Seamless Mobile Service for Pervasive Multimedia

Enyi Chen, Degan Zhang, Yuanchun Shi, and Guangyou Xu

Key Lab of Pervasive Computing
Dept. of Computer Science and Technology
Tsinghua University, Beijing, 100084, P.R.China
{chenenyi00@mails,gandegande@mail}.tsinghua.edu.cn
http://media.cs.tsinghua.edu.cn/~pervasive

Abstract. In this paper, we propose a manager of seamless mobile service for pervasive/ubiquitous multimedia, which can dynamically follow the user from place to place without user awareness or intervention by layering architecture of component platform and agent-based migrating mechanism under consideration of robustness, scalability and load balancing. The manager can custom multimedia task and encode service descriptions by the XML or SMIL technology to provide flexibility, it can also discover the services, filter, synthesize or handoff them according to the context information and match between the task and service, especially, it can manage seamless mobility of pervasive multimedia. The validity evaluation of the manager has been done by experimental demo.

1 Introduction

How does a mobile device utilize available resource in the surrounding for service advertisement, discovery, filtration, synthesis and migration? With the shift of the history and context of pervasive multimedia during the mobility of user or task of user, how does the computing device and software resource around it make adaptable change for seamless mobility [1] [2]. How to deal with the problem of seamless mobility and realize the transparent transferring of task is one of key technologies in pervasive computing. Currently, the useable technology is like mobile IP used as network-level protocol, fixed or mobile agent used in application-level cases. Migration based on mobile agent is one kind of important method in pervasive computing paradigm. The focus is how to automatically manage and coordinate useable distributed computing environment, record its history, and restore its context seamlessly so as to continue the task and meet the requirement of mobile application. The chief function requirement of seamless mobility is focused on the continuity and adaptability of pervasive multimedia [3]. The continuity is that the application can pause and continue to work later without the loss of the current state and the running history. The adaptability is that the application is not restricted by computing device and context of service but adaptable to its environment. In our opinion, this is a kind of mobile working paradigm [4]. But when seamless migration for computing task of pervasive

K. Aizawa, Y. Nakamura, and S. Satoh (Eds.): PCM 2004, LNCS 3332, pp. 754–761, 2004.

multimedia is realized on PC, laptop, or PDA, there are many difficult problems to be solved [5] [6]. The rest of this paper will be organized as follows. Firstly, we give the Scenarios and architecture of manager for seamless mobile service, then we describe mapping between task and useable service, design a kind of strategy of seamless mobile service, including solving the following several sub-problems: avoiding migration failure and remainder dependency. Finally, we evaluate the validity of our manager for seamless mobile service and draw a conclusion.

2 Scenarios

Scenario I: When you fly to Beijing Capital International Airport from Kyoto Airport, you may transfer in Tokyo International Airport. It is very boring since you have to wait for another two hours before your next flight in the lounge. So you use your wireless enabled PDA to search for "games". You find another traveller who launches a "Chinese Chess" game, which is just your favorite. So you connect to his device and play with him.

Scenario II: When you work in front of your personal computer (PC), your mobile phone rings. After you pick it up, what make you surprised is that the caller's live video is displayed on your computer screen though the user's mobile phone can display and capture the live video. Vice versa, your live video is also displayed on the caller's computer screen.

3 Architecture of Manager for Seamless Mobile Service

The architecture of our seamless migration manager is also Multi Agents System (MAS) including fixed or mobile agents. It can run on networks connected PC, laptop, or PDA. Now we introduce the manager. The runtime environment is composed of four kinds of components: Human-computer interaction interface, Task manager, Continuity manager and Service manager (as the figure 1). The introduction of them is as followings:

1) *Human-computer interaction interface* including agent interface and relative controlling. The agent interface is used as defining the attributes of agent, such as ID of agent, Name of agent, Type of agent (such as TA, SA, UA, VA, EA, DA, and so on), Current status of agent (one of five status are "Ready",

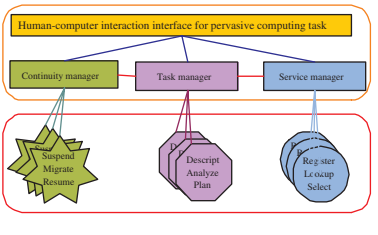

Fig. 1. Components of our seamless migration manager

"Waiting","Transferring", "executing", "Dead" or "Destroyed"), Association relationship of agent (including relationship between agent and task of learning, relationship between agent and agent).

2) *Task Manager* is for application service, which manages the application/task array, including task description of learning, task analyzing, mapping or binding between task and service, loading, executing, planning schedule of task of learning, etc.

3) *Continuity Manager* is for preparing "Migrating travel plan/schedule" of task of learning, sensing context, suspending of task, historical status (including log, configuration, etc.) recording, agent management, addressing of target node, determining of transferring granularity which is for avoiding the transferring failure, reducing the remainder dependency & contracting the transferring delay, resume of task of learning, and so on.

4) *Service Manager*. It manages the registration of service, service discovery [8], lookup of service, service selection/association, and mapping or binding between task and service.

These components can communicate each other, and may be controlled by human-computer application interaction interface including agents and relative control, which is individual interface for PC, laptop, PDA.

4 Adapted Mapping Between Task and Useable Service

Figure 2 shows adapted mapping between task described with XML or SMIL and useable service, where the transaction T wanted to be completed by user is named Task, which constitutes of subtask or sub-transaction Ti, each Ti is independent unit of function. Because of the diversity of task, its subtask is different each other, in order to keep their compatibility, the description of subtask should be abstract, mainly, the key and necessary parameters. The execution of task is finished by the service supplied by relative resources R, but different subtask, the done method is different too, we can be classified it into three kinds: Event Type, Stream Type, Bulk Type.

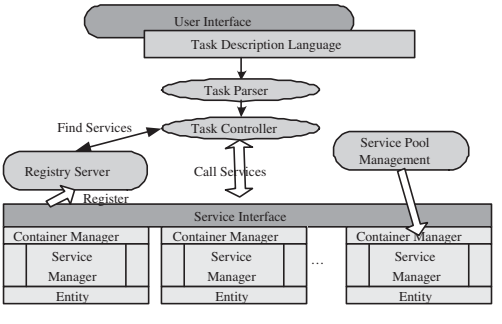

Fig. 2. Adapted mapping between task and useable service

Tasks can be described normalizedly by XML or SIML, the format by XML is as follows:

```
<?xml version="1.0"?>   <Task_Descriptor> <head><name>TaskX
</name> <comment></comment></head><body><Task_Owner></Task_Owner>
<Service_Attribute></Service_Attribute> </body></Task_Descriptor>
```

5 Strategy of Seamless Mobility Based on Agent

As a kind of special computing resource, agent supports deployment of computing resource and mobility freely, which makes the system manage and adjust easily, so it is suitable for application of seamless mobility.

If the transferred amount of data is partial, and this part of information must be transferred firstly so that the task can restore the runtime environment and run continuously on the target node, this part of information is regarded as "Key Set", so we can divide the migrated information into several chunks, such as executing code chunk, running status chunk, and so on. For resuming, the "Key Set" chunks must be integrated, otherwise it is impossible to go on running continuously.

According to classification of agent in our test bed, we make the following rules:

Navigation Agent (VA) need NOT do direct relative works with the task, which is familiar with the topological structure of Internet/subnet of target node and addressing in the network. The Data Structure of VA may be divided into two parts: one is itself "function body", another is MessageBox (MB, mark as) used as loading moved object and transferring in the Internet/subnet node. Task Agent (WA) does detail jobs, which includes executing the code, managing the data and environmental status, and so on. It can transfer with the Navigation Agent (VA) in the network and need NOT know the structure of Internet/subnet. When migrated, WA seeks relative VA and joins in its MB firstly, then sent to target node by VA.

The designed and adopted algorithm of seamless migration by us is as follows:

1) According to the prepared TRAVEL SCHEDULE/PLAN for migrating / transferring, logic node PA2 lets VA begin addressing in the network according to the address supplied by logic node PA3, when the connection is successful, VA sends instruction "TransferNode" to Logic node PA3 as target node, VA+WA transfer to PA3 after packing. Logic node PA3 sends instruction "UpdateLinking" to all logic nodes connected to PA2, such as logic node PA1.

2) When PA1 has received the instruction "UpdateLinking", it creates the association to new link, and sends instruction "LinksUpdated" to logic node PA2. When PA2 has received all expected message "LinkUpdated", and then sends instruction "ActiveNode" to logic node PA3.According to the topological relationship, under the rule of FIFO, PA3 activates the Messenger, up to now, the transferring work is finished. When all Messengers are activated, each WA will restore running environment and do instruction "ExecuteTask". During the

executing of each WA, on the one hand, the historical snapshots will be recorded and saved, on the other hand, VA do the instruction "ListenTask" continuously and get the next transferring instruction "TransferSignal". During the executing of WA, VA checks the prepared TRAVEL SCHEDULE/PLAN, if another new migrating plan is checked, the "TransferSignal" instruction will be sent to WA. If no instruction "TransferSignal" is received by WA, it will execute its task continuously until the task is completed, otherwise, it will stop executing, and Goto 1 for the new migrating/ transferring.

6 Focus of Seamless Mobile Service of Multimedia

6.1 The Migration Failure Problem

In the pervasive computing environment, because the position of agent is often variable, the cases may be occur that when the agent1 is being transferred to agent2 and embedded in it to do the task together from node C, but during the transferring, the agent2 has moved from node C to node D, when agent1 arrives at node C, it can NOT find the agent2. This case is called the migration failure problem. This kind of problem can NOT keep the continuity of transferring of task.

In order to solve this problem, we think there are three factors should be considered: 1) When the position of agent has moved, how to know this change by other relative agents. 2) When the transferring of agent, how to deal with the message sent to it. 3) During the transferring of agent, whether the receiving agent can be transferred freely or not.

Our solution is as follows: 1) When the agent moves to new node, it should send "Notation" Message to all other relative agents 2) Before the agent prepares to be transferred, it should query the current position of receiving agent, at the same time, the transferring relationship and event should be sent to it by message 3) Determine the transferring topological relationship of agent, based on the rule "FIFO", when each agent begins to be transferred, the receiving agent may be transferred but a little later should be locked, after the transferring process is over, and the receiving agent should be unlocked. For it, a signal semaphore may be set.

In our test bed, the "Notation" message may adopt three kinds: Unicast, Multicast, Broadcast. The synchronic mechanism "addressing first, then locking and transmitting" designed based on the "time-topological" relationship by us can realize the synchronization between transferring agent and receiving agent, the transferring failure problem can be solved radically and adapted for all kinds of application pattern, besides avoiding the failure, the restriction to receiving agent may be reduced, so it is general. Where, "time-topological" relationship may be considered in the "Travel Plan/Schedule for transferring". The schedule may consist of certain travel sequence, the data structure of each travel sequence is like Table 1.

In Table 1, TP_MC, TP_SP, TP_RE are important for basic migrating / transferring operation, that is to say, Only the TP_MC is OK, "Travel Plan for

Table 1. Data structure of each sequence

No	ID	Description	No	ID	Description
1	TP_ID	ID of plan	7	TO_IPC	Source IP of migrated object
2	TP_NP	Name of plan	8	TO_IPD	Target IP for migrated object
3	TP_MD	Made date of plan	9	TP_MC	Migration condition
4	TP_ON	Order number of plan	10	TP_SP	Snapshot point of suspension
5	TP_NO	Name of migrated object	11	TP_RE	Resume point of task
6	TP_MG	Migrated granularity	12	TP_RI	Running ID after resuming

transferring" may be run, meanwhile, record and save "TP_SP" and "TP_RE", both for restoring the running environment.

Whether TP_MC is OK or not, the following aspects should be checked: Whether the current status (may be one of five kinds: Ready 1, Waiting 2, Transferring 3, Executing 4 , Destroyed 5) of agent is Waiting 2, whether the target address TO_IPD may be reached or not , whether the threshold of transferring delay is OK or not, whether the reminder dependency cases may exist or not.

6.2 The Remainder Dependency Problem During Migrating

In fact, there are three valid mapping forms based on the three factors above:

1) Strong Transfer, suspends after transferring, resume/restore and run after finishing the whole information

2) Strong Transfer, suspends after transferring, resume / restore and run timely after finishing the Key Set information (at the same time, the whole remainder information will continue its transmitting)

3) Weak Transfer, suspends after transferring, resume/restore and run after finishing the Key Set information (at the same time, the selected partially information will continue transmitting to the target node)

In the first mode, the transferring delay of agent includes that packing the whole information and transmitting, restoring the agent and the whole information on the target node; In the second mode, includes that packing the Key Set information and transmitting, restoring the agent and the Key Set; In the third mode, includes that packing the Key Set information and transmitting, restoring the agent and the Key Set. In the same condition, the delay of the first one is the longest, the third one in the shortest. In the second and third migrating/transferring mode, because of selecting a part information as the "Key Set" and being transferred firstly, but for different application, it is NOT known that which part of information is necessary, a certain "Key Set" may NOT migrate to the target node timely, the running agent must wait for it, that is to say, running agent is still dependent on part of information on the source node, this case is called "remainder dependency". This problem may lengthen the transferring delay of task, when it is serious, it will influence the seamless mobility, so this

case must be avoided. In our opinion, the "remainder dependency" problem will be solved from two aspects:

1) Tuning reasonably the transferring granularity of task. We divide it into several parts, and they deal with by their relative agent, such as Execution-code Agent (EA), Data Agent (DA) and other Agents (such as Environment-state Agent). If too larger, it is restricted by the bandwidth; If too smaller, transferring time is much more. Both may lengthen the delay. Based on analyzing on theory and application tests, we adopt a kind of partition method called "subsection" or "pagination", the size or number of "section" or "page" is determined adaptively by bandwidth, cache buffer, volume of MessageBox (MB). When this case is occurred, the necessary information may be transmitted through "section interruption" or "page interruption", but the frequency should be adjusted adaptively according to historical record information.

2) Optimizing the Key Set. The Key Set will be determined adaptively according to nearest principle and used frequently principle and cut off the redundancy information. The relative adapted strategy may be referred the bibliography [9].

7 Evaluation of the Manager

Currently, we supply the function that the task dynamically follows the user from place to place and machine to machine. For example, the video-playing task may follow me from my house to other places, such as my office, stadium, coffee house, park, airport, etc., and vice versa. The task is described partially by SMIL in the following. Because the migrating/transferring mode based on agent is distributed or peer-to-peer / end-to-end, we have designed several relative communication primaries for migrating, for example,

1) BeginToListen Primary.

BeginToListen(UINT nPort, ACCEPT_CALLBACK callback);

Void(CAgent::*ACCEPT_CALLBACK)(UINT &Connection_ID);

2) BeginToRequest Primary.

BeginToRequest(UINT &nConnection_ID, CString IP,UINT nPort);

3) Tranfer Primary.

Transfer(UINT nConnection_ID, CString strMsg, CDate time_stamp);

Based on the above test bed, one snapshot comparison result of experiments for seamless migration from PC to PC, laptop and PDA is shown. From the results, the delay time from PC to PC is the shortest under the same evaluation framework, but the delay time from PC to PDA is the longest.

Of course, Security is a big problem for test bed. We will provide security by applying existing forms of public-key cryptography and Intrusion Detection Agent (IDA). During seamless migration, the intrusion cases include illegal access, hostile attack through all kinds of invalid channel or approach, etc, so the detection mechanism to be proposed by us includes monitoring, auditing, analyzing, warning, response, and so on [6].

8 Conclusions

Based on the application requirements of pervasive computing, we have proposed a seamless migration manager for pervasive multimedia, which supplies the function that the task dynamically follows the user from place to place. Our key insight is that this capability can be achieved by layering architecture of component platform and agent-based migrating mechanism. In this paper, we have given the architecture of manager for seamless mobile service, described mapping between task and useable service, designed a kind of mechanism of seamless migration, including solving these problems, such as method of seamless migrating, avoiding migration failure and remainder dependency. The validity of the manager has been evaluated by the demo.

References

1. M. Satyanarayanan, Pervasive Computing: Vision and Challenges, IEEE Personal Communications, vol.8, no.4, pp. 10-17, Augest, 2001
2. M. Kozuch and M. Satyanarayanan, Internet Suspend/Resume, In Proceedings of Fourth IEEE Workshop on Mobile Computing Systems and Applications, Calicoon, N.Y., Jun. 1, 2002
3. K. Takasugi, Seamless Service Platform for Following a User's Movement in a Dynamic Network Environment, In Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03), Fort Worth, Texas, USA, March 23-26, 2003
4. Y.C. Shi, W.K. Xie, G.Y. Xu, R.T. Shi, E.Y. Chen, Y.H. Mao and F. Liu, The smart classroom: merging technologies for seamless tele-education, IEEE Pervasive Computing, vol.2, no.2, pp. 47-55, April-June, 2003
5. E.Y. Chen, Y.C. Shi, D.G. Zhang and G.Y. Xu, A Programming Framework for Service Association in Ubiquitous Computing Environments, In Proceedings of 4th IEEE Pacific-Rim Conference on Multimedia(PCM2003), vol.1, IEEE Press, pp. 202-207, Singapore, December 15-18, 2003
6. D.G. Zhang, G.Y. Xu, Y.C. Shi and E.Y. Chen. Moblie agents with intrusion detection during sealess transfer, In the doctoral colloquium of Pervasive 2004, Vienna, Austria, April 18-23, 2004.
7. P. Ciancarini, Coordinating Multi-Agent Applications on the WWW: A Reference Architecture, IEEE Trans. on Software Engineering, vol.24, no.5, pp. 363-375, 2002
8. D. Garlan and D.P. Siewiorek, A. Smailagic and P. Steenkiste, Project aura: toward distraction-free pervasive computing, IEEE Pervasive Computing, vol.1, no.2, pp. 22-31, Apr-Jun, 2002
9. A.R. Tripathi, Ajanta - A System for Mobile Agent Programming, Technical Report(TR02-016), Department of Computer Science, University of Minnesota, 2002