proceedings of the First Workshop on Mobile Computing, April. 1995.

An Open Intelligent Messaging Network Infrastructure for Ubiquitous Information Service

Yao-Nan Lien

Computer and Communication Research Laboratories
Industrial Technology Research Institute
Hsinchu, Taiwan 31015, R.O.C.

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 as well as many commercially viable service networks providing various information services to the users. This paper proposes an open architecture that allows new services and facilities easily be added into networks, and an intelligent messaging service network on top of that architecture. Many issues regarding to the operation, administration, and maintenance are discussed with some solution provided.

1. Introduction

Because of the maturity of computer and communications technologies as well as the promotion of National Information Infrastructure (NII) initiated by the U.S. Vice President Al Gore, the progress of Personal Communication Systems (PCS) and *mobile computing* is accelerating into a revolutional speed making the ideal of ubiquitous information service, information anytime anywhere, a reality [6]. A layer of service system to help user to utilize the information and computing resources on this NII is the key to really bring the power of NII to the users.

To fit into the popular client-server model, the users can be viewed as clients and service providers can be viewed as servers. Theoretically, the problem can be greatly simplified if all servers in a network can be unified into a single server using distributed computing technology so that clients can access to the network resources transparently by interacting with a single server. Unfortunately, distributed computing technology in such a scale will have to take a much longer time to realize in the real world. Thus, clients have to access network resources in a prescriptive fashion by interacting with individual servers step by step to accomplish a single task. It is very difficult to accomplish a complicated task that requires an intensive interaction with multiple servers in mobile computing environments due to its intermittent communication nature and its battery energy limit. A new computing paradigm, *intelligent messaging*, that allows clients to interact with multiple servers in a dynamic fashion is created to cope with this problem [2],[3],[4],[5].

Simply speaking, an *intelligent message* is an electronic message that carries a computer program, whether procedural or declarative, which can be executed by the computer system of the recipient on behalf of the client who submits the message. The program can instruct the recipient computer to forward automatically the message itself to another server to execute the program in a pipeline fashion. It is also known as *intelligent agent* in other fields [2],[5]. In *Telescript* terminology, an intelligent message is called an *agent* [7].

2. Open Service Network Architecture

2.1. Requirements of Killer Services

There is no doubt that having some killer services is the most critical factor to establish a commercially viable service network. A killer ubiquitous information service requires at least following elements:

- some killer applications
- cost effective operation, administration and maintenance (OA&M)
- adequate financial reward

It used to take national resources to establish a service network such as Postal system and telecommun-

This work is supported by the MOEA Grant 35N7W0A.

ications networks because traditional infrastructure requires a very expensive OA&M system to guarantee the required quality of service. It will be almost hopeless to acquire similar resources to establish many expected information services. An architecture is needed to allow services of any scale and any quality to be added into the network easily. (Of cause, some kind of firewall must be provided to protect good quality services from the interference of poor quality services.) Service operators can choose whatever operation model they want depending on their available resources.

In this paper, we propose a simple high level open architecture that separates logical architecture from physical architecture and allows new services and facilities be easily added into the network shown in Figure 1. Further refinement can be made for detailed implementation based on this high level architecture [4].

2.2. An Open Architecture

Basic entities are servers each providing a specific service connected by various logical or physical communication networks such as Internet, PSDN, or ARDIS radio network. To avoid confusion, they are referred to as *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. *Terminal* is a physical device that allows a client to interact with a transport network. A terminal could be a telephone, a 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 physical transparency to the logical layer of the service networks as well as the required flexibility. The relationship between service networks and servers is very similar to many real world systems such as travel agent networks and airlines. An airline can be viewed as a server and a travel agent network can be viewed as a service network. A travel agent integrates many airlines together to offer its customers total solutions; and an airline can give ticket selling contracts to many travel agent networks. (A complete traveling package may consist of several flights from various airlines and probably with hotels and rental cars reserved. Not only the packages are more complete, it may even be cheaper as compared to DIY solutions.)

Theoretically, the physical location of a server can be flexible. However, it's mobility is insignificant compared to other issues. Therefore, we assume that every server has a fixed physical location.

Followings are the openness characteristics of this architecture:

- *With respect to a transport network, every terminal (server) has a unique ID.*
  Transport networks are assumed independent to each other.
- *With respect to a service network, every client (server) has a unique client (server) ID.*
  Similarly, service networks are assumed independent to each other.
- *A server may connect to multiple transport networks.*
  (A server must have an independent ID for each network it connects because transport networks are independent to each other. For example, one will not be able to send an E-mail to a phone number, although their destinations may be the same.)
- *A server may participate into multiple service networks.*
  (Similarly, the server must have an independent ID for each service network it connects.)
- *A service network may be supported by multiple transport networks.*
  That is, a client may access to the service network through different transport networks. For example, a PDA may use either Internet or phone to access the information provided by the same stock information service network. A client may submit a request from a cellular phone on the way to his office and then get the result back from a PC in his office later through Internet.
- *A (mobile) terminal may access to a transport network from more than one physical location using the same terminal ID.*
  (In fact, a terminal may be able to access to multiple transport networks. However, it is treated as different terminals. For example, a PDA may be equipped with both cellular phone and ARDIS modules, but this PDA must have a separated phone number and a separated ARDIS ID. If *Personal Communications Network (PCN)* technology can unify these two networks together in the future, they will be treated as a single network.)
- *A client may use different terminals in the same transport network to submit requests to a service*

network.
That is, clients may not necessarily identified by terminals. (For example, one can use any phone to make long distant phone calls and charge to his/her own account, a phone number or a credit card.)

Note that many of these characteristics are only optional. Network operators may construct their own networks based on their own needs. For instance, a service network may choose not to offer mobility or multiple network access capability to its clients.

2.3. Name Server and Mobility Management

In this architecture, an entity (server, clients, terminals) may have many IDs each corresponding to a network, either service or transport network. For simplicity, we refer transport network IDs as physical IDs, and service network IDs as logical IDs.

Within a service network, we must be able to reach each entity, either a client or a server, by its logical ID independent of the transport network it is being connected. Therefore, in a service network that supports the proposed polymorphism, a service is needed to convert a given logical ID into its physical IDs, including transport network ID and terminal ID. Once its physical IDs are known, the terminal location can be determined by the transport network itself. For instance, if a service network knows that a client is accessing the network via a cellular phone systems, the network can directly use his cellular phone number to reach that client without knowing his physical location.

2.4. Operation Modes

There are many ways to operate a service network. Among them, centralized and fully distributed operation modes are two extremes. Prodigy and American Online are typical centralized operations, while the *World-Wide-Web* (WWW) is a typical fully distributed operation that every WWW server is autonomous. In general, a commercial service network that guarantees a certain quality of service usually uses centralized operation, while a voluntary service network that doesn't guarantee any quality of service usually uses distributed operation for a lack of funding and liability.

3. Intelligent Messaging Service

On top of the open architecture shown above, a service network that is able to support intelligent messaging service can be easily created. We follow the Telescript terminology to call an intelligent message as an *agent*. In many cases, an agent can be viewed as a representative of a client. It travels in a service network acting as a client to request services from servers it visits. However, because a client can submit more than one agent to accomplish a single task, an agent may not be always equivalent to a client.

In the following examples, we assume that TravelNet, an intelligent messaging service network, that provides travel services, consists of the reservation servers of airlines, car rentals, and hotels, etc. These servers are all on the Internet and using Internet addressing scheme. Thus, Internet serves as the transport network for TravelNet.

3.1. An Example

The following example of intelligent message, written in a hypothetic language, tries to book a flight from New York to Chicago, reserve a rental car and a hotel room in their individual reservation servers [4]:

---

1. Global DATE = 4/20/95
2. GOTO reserve@airlineA.com
3. Flight = RESERVE first class from New York to Chicago on DATE after 12PM
4. GOTO reserve@chicago.avis.com
5. Rental_car = RESERVE 1 midsize for 2 days on DATE at Flight.arrive + 00:30
6. GOTO reserve@chicago.hilton.com
7. Lodging = RESERVE single room for 1 night on DATE at Flight.arrive + 02:00
8. GOTO $HOME
9. SAVE Flight, Lodging, Rental_car into Trip
10. DISPLAY Trip

Line 1 sets the global variable DATE to 4/20/95; Lines 2,4,6 move the agent to the reservation servers of airlineA, Avis Car Rental, and Chicago Hilton Hotel respectively; Line 8 moves the agent back to the originating terminal; Lines 3,5,7 make the reservation; Line 9 saves the result; and Line 10 displays the result.

The execution of above message has no magic. The client terminal executes the program until it hits Line 2, it then packs the the program with the program counter marked on Line 2 and sends the message to reserve@airlineA.com. When reserve@airlineA.com receives the message, it continues the execution from Line 3 until it hits Line 4. It then packs the program with the program counter marked on Line 4 and sends the message to reserve@chicogo.avis.com. The server reserve@chicogo.avis.com and all succeeding servers execute the message in the same manner.

As we can see from above example, an intelligent message is nothing more than a program whose execution place can change during the course of execution.

3.2. A More Complicated Example

A salesman wants a travel agent to make a trip plan from D.C. to San Francisco and to L.A. within a budget limit while he is visiting a customer in D.C. He wants the agent to book flights, hotels, and rental cars. The hotel he can stay depends on the expense on airfare; and the type of rental car he can rent depends on the total budget left. He might have to stay in Motel 6 rather than Hilton Hotel if he couldn't get a cheap airfare; he might have to take a limo rather than a rental car if he has no budget left. A traditional human travel agent might be able to offer this type of services without his intervention. The trip plan becomes more complicated when the plan needs to be more flexible adapting to his dynamically changed meeting schedule. He might have to choose to visit customers in San Francisco first then L.A. or the other way around depending on his meeting schedule. Some customers might not be available for some reason or simply a bad weather in his scheduled visits. Therefore, he has to make his travel plan dynamically adapting to other conditions. Assuming all these activities can be done electronically, a program can do a much better job than a human travel agent. Intelligent messaging paradigm will be suitable for this type of services.

3.3. Minimum Configuration

One may wonder it may have to take a long time and a capital investment to establish such a network. In fact, it only takes a PC connecting to a network, Internet or phone network, to form a service network. The popular mail-ftp is a typical free intelligent messaging service network. One can send an E-mail to a mail-ftp server to "ftp" a file. Many other simple IM service networks can be established in a similar way as long as they all agree upon a protocol.

3.4. Intelligent Messaging vs. Intelligent Network

Recently in telecommunication network area, *Intelligent Network* (IN), has been proposed to offer its customers "information based advanced services" [1]. Typical IM services are:

- route a phone call to a destination based on time of day and day of week;
- route a phone call to the nearest pizza store and automatically provide customer's profile to the store; and
- televoting.

One may wonder there might be a major overlap between Intelligent Messaging Service (IM) and Intelligent Network Service. In fact they are quite different:

1. Most IN services only offer connectivity services, while IM offers general information services and doesn't necessarily have the knowledge of telecommunication networks. IM may use IN as its transport network, though.
2. IN usually does not distinguish clients from terminals, while IM does.
3. An IN network tends to be offered by a single vendor in a highly centralized management environment, while IM tends to be offered by individual service providers.

4. OA&M Problems

To make an intelligent messaging service network commercially viable is not simply grouping all servers together. A good OA&M environment must be in place to provide a certain quality of service. To operate, administer, and maintain a service network is non-trivial that requires a major capital investment. Followings are some problems that anybody who is familiar with network OA&M would concern with.

1. How to locate a service?
2. How to know the status of an agent?
3. How to locate an agent in a service network?
4. How to control the execution of an agent that had already been submitted into a service network?
5. How to support transactions?
6. How to trace the execution of an agent (e.g. for debugging or auditing)?

7. What to do if an agent can create another agent (concurrent execution)?
8. How to handle the exceptions that might happen in the middle of a transaction?
   (a) an error is found in a task;
   (b) a server goes down in the middle of a task execution;
   (c) the server of a requested service is not available;
   (d) the originating client is not available when an interaction is needed. For example, an agent is suspended in a service node while the originating client goes down (or lose battery power, stolen, busy, etc). Without an adequate auditing and garbage collection mechanism, the network resources will be all tied up eventually.

5. Infrastructure for OA&M

5.1. Basic Definitions

5.1.1. Atomicity

Because an intelligent message may travel in a network to accomplish a single task in several prescriptive steps, some notion of atomicity is needed to maintain the integrity of service network in facing exceptions. Following terms are defined to support atomicity. Further details are yet to be addressed.

- *primitive service* - a service that is defined ahead of time and is known by all clients and servers.
- *prescriptive service* - a service that is not defined ahead of time and requires the service provider to fulfill the request in specified steps.
- *atomic service* - a service, whether primitive or prescriptive, that must be either fulfilled or denied completely as a whole. Each primitive service must be atomic by definition; and any segment of consecutive steps in a prescriptive service can be designated as an atomic service.

5.1.2. 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 message-ID is a unique sequence number generated by the client.

5.1.3. Agent Status

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

- *running* - an agent is being executed by a server. A running agent may actually be blocked waiting for local resources or is waiting for an external message such as client's input. In general, it is not observable by external entities. However, for reference purpose, we call such an agent to be in *spinning* state, although, to external entities, it is still in running 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 different between "spinning" and "suspended" is that a spinning agent can resume its execution by itself without an 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.)

5.1.4. Execution Record

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

5.2. Network Facilities

5.2.1. Network Management Center (NMC)

There might have some central facility providing network management functions, such as client and server registration, authentication, name server, coordinations, or client specific services. Although, it looks like a single node, it may actually be a number of nodes distributed over the network.

Note that a service network may not necessarily need a NMC. For example, the WWW which uses distributed operation mode doesn't have a NMC. In general, a commercial service network usually needs a NMC for such functionality as billing, authentication, security, etc.

5.3. Home Base Node (HBN)

Even though a client may change its location from time to time, it usually has a home location and a most frequently used terminal. (Because a mobile terminal device is vulnerable in a hostile environment, it is not wise to save all personal data on such a device without a backup information repository.) When a client subscribes to a service network, a *home base node*, can be assigned to the client, either by the NMC or clients themselves. The HBN for a client can be any node on the network such as the NMC, or a personal computer belonging to that client. In general, it would be better to assign to the most frequently accessed node to reduce the traffic. A lot of overhead in managing a service network can also be saved by using this facility. For example, when trying to locate a client in a service network, one can start the search from that client's own HBN. A client can also keep reporting his/her locations to the HBN so that it can be easily located. A HBN can also offer auxiliary computing resources to help mobile terminals to cope with their residential resource limitation. For example, a fax sent to a client can be converted into text data and then forwarded to client's mobile terminal. Followings are some example applications that HBN can perform:

- location register
- OCR server
- fax server
- answering machine
- news filter

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

5.4. 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 also choose whatever the events hs/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. These all depend on implementation details.

5.5. Access Nodes (AN)

Since a client may access to the network from different terminals and from different locations, there may be more than one access node during the life of an agent.

- *Original Access Node* - the service node through which the agent was originally submitted by the client.
- *Current Access Node* - the service node through which the client accesses to the network most recently.
- *Registered Current Access Node* - the current access node of an agent shown in its status holder. A registered current access node may be different from the real current access node due to the information update and message latency.

5.6. 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.

6. Future Work

Based on the proposed infrastructure, it will be much easier to solve many OA&M problems for intelligent messaging services. For example, a client can easily inquire current status of an agent by sending an inquiry agent to its status holder. Nevertheless, many issues, such as privacy and security, are yet to be resolved before a service network can reach certain level of service quality.

7. Concluding Remarks

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 as well as many commercially viable service networks providing various information services to the network users. It will take a major effort to create such an environment. Among many challenging issues, a service infrastructure like today's telecommunication networks or banking networks is definitely a very important one. The open architecture and OA&M

infrastructure proposed in the paper will allow new services and facilities be easily added into the network.

Reference

[1] J. Garrahan, P. Russo, K. Kitami, and R. Kung, ''Intelligent Network Overview'', *IEEE Communications*, pp. 30-36, March 1993.

[2] Michael Genesereth and Steven Ketchpel, ''Software Agents'', *CACM*, pp. 48-53, July 1994.

[3] James Lee, ''Intelligent Messaging Paradigm and Telescript,'' *CCL Technical Journal*, No. 35, Dec 1, 1994, pp. 37-41.

[4] Yao-Nan Lien, ''Perspective of Service Networks on National Information Infrastructure,'' *CCL Technical Journal*, No. 35, Dec 1, 1994, pp. 28-36.

[5] P. Maes, ''Agents that reduce work and information overload'', *CACM*, pp. 30-41, July 1994.

[6] M. Weiser, ''The computer for the 21st century'', *Scientific America*, pp. 94-104, 1992.

[7] James E. White, ''Telescript Technology: The Foundation for the Electronic Marketplace'', General Magic, Inc.

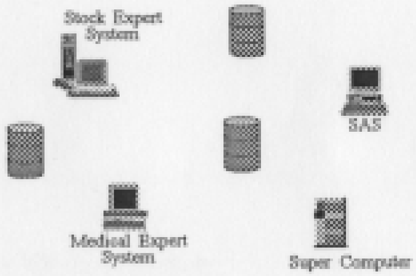

Figure 1(a) Servers

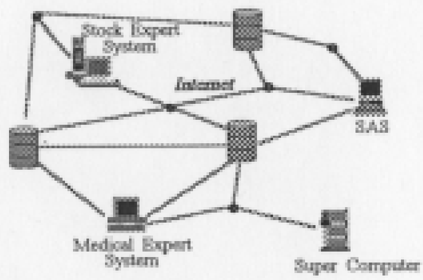

Figure 1(b) Servers interconnect to each other through Internet

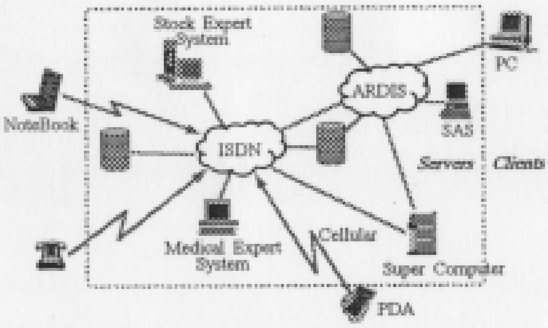

Figure 1(c) Servers interconnect to each other through various networks

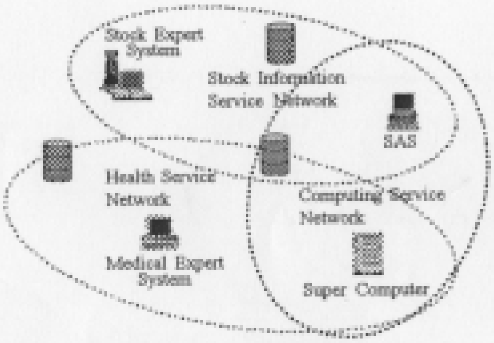

Figure 1(d) Servers can participate in any service network

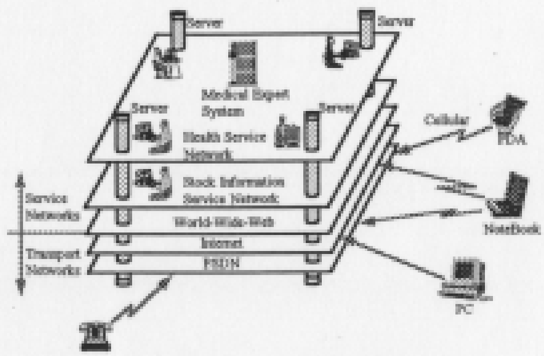

Figure 1(e) Overall Architecture