一九九六分散式系統技術及應用研討會
1996 Workshop on Distributed System Technologies and Applications

A Mobile Agent Service Network Prototype

Yao-Nan Lien, Yenlin Yin, Fuhan Liu, and Yuli Hwang

Department of Computer Science, National Chengchi University
Taipei, Taiwan, R.O.C.

Abstract

To make information ubiquitously available to the people in the world requires not only the Information Superhighway, but also a non-traditional computing paradigm, such as the intelligent messaging, to overcome the intermittent connection problem inherent in a mobile environment. This paper describes the development of a mobile agent service network prototype currently undergoing in the National Chengchi University. The main objective of this prototype is to simulate a real operational mobile agent service network, to analyze the network behavior, and to exercise our solutions. The system will be developed according to the previously proposed open architecture [15] and hybrid mobility management infrastructure [16].

1. Introduction

1.1 Agent and Agent Mobility

Because of the advance of computer and communication technologies as well as the promotion of National Information Infrastructure (NII), the progress of *mobile computing* is accelerating into a revolutionary speed making the dream of ubiquitous information service a reality [3,4,14,22,25]. The goal of a ubiquitous information service network is to provide information to the users anytime anywhere. To accomplish this goal, the service network must be supported by some ubiquitously available communication networks and be able to conveniently access to various information resources. Currently, wireless communications networks such as AMPS or GSM cellular networks will be able to support the ubiquitous communication requirement. As for the convenient information services, distributed computing technology seems to be an ideal computing paradigm. In a distributed system, all servers in a network are integrated into a single logical server such that clients can access the network resources transparently by interacting with a single server. Unfortunately, applying distributed computing technology in such a scale and heterogeneity will have to take a much longer time to realize in the real world.

Thus, clients will have to access network resources in a prescriptive fashion by interacting with individual servers step by step to accomplish a complicated task. However, in most mobile computing environments, the nature of communications is intermittent and the battery energy is limited. Thus, it is very difficult to accomplish a complicated task that requires its client to interact with multiple servers intensively. A non-traditional computing paradigm, the *intelligent messaging*, that allows clients to interact with multiple servers in a dynamic fashion has been brought up to cope with this problem [7,9,10,12,13,18,26].

Simply speaking, an *intelligent message* is an electronic message that carries a computer program, whether 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 forward automatically the message itself to another server, on which the program is executed continuously in a pipeline fashion. Such a message is also known as an *intelligent agent* in other fields [7,11]. For simplicity, it is called an *agent* in the rest of this paper. Good examples can be found in [14,15].

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

1.2 Client Mobility

The telecommunication network technology is moving toward providing more mobility and flexibility to the users. The recently emerging Personal Communication Services (PCS) technology makes use of the Intelligent Network technology to push the telecommunication technology further. It provides a higher degree of mobility to the users such that a user can move across different telecommunication networks, referred to as the *personal mobility*, and can be independent of terminal devices, referred to as

the *terminal mobility* [17,19,27]. In [16], Lien proposed a new mobility, the *client mobility*, that allows users communicate across different logical networks. For instance, a cellular phone will be able to send a message to or to receive messages from the Internet by its phone number.

1.3 Mobility Management

To make a service network commercially viable, it is essential to have a high quality and cost effective operation, administration, and maintenance system (OA&M) in place to guarantee a certain service quality. Followings are some critical OA&M problems with respected to the mobility management raised in [15]:

1. How to locate an agent in a service network?
2. How to locate a client?
3. How to know the status of an agent?
4. How to control the execution of an agent that is traveling in a service network?
5. How to trace the execution of an agent (e.g. for debugging or auditing purposes)?

They are by no means exhaustive. Furthermore, there are other issues such as transaction and security supports needed to be addressed as well [15]. The initial version of our prototype will facilitate the study of the agent and client mobility management.

1.4 Open Service Network Architecture

Traditional telecommunication networks such as PSTN and 800 Toll-Free service used to take considerable resources and long deployment duration to establish. One major resource drain in such networks is the OA&M (and provisioning in some cases). It will be impractical to demand the comparable resources to support the OA&M functionalities in many perspective information services. The computing community will have to rely on themselves, rather than the telecommunication community, to develop and deploy the demanded functionalities. All infrastructures and solutions must not requires any change to the existing telecommunication network. To achieve this, we employ the open service network architecture proposed in [15], which separates service networks from transport networks. It also allows services of any scale and any quality to be introduced into the network easily. Service providers can choose whichever operation model and the quality level based on the resources available to them. That architecture is summarized as follows:

Basic entities are servers each providing a specific information service. They are connected by various logical or physical communication networks such as the Internet, the PSTN, or the ARDIS radio network. The networks that provide connectivity services between servers and clients are referred to as the *transport networks*. Any number of server can be integrated together to form a *service network* providing higher level services to their clients. A server can participate into more than one service network. A *terminal* is a physical device that allows a client to interact with a transport network (and service networks). A terminal could be a telephone, a Personal Digital Assistant (PDA), a desktop PC, or a workstation, etc. (A terminal is associated with a transport network, while a client is associated with a service network. There is no fixed relationship between a client and a terminal.) This architecture provides the transport network transparency as well as the required flexibility to the service networks. Readers are referred to [15] for details.

1.5 Hybrid OA&M Supporting Mode

Centralized OA&M support is easier to achieve higher service quality. However, it suffers higher operation cost and long deployment time duration. On the other hand, distributed support is usually more flexible in introducing new services and has much lower operation cost, but it suffers lower service quality. The open service network architecture accepts both approaches. However, many perspective information services will not be able to afford the expensive centralized OA&M support, while they need certain level of service quality beyond what a distributed approach can offer. In [16], Lien proposed a hybrid OA&M supporting structure that can take advantages of both approaches. In that hybrid approach, all OA&M functionalities are classified into two categories: distributable and non-distributable. Distributable functionalities can be supported by any server in the network or even client's own resources. Non-distributable functionalities must be supported by

designated servers. Those critical and more appropriate to be managed centrally, such as security and billing functions, are classified as non-distributable and must be managed centrally. It is yet to be researched to define and to classify all OA&M functions. Many perspective services will be benefited from this hybrid OA&M supporting model.

Judging from the fact that the Internet users are growing in an exponential rate, most of the mobile computing users in the future will have stationary access to the Internet from their offices or homes. These facilities are called *Home Base Nodes* (HBNs). To further reduce operation cost, Lien proposed to use these personal facilities to help managing service networks [16]. A user can choose to use the facilities provided by the service providers or his/her own HBN to manage the client and agent mobilities.

2. Infrastructure for Mobility Management

2.1 Basic Definitions

In this section, we describe the overall infrastructure that our prototype will adopted [16]. This infrastructure allows personal Internet facilities to participate in the management of client and agent mobilities. For simplicity, we assume clients and agents can only travel within the Internet in our initial prototype.

2.1.1 Agent Identification

Every agent must have an identification that is unique with respect to a particular service network. A simple two-segment ID structure, (client-ID, message-ID), would be sufficient in many cases, where the message-ID is a unique sequence number generated by the client.

2.1.2 Agent Status

The following is the list of agent status observable by the external entities such as clients or other authorized entities:

- *running* - an agent is being executed by a server. Please note that the running state defined here is different from what is defined in the traditional operating systems. A running agent here may actually be blocked waiting for local resources or is waiting for an external message such as the input from its originating client. Since a blocking state is not observable by the external entities, it is still considered in the running state. However, for reference purpose, we call such an agent to be in *spinning* state.
- *hopping* - an agent is being forwarded to another server.
- *terminated* - an agent is terminated.
- *suspended* - an agent is suspended in the middle of or before an execution by an authority external to that agent.
- *frozen* - an agent in a server is not being executed, but is waiting to be forwarded to another server.

The difference between "spinning" and "suspended" is that a spinning agent can resume its execution by itself without any external permission while a suspended agent can't. (Even if a spinning agent is waiting for something, it can always resume itself if it decides not to wait.)

2.1.3 Current Execution Node (CEN)

- *Current Execution Node* - the node that an agent currently resides.
- *Registered Current Execution Node* - the current execution node of an agent shown in its status holder, which will be explained later in Section 3.4.

2.1.4 Execution Log

The execution status of an agent such as which node it came from and which node it was forwarded to may have to be saved in each of the servers visited by that agent. This is especially useful for tracing purpose.

2.2 Facilities for Mobility Management

2.2.1 Network Management Center (NMC)

A central facility is needed to support all non-distributable management functions as well as distributable functions if it is needed. Typical OA&M functions are client and server registration, authentication, name server, coordination, or client specific services. Although, it looks like a single node, it may actually be a number of nodes distributed over a network. For simplicity, we assume a NMC is a single node.

In general, a commercial service network usually needs a NMC for such functionalities as billing, authentication, security, etc.

2.3 Home Base Node (HBN)

Even though a mobile client may change its locations from time to time, it usually has a home location and a most frequently used Internet access point such as a personal account on an Internet-connected system or a PC, called *Home Base Node*. This infrastructure proposes to make use of HBN to share the OA&M workload. When a client subscribes to a service network, his/she can choose to use or not to use his/her own HBN. A lot of overhead in managing a service network can be saved by using HBNs. A HBN can also offer auxiliary computing resources to help mobile terminals to cope with their residential resource limitation. Readers are referred to [16] for details.

In fact, if a large portion of the workload is performed on the HBN itself, a mobile terminal can be treated as an intelligent terminal for that HBN.

2.4 Status Holder

The *status holder* for an agent is a place to store the status of an agent so that the client or other authorized entities can access this information easily. It can be any node such as client's HBN or the original access node, or even the NMC itself. A system may require each agent to report its status to its status holder on the designated events such as arrive-a-node, suspended, frozen, etc. A client can choose whatever the events his/she is interested when he/she submits an agent. An alternative status holder may be needed when the availability of the original status holder is a concern. A client can even request an agent **not** to report its status to save communication cost. These all depend on implementation details.

3. Location Support

One of the most important issues in mobility management is the ability to locate an entity such as an active agent or a client. In this section, several solutions will be proposed based on the infrastructure shown in the last section.

3.1 How to locate a client?

Due to the separation of service and transport networks, the client location problem must be handled by the service networks themselves, not the transport networks. In the proposed hybrid OA&M infrastructure, the facilities available for mobility management are quite rich as compared to the traditional PCS approach. Both NMC and HBNs are available for keeping track of client locations. By distributing the location registration functionalities to these facilities, the traffic to the NMC can be significantly reduced, although it may take more messages and longer time to locate a client.

A straightforward scheme is very similar to the HLR/VLR scheme in PCS. Each moving client registers his/her current location in his/her HBN or NMC. When locating a client, an inquiry is sent to the NMC to obtain the client's HBN, then another inquiry is sent over there to obtain the registered location.

3.2 How to locate an agent?

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

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

Therefore, better ways to locate an agent are needed. The concept of *status holder* is specially useful to support such a need, which will be shown in the following subsections.

3.3 Chase-from-holder Algorithms

When using a Chase-from-holder algorithm, a search agent will visit the status holder of that client first to inquire the registered location of the target agent and then route to that node. If the target agent is found in that node, the search agent sends the result back to

the client and terminates. Otherwise, it uses a trace algorithm to chase the original agent along the original route.

3.4 Status Holder Independent Search Algorithms

When a status holder is not available for search, or when the agent being searched losts contact with its status holder, a blind search is inevitable.

3.4.1 Binary Search Algorithms

A search algorithm similar to the binary search in searching a value in an array of data may be good for blind search. It can only work in deterministic routing where the expected visiting nodes as well as the visiting sequence are known in advance.

3.4.2 Execution Time Dependent Search Algorithms

If a client has a good estimation on the service time in each step, he/she may have a good guess on the current location of an agent. By using this information in a search, it may be able to reduce the search time significantly.

3.5 Related Issues

3.5.1 Exact Search vs. Approximate Search

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

3.5.2 Non-deterministic Execution

The search algorithms presented so far all assume that the agent to be searched traveled along a predetermined path. It may not be the case when the path it took depends on real time conditions and thus, is non-deterministic. In this case, the chase-from-holder algorithms are more appropriate. It is yet to be researched if the status holder is not available.

3.5.3 Lost Agent

An agent may get lost due to a failure in the agent itself, the server it resides, or the network it travels such that no search agent can find its status. Sometimes it may have no harm to just ignore the lost agent. However, it may be necessary to recover the lost agent in some critical situations such as business transactions. This problem is further complicated when concurrent execution is allowed because it is difficult to know whether all child agents are alive and all finish their assignments. Further research on this issue is needed.

3.5.4 Race Conditions

Since the target agent may be moving while a search agent is looking for it, a non-sequential search may fail to locate an agent. Therefore, it is essential for a search algorithm to avoid all possible race conditions. One of the main objectives of this prototyping experiment is to analyze all possible race conditions.

3.6 Concurrent Search

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

Open issues are

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

4. Status Inquiry

The most straightforward way to inquire the status of an agent is to send a search agent together with a status-inquiry agent. On the other hand, if an agent always reports its status to its status holder, its status can be easily obtained right from there. Again, because the execution of an agent continues to progress, the exact status of an agent is not easy to obtain due to the potential message traveling latency. One can send the status inquiry agent directly to the current execution node or let the status to be sent directly to the client (with higher communication cost) rather than its status holder if the situation is permitted. One has to balance the accuracy and the

2. The system will be designed with flexibility so that each of its components can be easily replaced.

3. The system will be designed evolutionally that only the simplest platform will be implemented initially and will be evolved gradually into a more complete system.

6.3 Server

There are plenty of daemon sharewares available over the Internet that are activatable by email messages and can be easily modified to fit our need. We will focus on the internal design of a server.

The first design consideration is the management of sharable information such as clients profiles, execution log, agent status, etc. Considering the reliability objective, distributed database is rejected and decentralized database will be used. Each server maintains its own local information such as execution log. Client information will be centrally maintained and distributed to all other sites. Considering its low update frequency, replicating client information to all servers won't be too much a burden. Nevertheless, in our initial design, the authentication process will be carried out by the Home Base Node of each client. To simplify the registration process, each client except the network manager can only register to a server (and will have only one authentication server for each client), all requests that need to be authenticated will be forwarded to its HBN first. In the future design, authentication will be distributed to all other servers in the same service network to improve the accessibility preventing a client from being rejected by the network due to a failure occurring in its HBN.

The network manager who maintains the agent network using a management system must be able to access to all other servers. It can be seen as a *super-client* who has registration on all servers.

Normally, the path taken by an agent is dynamically defined by the script contained in the agent. The agents will travel automatically in the network while the script is being executed. However, an agent can also travel through a predefined path with much less overhead. A typical example is an agent that performs routine network management functions such as server status collection. (Note that although a network manager, with a risk of traffic congestion, can broadcast messages to all servers to collect their status simultaneously, it can also send an agent to collect the status of all servers sequentially without risking traffic congestion.)

6.4 Agent Script Language

Designing a sound and complete script language can never be a trivial task, especially for such a newly emerging paradigm. Currently, there are two competing standards, tcl/tk and Java. Neither of them is mature enough to be accepted by most of the users. Our initial prototype will have to create a higher level of abstraction that can be built based on either one.

6.5 Agent

6.5.1 Agent Status

As defined in Section 2.1.2.

6.5.2 Agent Resource Control

Similar to a process in an operating system, each agent is allocated with some resources, such as size in byte and execution cycle time. There must be some control mechanism to prevent an agent from overruning the allocated resources. There are plenty of resource control policies that can be borrowed from operating system area. This system will focus on the problems that is unique in our environment.

One way to avoid information overflow in retrieving large amount of information from a server is to transfer the retrieved information back to the client immediately before traveling to the next node.

6.5.3 Agent Log

In order to trace the execution history, the execution of an agent will be logged in each server it visited as well as in the agent itself. Basic logged information in servers is as follows:

- client ID (optional if it is part of agent IDs)
- agent ID
- arrival time
- departure time
- resource usage

cost according to his/her own need.

5. Execution Control

After an agent is submitted, a client may want to regain the control of its execution. For example, if one find that the balance in his/her bank account is not enough to pay the bill for a submitted shopping agent, or he/she may simply change his/her mind, he/she has to terminate that agent before it is too late. The defined control functions are:

- *terminate* - kills an agent;
- *freeze* - postpones the forwarding of an agent until a resume message is received;
- *suspend* - suspends the execution of an agent until a resume message is received; and
- *resume* - resumes the execution of a suspended or frozen agent.

There is an open issue: what if a frozen or suspended agent never gets resumed or terminated? Some kind of garbage collection function must be provided to control these orphan agents.

5.1 Look-Ahead Control

A client may want to send a control agent to a node ahead of time before the target agent arrives that node. For example:

- to change the original plan, or
- advanced freeze or suspension.

The control function will be performed when the target agent arrives the node. The defined look-ahead control functions are:

- *terminate* - kills the execution when the target agent arrives;
- *freeze* - executes the agent when the target agent arrives, but postpones the forwarding of the agent to the next node until a resume message is received;
- *suspend* - suspends the execution of the target agent immediately when it arrives at the node until a resume message is received;
- *cancel* - cancels a previously submitted control request.

There are some exceptions: a target agent may never reach the node such that the look-ahead control agent may not get a chance to apply and becomes an orphan eventually. Some kind of garbage collection must be provided.

Another interesting exception is that a look-ahead control agent may arrive a node later than the target agent. (This can be detected from the execution log.) In that case, a chase algorithm must be activated to chase the target agent along its traveling trace.

6. Current Implementation

Intelligent messaging paradigm is such a newly emerging area that we do not have sufficient knowledge about its behavior under all possible operation conditions, especially in a real operation system. Thus, there is a need to create a simulated environment to facilitate further study in various critical issues mentioned above. A project is currently undergoing in NCCU to develop an agent mobility management network prototype. The platform will consist of a set of servers capable of intelligent messaging support, a script language, and an agent management system.

6.1 Transport Mechanism

To compliant with our open architecture and to simplify the implementation, email system (MIME protocol) over the Internet is adopted in the initial design. Servers, clients, and agents all communicate with each other through Email. Every agent is wrapped within an email message. However, the system will be designed with necessary flexibility that the underlying transport network can be easily replaced if a better mechanism is available.

6.2 Design Philosophy

In addition to the architecture principles we proposed in [15], the system will be designed with the following guidelines:

1. In order to maintain the quality of service, system reliability is the main objective with highest priority. (As a consequence, most components in our system will be as simple as possible.)

- next node
- termination condition

6.6 Management System

The mangement system acts as a super client using the same Email protocol to communicate with all servers. Major managerial functionalities are agent control and network monitoring. Detailed functionality are described in Section 3-5.

7. Summary

A ubiquitous information service environment needs to offer the client mobility allowing its client to move across different logical networks. To overcome the intermittent connection problem inherent in mobile environments, it also needs to offer the agent mobility allowing its clients to access network services by sending an intelligent message to cruise the network. To guarantee the service quality for a service network, an OA&M system is needed to manage both mobilities.

The OA&M system are often to be the most expensive and most complex system module in supporting a service network. Due to a lack of real experience in operating an ubiquitous information service network, there are many problems related to this new computing environment yet to be studied. This paper describes the prototyping experiment undergoing in the National Chengchi University. This prototype is designed based on the previously proposed open architecture that is independent of physical communication networks and has greatest flexibility for introducing new services. It also employees the hybrid management mode proposed in [16] to incorporate personal Internet facility for reducing OA&M cost. This experiment will allow us to observe this new computing paradigm more closely.

References

1. T. Alanko, et al., "Measured Performance of Data Transmission Over Cellular Telephone Networks," *ACM SIGCOMM Computer Communication Review*, pages 24-44, August 1994.
2. J. F. Bartlett, "W4 - The Wireless World Wide Web," *Proc. of the IEEE Workshop on Mobile Computing Systems and Applications*, December 1994.
3. T. Berners-Lee, et al., "The World-Wide Web," *Communications of the ACM*, vol. 37, no. 8, August 1994, pp. 76-82.
4. T. Berners-Lee, R. Cailliuau, J.-F. Groff, and B. Pollermann, "World-Wide Web: The Information Universe," *Electronic Networking: Research, Applications and Policy*, vol. 1, no. 2, Meckler, Westport, CT., Spring 1992.
5. G. H. Forman and J. Zahorjan, "The Challenges of Mobile Computing," *IEEE Computer*, March 1994, pp. 38-47.
6. J. Garrahan, P. Russo, K. Kitami, and R. Kung, "Intelligent Network Overview", *IEEE Communications*, March 1993, pp. 30-36.
7. Michael Genesereth and Steven Ketchpel, "Software Agents", *CACM*, July 1994, pp. 48-53.
8. S. Gessler and A. Kotulla, "PDAs as Mobile WWW Browsers," *The Second World Wide Web Conference '94*, October, 1994.
9. T. Imielinski and B. R. Badrinath, "Mobile Wireless Computing: Challenges in Data Management," *Communication of ACM*, August 1994.
10. M. F. Kaashoek, T. Pinckney, and J. A. Tauber. "Dynamic Documents: Mobile Wireless Access to the WWW," *Proc. of the IEEE Workshop on Mobile Computing Systems and Applications*, December 1994.
11. K. D. Kotay and D. Kotz, "Transportable Agents," *Proc. of the Third International Conference on Information and Knowledge Management*, December 1994.
12. James Lee, "Intelligent Messaging Paradigm and Telescript," *CCL Technical Journal*, No. 35, Dec 1, 1994, pp. 37-41.
13. Ichiro Lida, Takashi Nishigaya, Koso Murakami, "DUET: An Agent-Based Personal Communications Network", *IEEE Communications*, Nov. 1995, pp. 44-49.
14. Yao-Nan Lien, "Perspective of Service Networks on National Information Infrastructure," *CCL Technical Journal*, No. 35, Dec 1, 1994, pp. 28-36.
15. Yao-Nan Lien, "Client and Agent Mobility Management," *Proc. of the Second Workshop on Mobile Computing*, Hsing-Chu, Taiwan, March 1996, pp. 141-152.

16. 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.

17. Yi-Bin Lin, "Determining the User Locations for Personal Communications Services Networks", *IEEE Transactions on Vehicular Technology* , vol. 43, no. 3, Aug. 1994, pp. 466-473.

18. P. Maes, "Agents that reduce work and information overload", *CACM*, July 1994, pp. 30-41.

19. Jay Padgett, Christoph Gunther, Takeshi Hattori, "Overview of Wireless Personal Communications ", *IEEE Communications*, Jan. 1995, pp. 28-41.

20. Charles Perkins, Pravin Bhagwat, "A Mobile Networking System based on Internet Protocol", *IEEE Personal Communications*, vol. 1, no. 1, 1st Qtr. 1994, pp. 32-41.

21. Charles Perkins, Kevin Luo, "Using DHCP with computers that move", *ACM Wireless Networks*, vol. 1, 1995, pp. 341-353.

22. R. J. Vetter, C. Spell, and C. Ward, "Mosaic and the World-Wide Web," *IEEE Computer*, October 1994, pp. 49-57.

23. G. M. Voelker and B. N. Bershad, "Mobisaic: An Information System for a Mobile Wireless Computing Environment," *Technical Report, Department of Computer Science and Engineering*, University of Washington, September 1994.

24. T. Watson, "Wit: An Infrastructure for Wireless Palmtop Computing," *Technical Report* UW-CSE-94-11-08. University of Washington, November 1994.

25. M. Weiser, "The computer for the 21st century", *Scientific America*, 1992, pp. 94-104.

26. James E. White, "Telescript Technology: The Foundation for the Electronic Marketplace", General Magic, Inc.

27. Mohammed Zaid, "Personal Mobility in PCS", *IEEE Personal Communications*, vol. 1, no. 4, 4th Qtr. 1994, pp. 12-16.

28. TIA/EIA IS-41, "Cellular Radio Telecommunications Intersystem Operations", *Telecommunications Industry Association*, Dec. 1991.