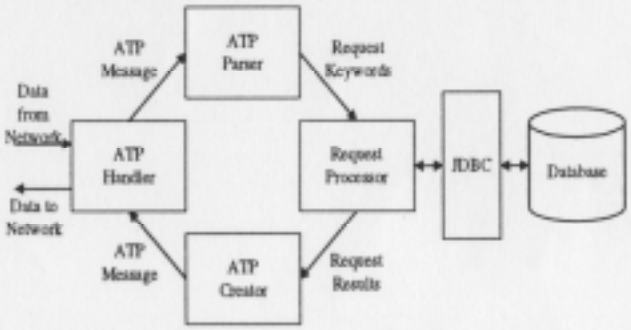

Figure 12. The Structure of the Directory Server.

Note that the Directory Server itself is an Agent Server and it shares the common parts, such as ATP Handler, ATP Parser, and ATP Creator, with the Agent Server. The Request Processor accepts the services requests and interacts with the directory information stored in the Database through the JDBC interface.

After receiving a request submitted by a mobile user, the Mobile Supporting Server will first consult with the Directory Server to get the information about which servers to submit the agents to accomplish the requested services. However, since the requested service might have to be performed at the Agent Servers in some specific order, the Road_Map Server will construct a "map," according to the results received from the Directory Server and some user profiles, as an indication of the list of servers the agent should visit. The servers list will be a part of the agent parameters.

4 Implementation Issues

We have presented in the previous section the conceptual design of the infrastructure for mobile agent in a mobile computing environment. In this section, we describe our implementation of a prototype to realize the architecture. The prototype consists of five major components: Agent Server, Road_Map Server, Management Server, Agent Transfer Protocol, and Agent. The relationships among them are illustrated in Figure 6.

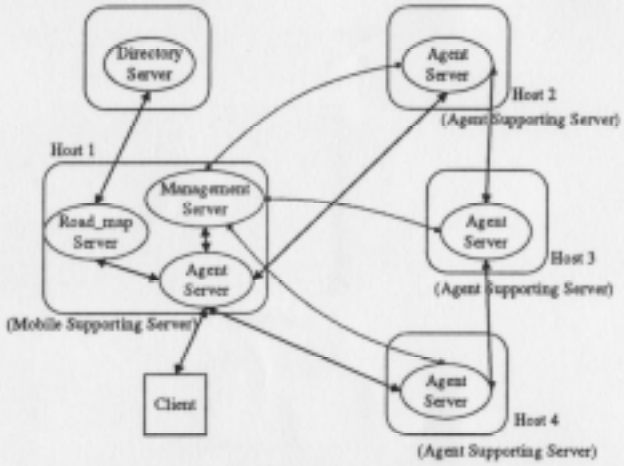

Figure 6. The components of the prototype implementation.

4.1 The Life Cycle of an Agent

We use an example scenario (as shown in Figure 7) to illustrate the life-cycle of an agent.

1. The user selects a template from a set of service templates from the WWW Server in the fixed network. The template is a homepage that has a corresponding applet in it. The user then can input the needed information for the applet.

2. The applet then sets up a connection with the Agent Server at the Mobile Supporting Server (the Host 1 as shown in Figure 7), passing the user input to the Agent Server, and requesting the Agent Server to generate a corresponding agent on behalf of the client. After the agent is created, the user can then disconnect from the fixed network. The Agent Server will contact its Road_Map Server, which in turn will consult with a Directory Server, to query about which hosts the agent should be sent to in order to carry out the assigned task. The Road_Map Server will then build a servers list and add the information to the parameters of the agent. As an example, Figure 7 shows that the agent will visit Hosts 1, 2, 3, and 4 in sequence .

3. The Agent Server at Host 1 creates an execution environment for the agent, and starts the execution of the task specified in the agent. After the corresponding task is finished, the agent reports its status to the management server, records its next destination and its status in a local database. The Agent Server then packs the agent, along with the intermediate results it obtains at Host 1, and delivers the agent to Host 2.

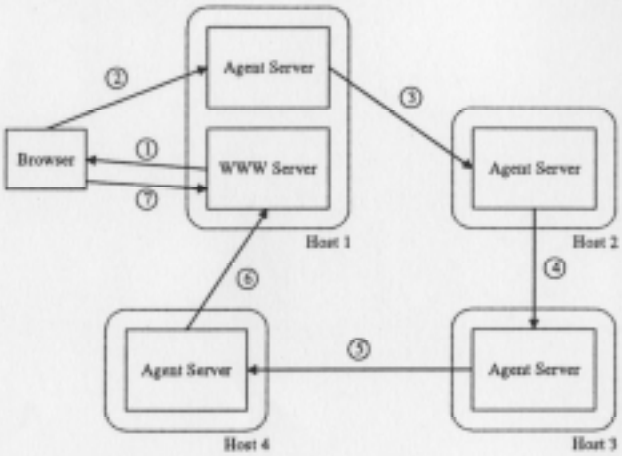

Figure 7. The Life-Cycle of an Agent.

4.-5. The Agent Servers at the Hosts 2 and 3 perform the same procedure as specified in step 3.

6. The agent finishes execution at all specified hosts and carries the results back to the Host 1, the Mobile Supporting Server where the agent first started. The agent is then terminated, the results stored. A notification is then generated and a mail message (in HTML format) sent to the user's account, indicating where the results can be retrieved.

7. The user then uses a browser to connect to the WWW server at the Mobile Supporting Server to retrieve the results.

4.2 Language Selection

We choose Java [14] as the language to implement the mobile agent environment. The decision is based on the facts that Java is similar in syntax to C++, the Java compiler generates byte codes for the Java Virtual Machine, which is platform-independent, and the runtime is typesafe and supports a form of secure loading to dynamically add code from other sources. In addition, the Java applet is closely coupled with the browser, making it easier to implement the interface between the mobile user and the Agent Server.

However, Java by itself does not support the mobility as described in Section 3.1, where the agent moves from one Agent Server to another. Since mobility is important in the infrastructure, in the prototype, we add a mobility enablement module to the Agent Server. The structure of the mobility enhancement is shown in Figure 8. Note that the mobility support is implemented in the Agent Context module (the interface between an agent and the Agent Server) by a Java method called "Move". Whenever an agent requests a service, such as reading a local file, the request will be forwarded to the Agent Context and verified by the Security Manager. If the request is legal, it is carried out by the Agent Server. When an agent needs to move from the current agent server to the next server, it will invoke the "Move" method; the Agent Server will package the agent along with the

4.5 Implementation of the Management Server

The Management Server is used to provide management supports, as discussed in Section 3, for the mobile agents. To retain controls over the agents are submitted into the network, the user can request management functions from the Management Server. To enable the management functions, the agent needs to located, and the Agent Servers have to perform the requested management functions.

There are several methods to locate the mobile agents. For a detailed description of the agent location methods, the reader is referred to [16].

4.6 Implementation of the Agent Transport Protocol

To support agent mobility, a protocol to transport the agent in between servers is needed. We implement this application-layer protocol base on the Agent Transport Protocol (ATP) [13] specification developed by IBM Tokyo Research Laboratory. The ATP is to provide a standard for transport of agents in different platforms. (Note that the underlying transport protocol is still TCP/IP.)

At the time of this writing, the ATP is of version 0.1. It covers the naming of the agent services, format of agent identifier, agent transport, and agent protection (with *cookie* mechanism). In addition, the ATP protocol adopts the request/reply mechanism and defines three types of request: dispatch, retract, and fetch. For a detailed description of the semantics of the request, refer to [13].

However, the three request types cannot meet the requirements to support the management functions defined in Section 3; For example, an agent should be able to be suspended or terminated by the user, and Agent Servers should be able to deliver an agent among them without intervention from the Mobile Supporting Server where the agent is first created. In addition, we should allow the user to *dispatch* an agent directly into the network, a function not defined in the original ATP protocol. We propose to add the following actions to the message body:

1. **launch**: In the infrastructure, the user will first contact the Mobile Supporting Server for a template, which is used to "launch" a request to the Server. The server will then generate a corresponding agent for the user.

2. **suspend**: An Agent Server can suspend the execution of an agent in another Agent Server by sending a "suspend" request to that server. The message body should include the agent and user identifiers. The target server should return a response.

3. **restart**: An Agent Server can send a "restart" request to another Agent Server to restart the previous suspended agent. The message body should include the agent and user identifier. The target server should return a response.

4. **report**: When the task is done, the last Agent Server in the road map of an agent will issue a "report" request to the Mobile Supporting Server where the agent was created. The request should contain the results. The Mobile Supporting Server will issue a response after the request is received.

5 Example Agent Implementation

In our prototype implementation, we have created an abstract class, called Agent.java, for agents. In the abstract class, setAgentContext() is used to set the agent context at an agent server, and setArguments() is used to set the parameters of an agent. In addition, to test the feasibility of the prototype, we have implemented a simple SearchFile agent, which inherits the Agent.java abstract class, to search files in several machines. Note that the current implementation of SearchFile agent takes a file name as a parameter and does not search the contents of files. However, the extension to include content search should be trivial.

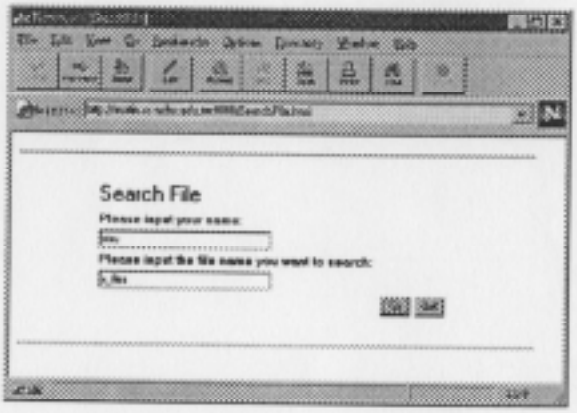

Figure 13: The User Interface of the SearchFile Agent.

To search files located in different machines, the user can download an applet from the WWW server (in the Mobile Supporting Server) and fills in the name of the file to be searched. This information will be passed to the SearchFile agent when it is created. The agent will then traverse the predetermined list of Agent Servers to see if the file resides in them. The collected information will then be reported back to the Mobile Supporting Server in HTML format. The user can then retrieve the result using a WWW browser. Figure 13 shows the interface presented to user when the user requests to search files. Figure 14 depicts the results reported by the SearchFile agent. Note that the agent has traversed four Agent Servers to collect information about the specified file.

6 Conclusion

Since mobile computers do not have a high-bandwidth, reliable connection to the Internet and are limited by small memory and display area. As a result, to provide the mobile computers with the compatible power to that of the stations with fixed connections so that they can

participate in the information dissemination in the Internet will be an important problem to address.

In this paper, we propose an open infrastructure for mobile agents to allow mobile users to access the vast amount of information available in the fixed network. It is believed that the a mobile computer can better cope with its physical limitations and frequent link disconnection with the infrastructure. A prototype system is used to illustrate the feasibility of the infrastructure and to provide some insights for the future full implementation.

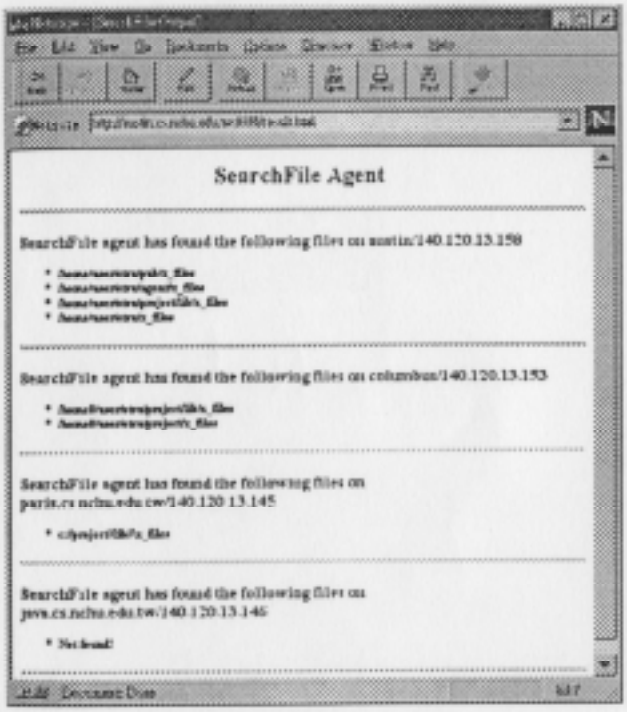

Figure 14: The Result Reported by the SearchFile Agent.

References

[1] H. Forman and J. Zahorjan. "The Challenges of Mobile Computing," *Technical Report UW CSE 93-11-03*, University of Washington, March 1994.

[2] T. Imielinski and B. R. Badrinath, "Mobile Wireless Computing: Challenges in Data Management," *Communications of the ACM*, August 1994.

[3] M. Weiser, "Some Computer Science Issues in Ubiquitous Computing," *Communications of the ACM*, July 1993.

[4] M. Weiser, "Ubiquitous Computing," *IEEE Computer*, October 1993.

[5] M. R. Genesereth, and S. P. Ketchpel, "Software Agent," *Communications of the ACM*, vol. 37, no. 7, July 1994, pp. 48-53.

[6] K. Ousterhout, "Script And Agents: The New Software High Ground," Invited talk, USENIX Conference, 1995.

[7] K. D. Kotay and D. Kotz, "Transportable Agents," *Proc. of the Third International Conference on Information and Knowledge Management*, December 1994.

[8] S. Gessler and A. Kotulla. "PDAs as Mobile WWW Browsers," *Proc. of the Second World Wide Web Conference*, October, 1994. URL: http://www.ncsa.uiuc.edu/SDG/IT94/Procedings/DDay/gessler/www_pda.html

[9] D. Chess, *et al.* "Itinerant Agents for Mobile Computing," *IEEE Personal Communications Magazine*, October 1995.

[10] W.-S. E. Chen and Y.-N. Lien, "Intelligent Messaging for Mobile Computing over the World-Wide Web," in *Proceedings of the Second International Mobile Computing*, Hsin-Chu, April, 1996.

[11] W.-S. E. Chen, Y. N. Lien and W. Y. Hsien, "An Infrastructure for Mobile Computing with Intelligent Messaging: Implementation Issues," In *Proc. of the 1996 Workshop on Distributed System Technologies & Applications*, pp. 270-277 , May 1996.

[12] T. Berners-Lee, "Hypertext Transfer Protocol (HTTP)," IETF Internet Draft (work in progress), 1994.

[13] IBM Aglet WorkBench, URL: http://www.trl.ibm.co.jp/aglets

[14] "The Java Language Environment: A White Paper," Sun Microsystems Computer Company, May 1995.

[15] R. Caceres and L. Iftode, "Improving the Performance of Reliable Transport-Layer Protocols in Mobile Computers," *IEEE Journal on Selected Areas in Communications*, 13(5), pp. 850-857, June 1995.

[16] Y.-N. Lien and C.-W. R. Leng, "On the Search of Mobile Agent," in *Proc. of the 7th IEEE Symp. Of Personal, Indoor, and Radio Communications*, pp. 703-707, October 1996.