Workshop on Computer Networks

The Design and Implementation of a Mobile Agent-Based Framework for Context Aware Computing

Tzu-Han Kao, Yi-Hsiang Chou, Ming-Chun Cheng, Hsin-Ta Chiao,

Shyan-Ming Yuan

Department of Computer and Information Science
National Chiao Tung University,
1001 Ta Hsueh Rd.,
Hsinchu 300, Taiwan.

Email: {gis89539, gis89515, native,
gis84532,smyuan}@cis.nctu.edu.tw

Abstract. With the progress of wireless and mobile network technology, people can access information via various kinds of handsets or facilities of Information Appliance (IA), while these facilities range from high-computed ones to low ones. Within the handsets, however, cellular phones, PDA, Tablet PC enable people to surf the internet and move at same time in the wireless network environment. Under such circumstance, people can access different type of internet information, like words, sounds, images or videos, by specific terminal device at anytime and anywhere. In such case, however, here come two problems: (1) The movement of users is limited; (2) The users must carry their terminal device with themselves. In order to solve these problems, we combine both the concepts of pervasive computing and context awareness to develop a framework possessing mobile agent paradigm and context data modeling.

Aimed at the present pervasive computing in the wireless network environment, we propose a Mobile Agent Context Aware Framework– MACAF. In MACAF, we regard mobile agent system as the fundamental computing platform, and describe three kinds of context data by RDF - Component Profile, User Device Profile, and User Preference Profile. In addition, we use dynamic class loader of JAVA and XML parsing techniques so that when personal mobility and terminal mobility arise, users' applications can move with users via two kinds of mobility technique of agent of this system - Agent Migration Approach (AMA), and State Transfer Approach (STA). This system makes applications follow users and applications be reconfigured automatically after mobility so as to accommodate various execution environments to reach the purpose of the context awareness and the pervasive computing.

Keywords: context awareness, pervasive computing, mobile agent

The Design and Implementation of a Mobile Agent-Based Framework for Context-Aware Computing

Tzu-Han Kao, Yi-Hsiang Chou, Ming-Chun Cheng, Hsin-Ta Chiao,

Shyan-Ming Yuan

Department of Computer and Information Science, National Chiao Tung University,
1001 Ta Hsueh Rd., Hsinchu 300, Taiwan.
{gis89539, gis89515, native, gis84532, smyuan}@cis.nctu.edu.tw

Abstract. With the progress of wireless and mobile network technology, people can access information via various kinds of handsets or facilities of Information Appliance (IA), while these facilities range from high-computed ones to low ones. Within the handsets, however, cellular phones, PDA, Tablet PC enable people to surf the internet and move at same time in the wireless network environment. Under such circumstance, people can access different type of internet information, like words, sounds, images or videos, by specific terminal device at anytime and anywhere. In such case, however, here come two problems: (1) The movement of users is limited; (2) The users must carry their terminal device with themselves. In order to solve these problems, we combine both the concepts of pervasive computing and context awareness to develop a framework possessing mobile agent paradigm and context data modeling.

Aimed at the present pervasive computing in the wireless network environment, we propose a Mobile Agent Context Aware Framework– MACAF. In MACAF, we regard mobile agent system as the fundamental computing platform, and describe three kinds of context data by RDF - Component Profile, User Device Profile, and User Preference Profile. In addition, we use dynamic class loader of JAVA and XML parsing techniques so that when personal mobility and terminal mobility arise, users' applications can move with users via two kinds of mobility technique of agent of this system - Agent Migration Approach (AMA), and State Transfer Approach (STA). This system makes applications follow users and applications be reconfigured automatically after mobility so as to accommodate various execution environments to reach the purpose of the context awareness and the pervasive computing.

Keywords: context awareness, pervasive computing, mobile agent

1. Introduction

As the proliferation of the Internet, the Internet has played an important role in people's life. Logging on line has been already what many people do everyday. Nowadays most people still log on line by using workstations or personal computers, these machines are usually fixed in one location. Via these machines, which are capable of connecting to the Internet, users can choose services they need on the Internet and make a request to the services.

Recently, the development of wireless network goes to rapidity. Public places, such as schools, cyber cafes, airports and hotels, are starting to construct wireless network

environment. Being the case, you can also enjoy wireless network in the living room, bedroom or any other places at your home.

In addition to desktop PCs and laptops, the wireless devices are getting multiform. Because handsets and PDAs are easy to carry, they will play a significant part in wireless network in the future. People can access abundant information on the Internet whenever or wherever they want via heterogeneous wireless devices.

With the development of wireless and mobile network technology, these various devices, such as PDA, Palm, and mobile phones provide an environment for people to intercommunicate or access the content of the Internet in anytime or any place. These operating systems, application execution environments and protocols of these devices are different strongly, making software designer and user mobility become complicated increment.

For instance, you may use a desktop PC with Ethernet in a company or a school, and you may use a PDA or a cellular phone with various standards of wireless network, such as GPRS, GSM or Bluetooth. Meanwhile, here comes a problem. The cause of this problem is that the capabilities of wireless devices are different. The differences range from the screen size, number of color, screen resolution and etc. Of these devices, there will cause incompatible data between these devices, and this difficulty is inevitably for content providers to face, for they must provide information for customers with all kinds of devices.

Based on the phenomenon mentioned above, we enumerate the problems in wireless network nowadays:

- Wireless environments and devices are heterogeneous
- Lack a universal mechanism to provide ubiquitous computing
- The capabilities of handheld devices are limited

Some researches investigate a concept of context awareness[3][4][5][6] to solve this problem, including context awareness application and context aware mobile portal. These applications or systems will sense the changes of users' preferences or users' device capabilities[7][8] to provide appropriate users' environments.

In this paper directed to the above, we integrate the concept of pervasive computing and the concept of context awareness[3][4][5][6][13], developing a JAVA-based framework– MACAF with the mobile agent paradigm[1][2][15] and context data modeling.

Aimed at the present pervasive computing in the wireless network environment, we propose a context awareness-possessing framework - MACAF. We regard the mobile agent system as the fundamental computing platform, and describe three kinds of context data by RDF - Component Profile, User Device Profile, and User Preference Profile. In addition, we apply a dynamic class loader of JAVA[14] and XML parsing techniques[19] so that when personal mobility and terminal mobility arise, users' applications can move with uses via two kinds of mobility technique of agent of this system- Agent Migration Approach (AMA), and State Transfer Approach (STA). Application programmers can write programs with two characteristics of environmental adaptation and auto-reconfiguration by inheriting or extending the software objects we developed to achieve the objective of context awareness and pervasive computing.

The organization of this paper are as follow: in section 2 we brief a few basic techniques, inclusive of mobile agent, context-awareness and Composite Capability/Preference Profile (CC/PP)[8] developed by W3C; in section 3, we describe in detail the system design of MACAF; in section 4, the context data modeling was introduced; in section 5, we discuss the system interacting procedure of Agent Migration Approach, State Transfer Approach, and profile change, when agent move in the system; section 6, we summarize this paper by conclusions and the future works.

2. Background

2.1 Mobile agent

In system perspective, an agent is a software object with internal state that possesses the following mandatory properties:

- Reactive: senses changes in the environment and acts according to those changes
- Autonomous: has control over its own actions
- Goal-driven: is proactive
- Temporally continuous: is continuously executing

And an agent may possess any of the following orthogonal properties:

- Communicative: able to communicate with other agents
- Mobile: can travel from one host to another
- Learning: adapts in accordance with previous experience
- Believable: appears believable to the end user

Agents can be found in heterogeneous environment such as computer operating systems, networks, databases, and so on. If we look at all these systems, we will find a property shared by all agents: they have the ability to interact with their execution environments and to act asynchronously and autonomously upon it.

Mobility is an orthogonal property of agents and we never want agents to move stationary agents. A stationary agent executes only on the system where it begins execution. In contrast, a mobile agent is not bound to a host where it starts execution. In conclusion, it has the unique ability to transport itself from the host to another through networking. Figure 1 illustrates mobile agent paradigm.

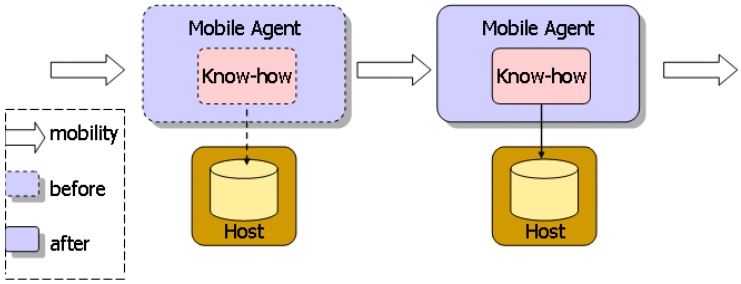

Fig. 1. The mobile agent paradigm

2.2 Context-aware computing

One challenge of the mobile distributed computing is to exploit changing environments with a new class of applications that are aware of the context in which they are running. Such context-awareness systems can examine computing environments and make application can re-execute when the execution environments changed.

Three important aspects of context are: where you are, whom you are with, and what resources are nearby. Context data is not merely the information of users' locations, but the other information related to users were involved. Exactly, context data includes information, such as user's location, device capabilities, network connectivity, communication bandwidth and even user's interpersonal relationship; e.g., whether you are with your manager or with a co-worker or not.

Device mobility is that a client using his mobile device and move from one local area network to another, and his mobile device, likes a mobile phone, is carried continually.

User mobility is that a client executes applications or services with a certain device in one local area network and he can resume his tasks with a different facility in another local area network after movement.

2.3 Composite Capability/Preference Profile (CC/PP)

CC/PP[8] is derived from earlier work done within the W3C Mobile Access Interest Group and the WAP Forum's User Agent Profile working group[7]. CC/PP is a general, and extensible framework for describing users' preferences and device capabilities. This information can be collected into a profile and provided by context information servers or content providers in the Internet.

The CC/PP framework is based on XML/RDF[9][12]. XML/RDF provides the framework with the basic tools for both vocabulary extensibility, via XML namespaces, and interoperability. The fundamental of the data modeling for a CC/PP is a tree. The initial branches are the contextual components or user agents described in the profile. Each major component may have a collection of properties or preferences. The description of each component is a sub-tree whose branches are the capabilities or preferences associated with that component. RDF makes modeling a wide range of data structures possible, including arbitrary graphs, however it is unlikely that this flexibility will be used in the creation of complex data models for profiles. A capability can often be described with a single RDF statement with a simple, atomic value. In situations where this is not sufficient, RDF makes capabilities with complex, structured values possible. This can be useful when multiple values are needed. For example, a browser may support multiple versions of HTML. Figure 2 illustrates this tree structure with a few components and properties. Each component has a type from which its class definition is inherited.

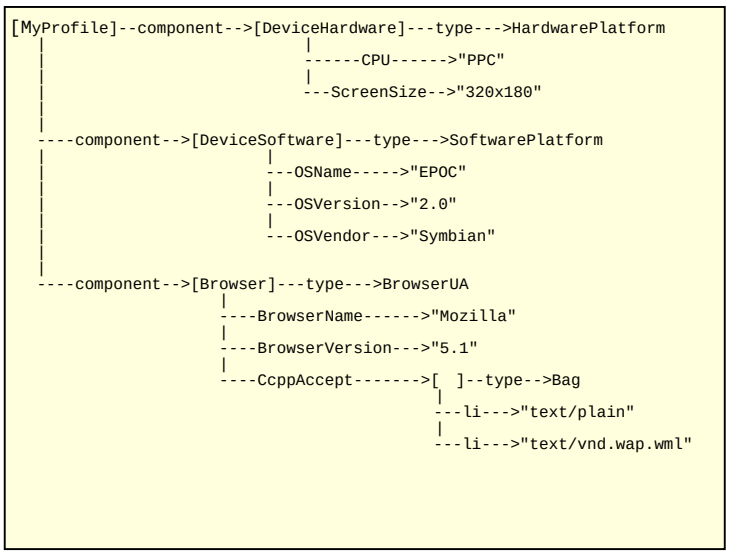

Fig. 2. The components and properties of a CC/PP profile

3. System Architecture

In the environment of our system architecture, we have to make an assumption beforehand. There is at least one gateway responsible for providing services to users in each local area network. The gateway plays the most important role in our system architecture. We design the critical system components in gateways to achieve providing various services to the users with heterogeneous devices (we call them clients). The deployment of our system architecture is illustrated in Figure 3.

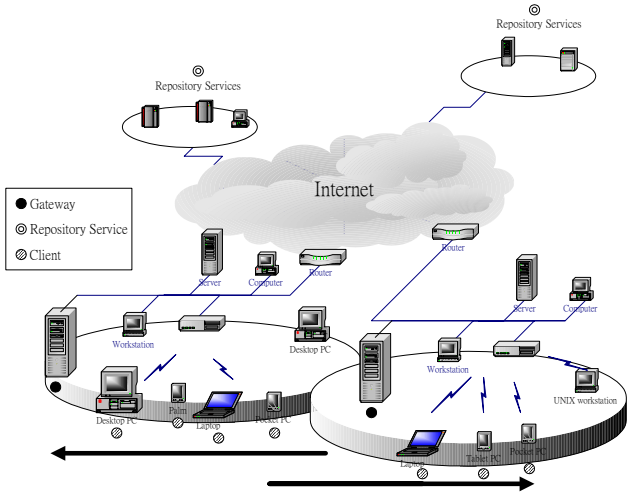

Fig. 3. The system infrastructure of MACAF

Two local area networks in the bottom of Figure 3 are connected with each other via the Internet. In each local area network, there are three important roles: gateway, client and repository service. We highlight each of them with different styles of circles. In a local area network, the gateway is responsible for providing services to the local clients and the repository service provide resources which gateways need.

Instead of the gateway in the original local area network, the new one should provide services continually for users when clients move from the original local area network to the new one.

We figure out explicitly the system architecture proposed by us. From the front-end to the back-end, our system architecture includes three tiers – Client tier, Gateway tier, and Repository services tier respectively.

The Figure 4 illustrates:

- The left tier, the **Client tier** shows the client device components in our system, such as Client API, User Agent, and Communication Interface.
- The middle tier, the **Gateway tier** shows the core components in the back-end side of

our system, such as Allocated Agent, Agent Manager, Component Manager, Agent Pool, etc.

- The right tier, the **Repository Services tier** shows the core repositories store contextual components and documents in back-end side of our system, such as Application Repository Service, Component Repository Service, etc.

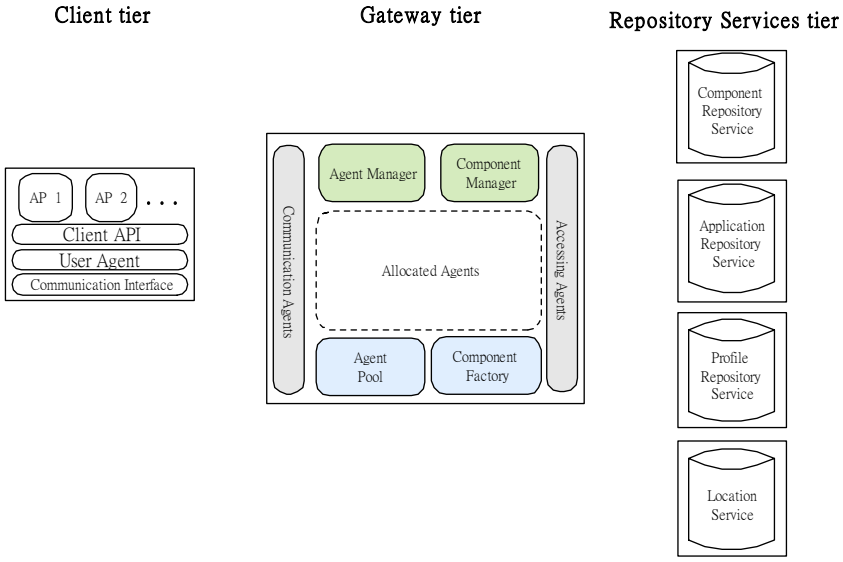

Fig. 4 The system architecture of MACAF

3.1 Gateway Tier

The main function of gateway is to provide personalized and adaptive services to users' applications. In other words, the system provides for the clients the most appropriate content based on the users' environments.

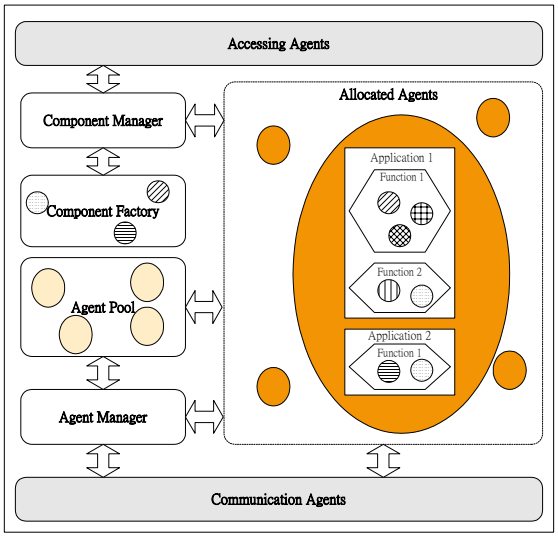

Fig. 5. The components of a gateway

First, we figure out the architecture of a gateway. It possesses these functionalities as follows:

1. Receiving clients' requests and serving them
2. Inter-operating among gateways to achieve the cooperation and collaborative services
3. Accessing and proxying repository services in back-end tier

In this paper, the term "agent" means agents in the agent pool and allocated agents, if not specified. An allocated agent we called is an agent delegated to a client for doing the client's job.

The modules in the gateway include the following parts, which are enumerated as follows:

1. Agent Manager

   An agent manager is the critical component in the gateway responsible for managing the agents in agent pools or allocated agents. The inclusions are:

   (1) Assigning an agent to serve a client

   (2) Collecting the abandon allocated agents back to the agent pool

   (3) Maintaining the number of agents in agent pool

   (4) Communicating with agent managers in other gateways to move agents between gateways

   (5) Receiving the registration request from clients

   (6) Reconfiguring agents for clients to provide personalized service

2. Agent Pool

   An agent pool is a data structure in the system memory instead of in the disk. Furthermore, agents as residents in it prepare to execute for performance consideration. In this pool, once a client's registration request is received, an agent initialized before will be allocated by an agent manager.

3. Allocated Agents

   An allocated agent is an agent was assigned to a client by the agent manager to provide services. Figure 5 illustrated an allocated agent be magnified for explaining intrinsic to it, and we can point out the following:

   (1) Executing client's applications

   (2) Maintaining its clients' profiles

   (3) Each function contains no component or more

   (4) Following a client who own it while the client move

4. Component Manager

   A component manager is the critical component responsible for managing contextual

components in the component factory and clients' profiles. The inclusions are:

(1) Supplying agents with contextual components if necessary

(2) Maintaining the number of components in the component pool

(3) Accessing the component repository service and the profile repository service via the accessing agent

(4) Processing the context data composed of the component profile and the user profile

5. Component Factory

   A component factory, like the Agent Pool, is a cache to place contextual components. Plus, prepared to be composed for performance consideration, contextual components have been created by the component manager. In the factory, once a client's request is received, the requested contextual component is returned.

6. Communication Agent

   A communication agent is responsible for wireless networking between clients and gateways. Several wireless access techniques, such as IEEE802.11, Bluetooth and the like, are provided. Trough them, agent managers can communicate and allocated agents can be migrated.

7. Accessing Agent

   An accessing Agent is responsible for the communication between the gateway and repository services by means of wired networks.

3.2 Repository Services Tier

A repository service can be combined with a gateway. In other words, a computer can be either a gateway or a repository service or both. The main function of repository services is to provide data to gateways or clients. These data include:

1. Component Repository Service

   The component repository service contains all necessary contextual components that gateway may use. More precisely, a component manager of a gateway request component repository service to transmit components, and provide them for applications of agents.

2. Application Repository Service

   The application repository service contains all class files of applications which provided by application developers, and an agent will request it in order to obtain necessary class files of applications.

3. Profile Repository Service

   The profile repository service provides the functionality to make gateways obtain

profiles through it; profiles, in which we defined some attributes to describe heterogeneous execution environments, is for context-aware computing.

The profile repository service contains two kinds of profiles:

(1) Component Profile: describe the relationship between contextual components and user profiles。

(2) User Profile: includes User Device Profile and User Preference Profile for describing the capability of users' devices and whose preferences respectively.

4. Location Service

   In order to trace the location of clients, the location information should be recorded when clients start to use service or move from some local area network to another.

3.3 Client Tier

Clients mean devices of end-users in which several components are designed for application developers to program.

In client device, a user agent is the critical component, which is responsible for:

1. Issuing a request to a gateway for registration
2. Maintaining user's state
3. Providing APIs, which hide the complexity of the low level in users' devices for convenience of applications development

3.4 Allocated Agent

Within our system framework, we partition a program into several object units: one user corresponds to one agent composed of several applications, which are not concrete. An application, as a matter of fact, is an execution unit conceptually, like Outlook, a necessary application when we receive e-mails. This Outlook is the very application. In the operation, we keep tracks of which applications are running at the moment and what functions the applications have. A function can be regarded as the functions in Outlook, just like e-mail reading and sending. In our design, an agent is composed of several functions executed in some applications, all of which are made up of objects and components.

Take the description above as reference. We can observe agent could be composed of functions, and each function ($Function_1$, $Function_2$...) is composed of different components (C_{11}, C_{12}, ..., C_{nk}). This partition has three advantages basically:

1. Programs can be composed dynamically and automatically.
2. Only the components of the agent need to be changed.

3. These components can be updated on versions.

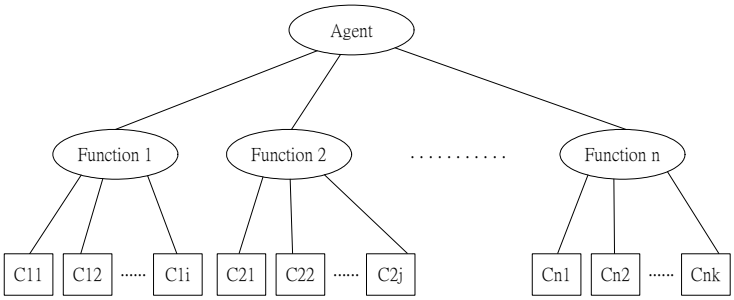

Fig. 6. The structure of an allocated agent

4. Context data modeling

In this section, the context data modeling include the component profiles and the user profiles in our system are discussed in detail.

4.1 Component Profile

Component profile is necessary. It would be applied in the auto-reconfiguration of allocated agents when users move from one local area network to another. In the scenario, the context of users' execution environment varies after mobility. A component profile can provide the related information, just like the relationship between attributes and contextual components, for reconfiguration of contextual components in an allocated agent to make a user execute his application as a consequence. The schema of these profiles is based on RDF. The part of description in a component profile is illustrated as below, Table 1. From the left side to the right: the first vertical column is filled in names of contextual components; the second is attributes corresponding to related contextual components; then the third is domain of attribute values.

Table 1. The schema of Component Profile

Contextual Components	**Associated Attributes**	**Values**
Compression	ApplyCompress	{yes, no}
	CompressAlgorithm	{ZIP, RAR, …}
Uncompression	ApplyUncompress	{yes, no}
	CompressAlgorithm	{ZIP, RAR, …}
Explode Text	MaxMsgSize	{1024, 1400, …}
	OutputCharSet	{US-ASCII, ISO-8859-1}
Convert to Gray	ImageCapable	{yes, no}
	ColorCapable	{yes, no}

Reduce Image Color Bits Per Pixel	BitsPerPixel	{2, 8 , …}
Resize Image	ScreenSize	{160x160, 640x480, …}
	ResizeImage	{yes, no}
Spelling check	CheckSpelling	{yes, no}
Encrypt Data	ApplyEncrypt	{yes, no}
	EncryptAlgorithm	{RSA, AES, …}
Decrypt Data	ApplyDecrypt	{yes, no}
	DecryptAlgorithm	{RSA, AES, …}
Language Encoding Transformation	CcppAccept-Language	{zh, en, …}
Speech to text	VoiceInputCapable	{yes, no}
Text to speech	TextInputCapable	{yes, no}
	SoundOutputCapable	{yes, no}

4.2 User Profile

User Profile is composed of two parts - User Device Profile and User Preference Profile. A user device profile is a document describing the capabilities and the configuration of a client device. For a client, many profiles about various devices may exist in the meantime. A user preference profile describes a client's preferences of applications or our system configuration about what if users want or not. The main difference between the user device profile and the user preference profile is that a client can has only one user preference profile, on the grounds that the number of user device profile is depended upon the number of devices the user has. Yet the user preference profile is not.

4.2.1 User Device Profile

User Device Profile describes the hardware configuration of Nokia 2160. In this profile, the default attributes of Nokia 2160 are described. The values of these attributes may be modified by users; like that the value of **Memory** showed below is modified from 16 MB to 32 MB, when client's device differs from the original in this document.

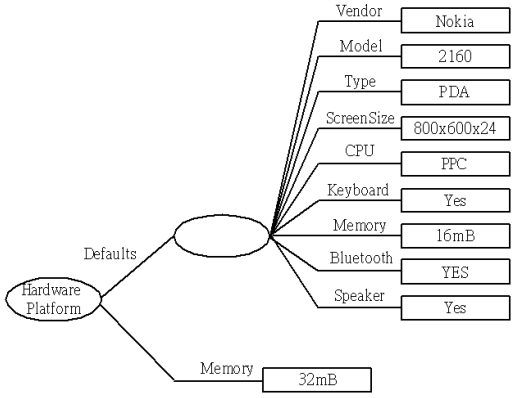

Fig. 7. The RDF graph of User Device Profile

4.2.2 User Preference Profile

User Preference Profile describes clients' preferences, some of which are the kind of languages, the availability of users and images, etc. A gateway can provide services dependent on these preferences. Take **Language** for example. It is the attribute set to English meaning that the client prefers to use English in the devices he owns. Another example is **Available**. The attribute refers that: if he does prefer communication with others, then set YES; if he does not, set NO.

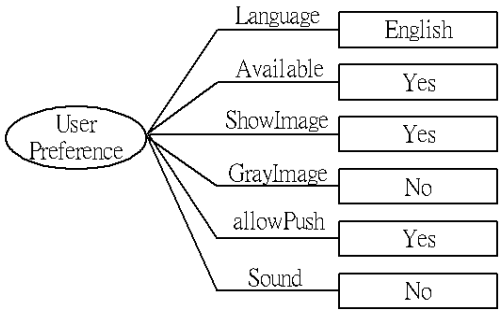

Fig. 8. The RDF graph of User Preference Profile

5. The interaction of system components

5.1 The Approach of Agent Mobility

To handle mobility of agents, we propose two approaches – Agent Migration Approach (AMA) and State Transfer Approach (STA), for the agent reconfiguration procedure while a client moves to another network and issues a registration request message. Therefore, the original gateway is requested by the new visited gateway for the allocated agent of the client.

5.1.1 Agent Migration Approach (AMA)

Agent Migration Approach (AMA) is the first approach applied while a client moves to a new visited network, and then the allocated agent serving him will be completely transferred to the visited gateway. This approach facilitates the allocated agent moved, which needs no reconfiguration. In fact, it can not only eliminate the efforts of programming, but also avoid spending time on reconfiguration in the new visited gateway.

Figure 9 illustrate this complete approach as follows:

1. The User Agent in the client device receives an **advertisement message** issued by Agent Manager 1 of the gateway in the new visited network.
2. The User Agent issues a **registration request message** to the Agent Manager 1.
3. The Agent Manager 1 issues an **entire agent request message** to the Agent Manager 2 in original network in order to obtain the entire agent.
4. If the Agent Manager 2 transfers the entire agent, it will issue an **entire agent reply message** with entire agent to the Agent Manager 1; otherwise this message will only include the state of the agent. This procedure will be discussed in the second approach in next section.
5. After the arrival of the agent, the Agent Manager 1 decides whether the agent should reconfigure or not, according to its environment configuration. Such being the case, the agent can conform to the new context in the visited gateway.
6. After the Agent Manager 1's making sure the agent is ready, it will issue a **registration complete message** to the User Agent to confirm the procedure is complete if ok, or a **registration reject message** will be returned to the User Agent.
7. Besides the above, the registration procedure fails while the following cases occur, and then **a registration reject message** is issued to the User Agent.

   (1) The Agent Manager 1 receives the reject message, such as an **agent state reject message**.

   (2) The timer expires during the time the Agent Manager 1 is waiting for replying from the Component Manager or the Agent Manager 2.

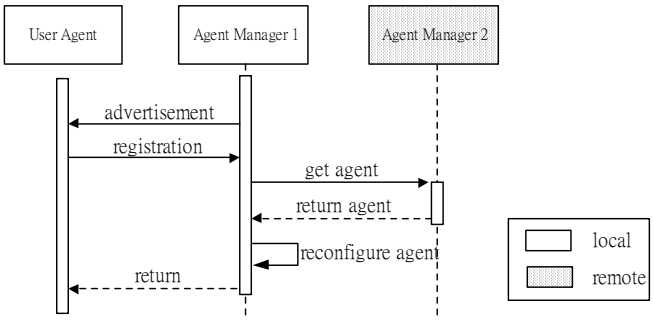

Fig. 9. The sequence diagram of Agent Migration Approach

5.1.2 State Transfer Approach (STA)

State transfer approach (STA) is the second approach applied while a client moves to the new visited network, and then the user's state of the allocated agent serving him will be transferred to the gateway in the new destination. The main contrary to AMA is that this approach only transfers the minimal information– the user's state to leverage the reconfiguration of the agent in the transferring throughput sensitivity. By doing so, the usage of bandwidth between gateways can be reduced.

Figure 10 illustrate this complete approach as follows:

1. The User Agent in the client device receives an **advertisement message** issued by the Agent Manager 1 of the gateway in the new visited network.
2. The User Agent issues a **registration request message** to the Agent Manager 1.
3. The Agent Manager 1 issues an **agent state request message** to the Agent Manager 2 in original network in order to obtain the state of the agent.
4. If the Agent Manager 2 provides, it will issue an **agent state reply message** with this agent state; otherwise an **agent state reject message** will be returned to the Agent Manage 1.
5. After receiving the **agent state reply message** with the state, the Agent Manager 1 will initiate the reconfiguration procedure.
6. At the beginning of this procedure, it issues a **component request message** to obtain the contextual components from the Component Manager in order to assemble these components with state into an entire agent.
7. When the Component Manager receives the **component request message**, it will return these requested components as possible as it can.
8. When the Agent Manger 1 receives the **component reply message**, it will assemble these received components and the state.
9. After the Agent Manager 1's making sure the agent is ready, it will issue a **registration**

complete message to the User Agent to confirm the procedure is complete if ok, or a **registration reject message** will be returned to the User Agent.

10. Besides the above, the registration procedure fails while the following cases occur, and then **a registration reject message** is issued to the User Agent.

    (1) The Agent Manager 1 receives the reject message, such as the **agent state reject message**

    (2) The timer expired during the time the Agent Manager 1 is waiting for replying from the Component Manager or the Agent Manager 2.

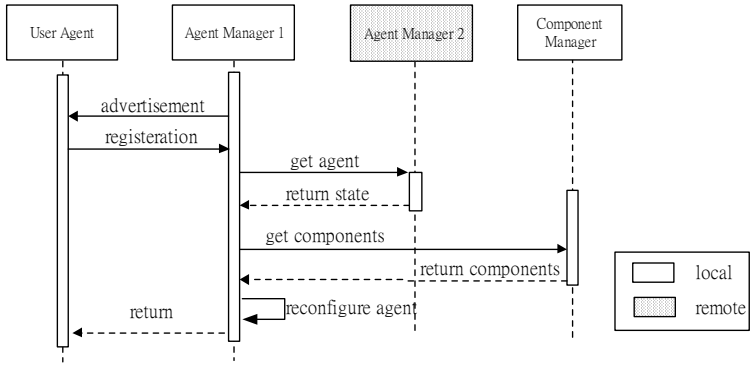

Fig. 10. The sequence diagram of State Transfer Approach

5.2 User Profile Change

The handling process of user profiles change, differing from cases stated in last section, is the adaptation mechanism to start the agent reconfiguration procedure to make each client's application run continually. However, it occurs when their user profiles change due to the operation of users' running applications, such like changing preferences. There can be two repository services applied to the reconfiguration procedure. First, the profile repository service provides the functionality to access user profiles in this mechanism; second, component repository service provides contextual components for the agent reconfiguration procedure.

Figure 11 illustrates a user profile of a client changed, when he set some preference of an application function. Then, the agent reconfiguration procedure will start in order to conform to new context.

The detailed following steps:

1. The User Agent issues a **profile change request message** to the Allocated Agent.
2. While the Allocated Agent receives the **profile change request message**, it will start the agent reconfiguration procedure.

3. In the commencement, the Allocated Agent will issue **a profile update request message** to the Profile Repository Service.
4. The Profile Repository Service will return a **profile update reply message** with the profile requested.
5. When the Allocated Agent receives this profile, it will check what component is needed for the agent reconfiguration procedure. Then, it will issue a **component request message** to the Component Manager.
6. When the Component Manager receives this message, it will issue a **component reply message** with the requested components to the Allocated Agent. Then, it will reconfigure.
7. After the Allocated Agent's making sure it is ready, it will issue a **profile update complete message** to the User Agent for confirming the procedure is complete if ok, or a **profile change reject message** will be returned to the User Agent.
8. Besides the above, the following case occurs: the Profile Change Procedure fails, and **a profile change reject message** will be issued to the User Agent.

   (1) The Allocated Agent receives the reject message, such as the **profile update reject message.**

   (2) The timer expired during the time the Allocated Agent is waiting for replying from the Component Manager or the Repository Service.

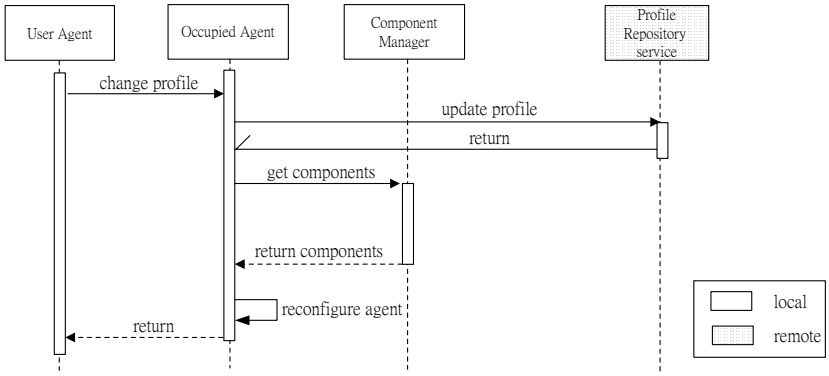

Fig. 11. The sequence diagram of profile change procedure

6. Conclusion and Future Works

In this paper, we propose the framework for pervasive computing in wireless network based on mobile agent paradigm, defining three kinds of profiles to be the description of the context data– Component Profile, User Device Profile, and User Preference Profile respectively. Moreover, by XML parsing techniques and the loading techniques of dynamic

class loader in JAVA, agent can be reconfigured automatically to be adapted to new environments when personal mobility and terminal mobility arise. We also propose two approaches, Agent Migration Approach (AMA) and State Transfer Approach (STA), to provide programs development choice for program designers.

At the present time, we propose two ways of agent mobility, AMA and STA separately. We cannot exactly judge which one is the best absolutely, nevertheless, we will measure their performance depended on the size of agent, the numbers and size of components, etc, in some scenarios. Afterward, through the analysis of the two approaches of agent mobility, we can strengthen our system to decide what approach to be performed automatically and intelligently.

In the future, we will make further exploration into several distributed issues, for example, the security issue, the fault tolerance issue and the load balance issue in the gateway.

As to the future research, we can integrate the Web Service [11] and MACAF, or apply MACAF to GPRS and UMTS toward reaching the objective of pervasive computing wherever and whenever proper.

References

[1] Danny B. Lange, Mitsuru Oshima, *Programming and deploying Java mobile agents with Aglets*, Addison-Wesley, 1998.

[2] IBM Aglets, http://aglets.sourceforge.net.

[3] Kovacs, E., Rohrle, K., Schiemann, B.,"Adaptive mobile access to context-aware services", *Agent Systems and Applications 1999 and Third International Symposium on Mobile Agents. Proceedings*. First International Symposium on 1999 Page(s): 190 –201.

[4] Prototyping Context-Aware Mobile Applications, http://www.cc.gatech.edu/fce/cyberguide /pubs/chi96-cyberguide.html.

[5] A System Architecture for Context-Aware Mobile Computing, http://citeseer.nj.nec.com/schilit95system.html.

[6] B. Schilit, N. Adams, and R.Want, "Context-Aware Computing Applications", *Proc. of Workshop on Mobile Computing Systems and Applications*, pp. 85-90, 1995

[7] WAP User Agent Profile Specification, http://www.wapforum.org/what/technical/SPEC-UAProf -19991110.pdf.

[8] CC/PP, http://www.w3.org/Mobile/CCPP/, http://www.w3.org/TR/CCPP-struct-vocab/, http://www.w3.org/TR/2000/ WD-CCPP-ra-20000721/.

[9] RDF, http://www.w3.org/RDF/, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[10] Grasshopper, http://www.grasshopper.de/.

[11] Web Services, http://java.sun.com, http://www.ibm.com.

[12] T. Bray, J. Paoli, C. Sperberg-McQueen, et al, *Extensible Markup Language (XML) 1.0, 2nd ed.*, W3C Recommendation, http://www.w3.org/TR2000/REC-xml-20001006

[13] G. Chen and D. Kotz, *A Survey of Context-aware Mobile Computing Research*, Technical Report TR2000-381, Department of Computer Science, Dartmouth College, United Kingdom, 2001.

[14] J. Gosling, B. Joe, and G. Steele, *The Java Language Specification*, Addison-Wesley, 1996.

[15] D. Lange, *Java Aglet Application Programming Interface (J-AAPI) White Paper – Draft 2*, IBM Tokyo Research Laboratory, 1997.

[16] Jean Bacon, "Toward Pervasive Computing", *IEEE Pervasive Computing*, 2001.

[17] Stephen S. Intille, "Designing a Home of the Future", *IEEE Pervasive Computing*, 2002.

[18] Nikos Anerousis, and Euthimios Panagos, "Making Voice Knowledge Pervasive", *IEEE Pervasive Computing*, 2002.

[19] JAXP API, http://java.sun.com/xml/