Available online at www.sciencedirect.com
ScienceDirect
Computers in Industry 59 (2008) 55–68
COMPUTERS IN INDUSTRY
www.elsevier.com/locate/compind

Mobile multi-agent based, distributed information platform (MADIP) for wide-area e-health monitoring

Chuan Jun Su *

Department of Industrial Engineering and Management, Yuan Ze University, 135 Far-East Road, Chung-Li, Taoyuan, Taiwan, ROC

Received 4 January 2006; received in revised form 1 March 2007; accepted 4 June 2007
Available online 17 September 2007

Abstract

e-Health care is characterized by high degree of distributed, labor-intensive works, mobility, and information access from various types of devices. Current and emerging developments in mobile computing integrated with developments in pervasive agent technologies such as mobile agent will have a radical impact on future health-care delivery systems.

In this paper we present the design and architecture of a mobile multi-agent based information platform – MADIP – to support the intensive and distributed nature of wide-area (e.g., national or metropolitan) monitoring environment. To exemplify the proposed methodology, an e-health monitoring environment was built on top of MADIP as an illustration. Aglets Software Development Kit (ASDK) was adopted for prototyping, concept-proofing, and evaluation. By integrating the proposed platform with light-weight, portable monitoring devices (e.g., portable vital sign monitor), continuous long-term health monitoring can be performed without interfering with the patients' everyday activities and without restricting their mobility. The optimal utilization of medical resources can be also achieved.

Keywords: Mobile multi-agent; Distributed information platform; e-Health monitoring; Aglets Software Development Kit (ASDK)

1. Introduction

The rapidly rising Internet technologies have helped drive the growth of electronic-health (e-health) in recent years. e-Health means leveraging the Internet to support a variety of health care information, products, technologies, and/or services. Typically, the e-health space could be divided into three different segments—content, commerce, connectivity [1], and now e-health is emerging to the new era, care space, to provide Internet-based care services. The e-health care indicates to record, measure, monitor, manage, and in the end to deliver patient-oriented along with condition-specific care services through the Internet in real time.

The telemedicine system, a currently used information system, enabled to maximize the collection, delivery, and communication of health care information, clinical messages, nursing interaction, and medical records from one location to another in e-health fields [2]. The telemedicine system has been a very promising tool to facilitate information exchanges among users such as generalists, specialists as well as patients, etc. and it also offered patient–physician tele-consultations over a distance. The inception of telemedicine systems encouraged the integration of many clinical activities in e-health; however, these systems are ineffective for providing wide-area e-health monitoring in which clients are usually mobile and situate in a low bandwidth, high latency, asynchronous transaction, and unstable connection communication environment while servers are highly distributed and heterogeneous. In this context, mobile agent (MA) technology is more suitable as it provides powerful and efficient mechanisms to develop applications for a distributed and heterogeneous environment.

To deal with the increasing amount of data/resources in e-health monitoring scenarios, and the large number of tasks which have to be performed to manipulate the data/resources, mobile agent technology offers the possibility of executing these tasks in an automated way, with minimal human intervention. This allows medical staff to concentrate attention on other activities and subsequently save valuable medical resources. Mobile agent technology provides basic functions such as creation, migration, execution of the mobile agents, as

* Tel.: +886 34638800x2529; fax: +886 34638907.
E-mail address: iecjsu@saturn.yzu.edu.tw.

0166-3615/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.compind.2007.06.001

well as specialized functions that involve agent security and management. The benefits of mobile agents consist of overcoming the limitations of a client device, customizability, higher survivability, asynchronous and autonomous computing, and local data access and interoperability:

- *Overcoming limitations of a client device*—Such as communication delays and throughput, memory size, processing power, and small storage, may be achieved if the agent is executed in the proximity of data source. For example, if a database needs to be searched according to an algorithm, the performance of searching can be improved by sending an agent to the database compared to accessing data remotely. In addition, the local computer may not have sufficient data storage to temporarily store large amounts of data, or may have insufficient network bandwidth or processing power.
- *Customizability*—It is comparably more difficult for the traditional client–server model to adapt to frequent changes while MA can be easily customized to user needs, and sent to the server where customized requests are executed. In this study, requests are sent by using SQL to the targeted MA server. The MA server represents a processor that accepts and executes accepted agents.
- *Higher survivability*—Because agents transfer both code and state encapsulated within the mobile agent abstraction, they have a higher degree of survivability compared to the client–server model.
- *Asynchronous and autonomous computing*—Mobility is increasingly important in terms of user requirement. Jobs started from mobile devices frequently need to continue while a user is disconnected. While it is possible to delegate this responsibility to a stationary proxy residing on a network, it is even more convenient to delegate it to a mobile agent that will pursue its owner's tasks even while the user is disconnected. The agent can be pulled back from its current location when the user is back online. Mobile agents have the ability to interact with their execution environment on which to act asynchronously and autonomously. They simply act continuously in pursuit of their own goals.
- *Local data access and interoperability*—Mobile agents access the data locally, which have at least two advantages (1) low traffic in the network and (2) a better use of network resources. The migration of a mobile agent within the network follows protocols concerning the agent transport, agent interaction and security to ensure interoperability among mobile agents which might adopt different operating systems.

1.1. Mobile agent systems

Several studies have been focused on mobile agents (MA) that may be considered as an emerging designed paradigm in the distributed environments [3–5]. Picco [3] gave an overview of the fundamental characteristics concerned with migration which is taken as the core support for mobility to ensure this new paradigm can deal with the problems encountered. Several benefits were explored in [5] to encourage the adoption of MA such as decentralized processing, better use of communication resources, and the most prominent feature is to support for designing applications that interact with human users. Vuong and Ivanov [4] examined and contrasted two systems at the scale in their ability to support program mobility: Java and Wave. Java offers a useful combination of some of the most attractive languages and environments to extend the implementation of mobile intelligent agents while Wave, another new programming paradigm, directly supports dynamic creation and processing of arbitrary knowledge networks. Lange and Oshima [6] made a comparison among three distributed computing paradigms in networks: client–server, code-on-demand, and mobile agent. A summary of the benefits using

Table 1
List of commonly used mobile agent systems

No.	Name	Developer	Language	Application
1	Agent Tcl	R. Gray, U Dart.	Tcl Tk	Information management
2	AgentSpace	Ichiro Sato, O. U.	Java	General purpose
3	AgentSpace	Alberto Sylva	Java	Support for dynamic and dist. Appl.
4	Aglet	IBM, Tokyo	Java	Internet
5	Ajanta	Minoseta U.	Java	General purpose
6	Ara	U Kaiserslautern	C/C++, Tcl, Java	Partially connected c. D.D.B.
7	Concordia	Mitsubishi E.I.T.	Java	Mobile computing, Data base
8	JATlite	Standford U.	Java	Information retrieval, Interface agent
9	Kafka	Fujitsu Lab. Japan	Java, UNIX-based	General purpose
10	Kali Scheme	NEC Research I.	Scheme	Distributed data mining, load balancing
11	Knowbots	CNRI	Python	Distributed systems/Internet
12	Messengers	UCI	C (Messenger-C)	General purpose
13	MOA	OpenGroup, UK	Java	General purpose
14	Mole	Stuttgart U. Germany	Java, UNIX-based	General purpose
15	OAA	SRI International, AI	C, C-Lisp, Java, VB	General purpose
16	Odyssey	General Magic	Telescript	Electronic commerce
17	Plangent	Toshiba Corporation	Java	Intelligent tasks
18	Tacoma	Norway & Cornell	C, UNIX-based,	Client/Server model issues/OS support
19	The Tube	David Halls, UK	Scheme	Remote execution of scheme
20	Voyager	ObjectSpace	Java	Support for agent systems

mobile agent in distributed network computing is also depicted in their work.

As outlined by Naylor et al. [7], each MA environments that have been developed possesses its own functionality: interoperability, migration option, language used, ease of implementation, and many other functionalities. The developer has to choose one in accordance with the requirements and objective of the targeted project.

Naylor [8] evaluated three mobile agent development toolkits, Voyager from Object Space, JATLite and the Aglets Software Development Kit (ASDK) from IBM. ASDK is recommended as the most suitable for system prototyping. Tracy Toolkit [9] that has been developed at Friedrich Schiller University Jena in Germany also gains popularity within mobile agent research society.

Hohl [10] compiled and listed in his web site 72 available mobile agent systems that include Aglet, Concordia, Odyssey, Voyager, etc. Table 1 summarizes the commonly used mobile agent systems for research and applications.

The widespread use of MA has been discussed in the arena of electronic commerce (EC) by a number of authors [11–15]. Nevertheless, little attention has been devoted to resolving current issues experiencing in the health care industry.

2. Requirement analysis and system architecture

2.1. Requirement analysis

Health monitoring has been traditionally a time-consuming and costly process which requires patients to frequently visit hospitals. It is relatively inconvenient for some populations (e.g., elders, surgical patients, pregnant women, etc.) to be present at the hospitals or clinics on a long trip regularly. However, these populations usually require frequent vital signs (e.g., heart rate, temperature, pulse, blood pressure, etc.) check-up to ensure their health condition. With the growth of this population, e-health care is now facing a profound challenge to provide better public health care especially in vital signs monitoring aspect.

Moreover, scarce medical staffs are occupied with observing physiology parameters on the monitor screens sitting in a monitoring center on a 24-h-per day, 7 days-per week basis. The monitored task is a tedious duty for medical staffs to simultaneously audit and interpret the massive amounts of information regarding a patient during the process of monitoring, performing diagnostics, and verifying therapeutic intervention. The complication have arisen from this situation that each medical staff may be associated with multiple patients, yet there is no way for hospital personnel to be released from this routine and time-consuming tasks in spite of the severely understaffed with fluctuating numbers of patients. Hospital staffs have to distinguish the time critical situation and/or sometimes they are required to make vital decisions. Consequently, there is an urgent need to develop a system that is capable of performing wide-area health monitoring automatically and autonomously to users who are usually mobile and situate in a low bandwidth, high latency, asynchronous transaction, unstable connection environment. In addition, the system also needs to accommodate the situation that wide-area health monitoring data is stored and distributed in heterogeneous local servers. To satisfy these aforementioned requirements, a mobile agent information platform MADIP that allows MAs to work on behalf of medical staffs, to collect distributed users' vital sign data, and to spontaneously inform abnormal situations to associated physicians in real time is proposed in this research.

2.2. MADIP framework

MADIP is composed of hosts running the agent platform and hosts running supporting services such as a directory agent and a naming agent. The directory agent provides supervisory control over services that other agents provide in the environment. It serves as 'yellow pages' in which agents have been registered. Agents may query the directory agent to explore the types of services that can be acquired in the environment. The registration with the directory agent is mandatory for all agents to provide services that other agents need for collaboration in MADIP. The naming agent provides supervisory management over access to the agents located in different hosts. Naming offers 'white pages' services to other agents based on the directory of agent identifiers it maintains, and contains transport addresses for running agents. All agents in MADIP have to register with the naming agent in order to obtain an agent identifier before being activated. An agent platform serves as the agent execution environment, which controls agents: it creates them, executes them, transfers and terminates them. Agent platforms can be added and removed dynamically to the environment. This allows for large scale, even Internet wide installations. In addition, external services such as SMS, Fax, etc. may be also installed and integrated into the environment. By the use of these external services, agents may communicate with systems outside of the agent environment.

MADIP is open in the sense that agent platforms may join and leave the environment without intervention by a system administrator. The installation of an agent platform may be on servers, desktops and mobile devises such as laptops, PDAs. External services which enable agents to communicate with systems outside of the agent environment may also be installed in the agent environment. The overall framework of proposed MADIP is depicted in Fig. 1.

2.3. System architecture

The proposed MADIP was built on top of the environment mentioned in the previous section, which is suitable to operate in a heterogeneous, networked environment such as the Internet to provide wide-area (nation or metropolitan wide) health monitoring service or other types of monitoring services. MADIP is composed of two types of agents: (1) static agents (SA) that provide resources and facilities to mobile agents and (2) mobile agents (MA) that move between network domains taking advantage of these resources to fulfill their goals and

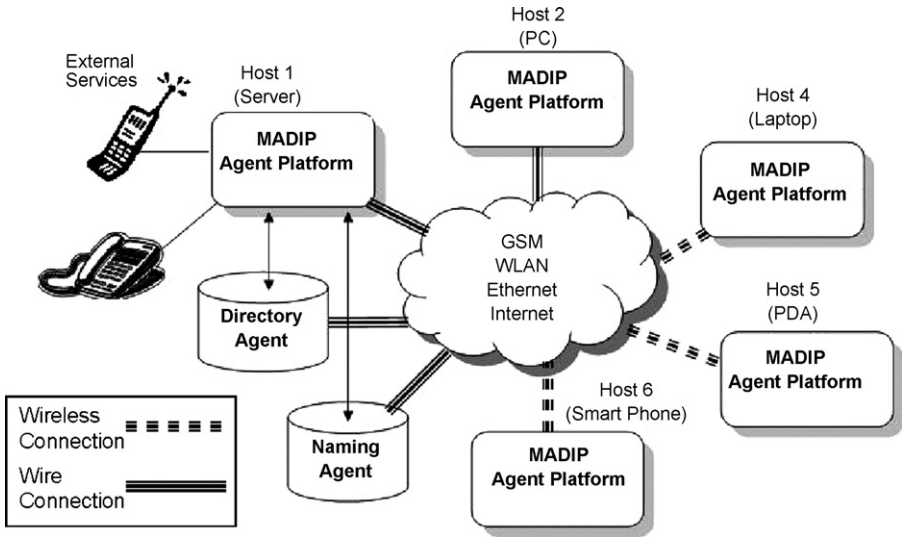

Fig. 1. The overall framework of MADIP.

work autonomously and may communicate with other agents and host systems. Agents use an asynchronous communication channel ''blackboard'' to communicate each other. The blackboard is maintained by the agent platform, and acts as a mailbox for agents. The information delivery is not automated; agents have to check the blackboard regularly for available information. The blackboard allows the storage of any kind of objects such as strings, images, XML, e-mail, etc. on the blackboard.

Six primary architectural features discriminate the proposed MADIP: (1) user interface agent, (2) agent service, (3) resource agent, (4) physician agent (5) diagnostic agent (6) data service, and (7) external services. The mobile agent architecture of this platform is illustrated in Fig. 2. Each component of the architecture and its general activities and purpose within the infrastructure is described below.

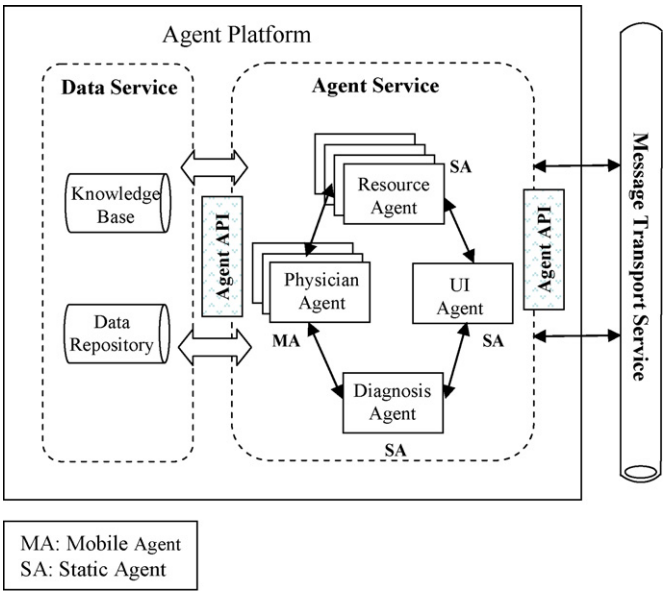

Fig. 2. The mobile agent architecture of MADIP.

2.3.1. User interface agent (UIA)

The UIA is a type of stationary agent that is used as a bridge to interface with host computers and applications. The UIA acts as an interpreter of heterogeneous agent and applications communications. It receives the requests from UI/application to invoke internal services. Hence, the UIA is an access control mechanism, which authenticates users before starting an agent service. The UIA is responsible for the final presentation of results by mobile agent or by diagnostic agent before passing data to the application layer.

2.3.2. Agent service

An agent service is interpreted as some kind of agent execution environment as described earlier. It provides base services which agent-based applications may utilize such as inter-agent communication. These services are offered through a dedicated Agent API and are always available. The Agent API comprises communication means for the agents, mechanisms to migrate to other hosts, and service lookup and security features. The agent API will examine the submitted code to guarantee that it conforms to the relevant protocols and does not violate security policy. Based on the identity of the agent creator, a set of credentials for the agent may also be generated at this time. These are transmitted as part of the agent, to allow other entities to identify it unambiguously. The mobile agents, which implement applications, can then be prevented from accessing system resources like the file system.

2.3.3. Resource agent

Resource agent is another type of stationary agent and conventionally, resource agent operates at a higher level of trust and mediates access resources from mobile agent to host computers. Resource agents are ''static'' because they do not have the ability to migrate; they usually reside on the host computers to provide expert advice or services locally.

Therefore, they are considered as secure and granted to access the resources of the host computer. In the proposed platform, resource agent plays an important role to dynamically interface with host's resources. It is through resource agent that mobile agent can have access to resources of the host system. The resource agent would take the place of the web sites to bridge the users to the databases.

2.3.4. Physician agent

The physician agent is a mobile agent used by medical staff. The physician agent is a computer program that can helps medical staffs to perform their tasks. The physician agent enables medical staffs to virtually monitor the conditions of patients and/or elderly in real time. In this unit, medical staffs may use their mobile devices (e.g., PDA, smart phone, etc.) to

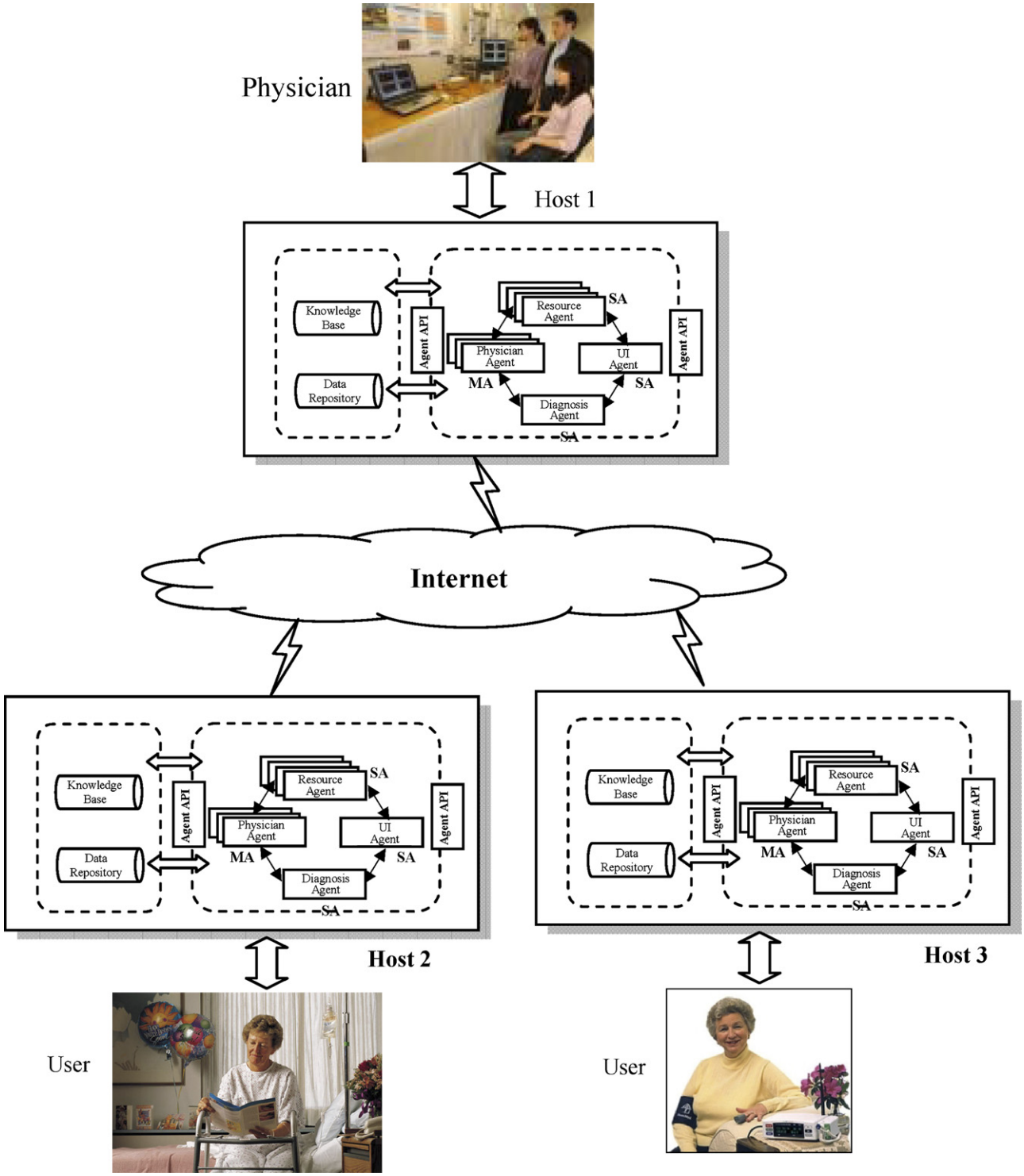

Fig. 3. The integrated operation of the MADIP with an example of one physician and two users.

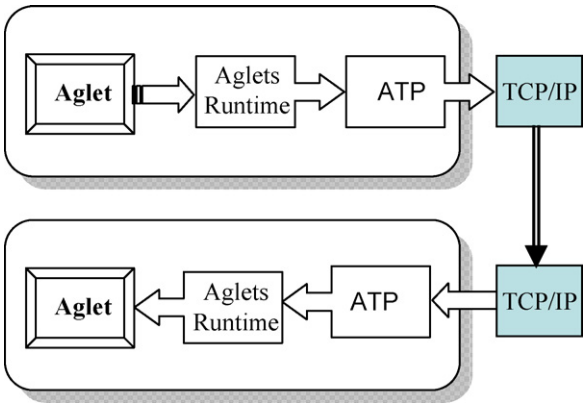

Fig. 4. The overall structure of the prototype.

trigger physician agents as delegates for them. For activating a physician agent, it must be dispatched to a specific agent platform. The agent platform authenticates the incoming physician agent using its credentials and determines the privileges to be granted to it. It then assigns a thread to execute the physician agent code. The physician agent will be subsequently sent to the Internet through wireless local area network (WLAN) or mobile network in the hospital. After connected to Internet, the physician agent will continue performing the assigned tasks depending on the characteristics of a given clinical case. The physician agent is devoted to data acquisition, after roaming in the networks to collect related physiological data and subsequently carry the collected information with it and to return to the host server.

2.3.5. Diagnostic agent

Diagnostic agent in the proposed platform can be treated as a data analysis engine. It is capable of analyzing data collected from a vital sign monitor or other forms of electronic monitoring devices. The first task of diagnostic agent is to check the collected data against the patient's personal profile that stores the criteria of abnormality for the patient, associated physician(s), history of medical treatment and other personal data. In the case of detecting anomalies, the diagnostic agent will utilize external services such as SMS or cellular phone call to inform the associated physician designated in the personal profile for taking appropriate actions.

2.3.6. Data service

It is evident that each person is an individual case with individual patterns, complications, and disease. In order to promote the performance of the proposed platform, a data service needs to be incorporated into the platform. The data service consists of two information repositories: 'user profiles' and 'data'. User profiles encapsulate anamnesis and physician defined criteria for abnormality while data repository is a database in which all vital sign check-up records are stored. Data repository is also used to keep track of system events, operations, agents in the system, and stores and maintains information of concerned activities and resources associated with the users.

The local agent server collects vital sign data from a user and transmits to resource agent that subsequently store in the data repository. A copy of the vital sign data along with the user's profile is also sent to diagnostic agent for real-time check-up.

2.3.7. External services

The external services contain the environment hardware and services including mobile phone, e-mail, and short message services (SMS), etc. The external service is an extensible component varied on different scenario of applications. In the

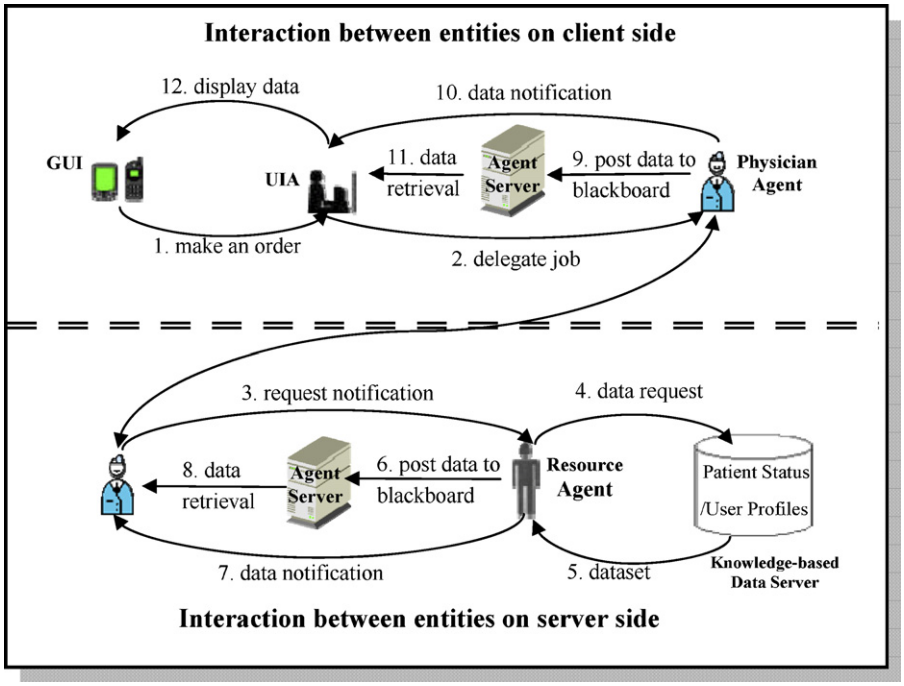

Fig. 5. Usage scenario from medical staffs' perspective.

Fig. 6. Three Tahiti servers and viewers running concurrently.

case that abnormality of a user is detected after analyzing the incoming physiological data against the user's personal profile, external services will be initiated by diagnostic agent to inform the responsible physician in real-time.

In the proposed MADIP, each agent is endowed with specialized capabilities and goals to perform tasks on behalf of physicians or users. The integrated operation of the MADIP, constructed with an example of one physician and two users, is depicted in Fig. 3.

3. Prototyping

This section describes the phase of MADIP prototyping in which the system design concept was implemented and tested by using Aglets Software Development Kit (ASDK) from IBM. The ASDK provides a mobile agent framework written in pure Java that facilitates the development of distributed applications using mobile agent architecture. The aglet derives its name from a combination of ''agent'' and ''applet''. Applet is event driven and provides methods that the programmer may override in order to control its life cycle, the ASDK provides mobility orientated and mobility triggered framework.

The overall structure of the prototype is depicted in Fig. 4. On the server side, there are aglet hosts (agent servers) and databases which store users profile and records of vital sign monitoring.

3.1. Hardware and software configuration

The prototype was built on top of Windows XP™ version 2002 with Java™ 2 Platform, Standard Edition Development Kit (J2SDK) installed. Microsoft SQL Server 2000 Personal Edition was adopted for Knowledge-base Data Server. A middle layer called Java Database Connectivity (JDBC) driver was installed for accessing the Knowledge-base Data Server. Aglet API and ASDK version 2.0.2 was used for implementation, which may be freely downloaded from: http://sourceforge.net/projects/aglets.

The hardware and software configuration of the prototype system is summarized in Table 2.

Fig. 7. The UIA main window in usage scenario 1.

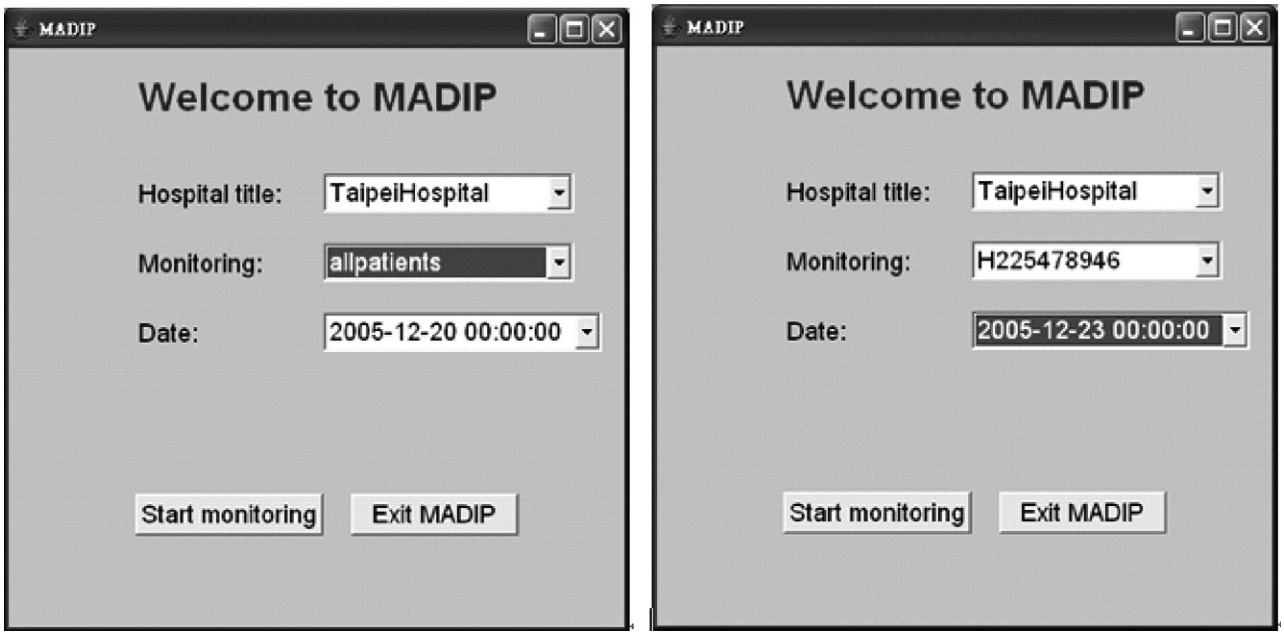

Fig. 8. Monitoring all patients' data and delegation a job to physician agent.

3.2. Database design

In the prototyping two databases TaipeiHospital and KaohsiungHospital were created. Each database consists of three tables namely Hospital, PatientStatus, and PersonalProfiles. The table ''Hospital'' encapsulates hospital ID and its name, ''PatientStatus'' comprises the physiological parameter of patients, and ''PersonalProfiles'' contains histories and abnormality guidelines of patients. The database design is shown in Table 3.

4. Implementation and usage scenarios

This research intends to design and develop a mobile agent based information platform to support the intensive and distributed nature of health care monitoring environment that allows physicians to acquire conditions of their patients and allows patients to check their vital signs at any time and anywhere. The pilot implementation was conducted with the assistance of Armed Forces General Hospital, Taoyuan County, Taiwan. Two types of users: medical staffs (physicians) and

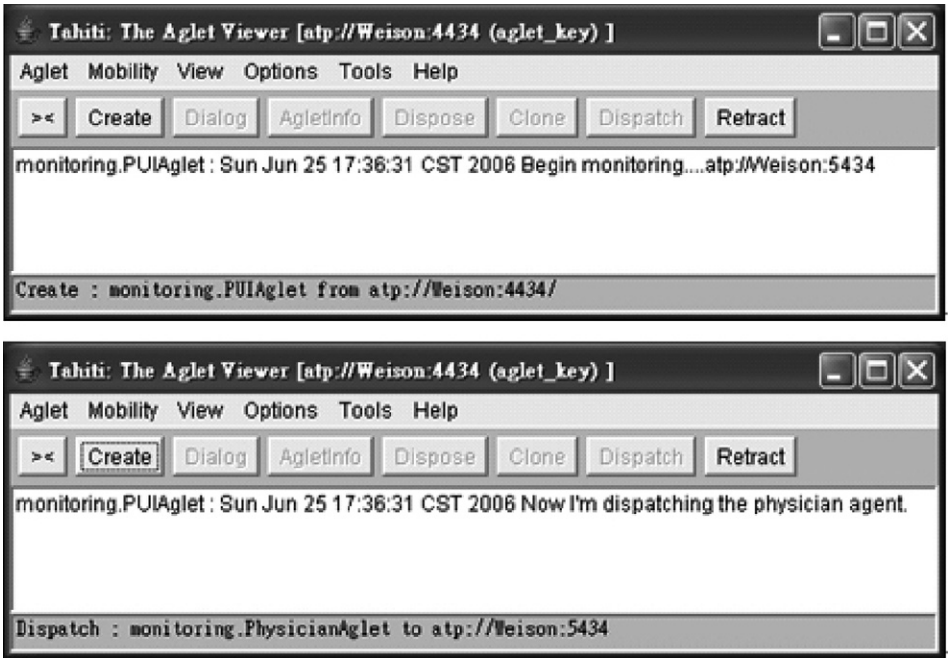

Fig. 9. UIA displayed the current state and dispatched a physician agent.

Table 2
The hardware and software configuration of the prototype system

CPU	Intel (R) Pentium (R) IV 3.00 GHz
Memory	480 MB
Operating system	Windows XP 2002
Java platform	Java 2 Source Development Kit (J2SDK) version 1.4.2_03
Agent platform	IBM Aglets Software Development Kit (ASDK) 2.0.2
Database	SQL Server 2000 Personal Edition
JDBC driver	Java Database Connectivity driver

regular users whose health needs to be monitored (users) were created and tested in a trial run of the prototype. In this section, we describe implementation scenarios from these two types of users.

4.1. Scenario 1: Physicians' perspective

It is time for Dr. Wang to perform a routine check for his patients. Through his PDA or mobile phone user interface, he interacts with the UIA and makes a request (step 1 of Fig. 5) to collect his patients' recent health data. Upon receiving the request, the UIA launches and delegates the job to a physician (mobile) agent (step 2 of Fig. 5) via a local agent server. Dr. Wang may then switch off his device and proceed with his daily work. In the mean time, the physician agent will surf in the logical mobile agent networks to acquire patients' data on behalf of Dr. Wang. When an agent server with needed information is found, the physician agent will send a request to resource agent for retrieving data (step 3 of Fig. 5). The physician agent will continue delivering message until it does all nodes of the network.

Upon receiving the request, the resource agent subsequently verify whether the physician agent is allowed to access the requested data. If the verification was passed, the resource agent will then retrieve the requested data set (steps 4 and 5 of Fig. 5) and post the data set on blackboard for the physician agent to fetch (step 6 of Fig. 5).

The physician agent migrates around the network and fetches the available dataset posted on the blackboard (steps 7 and 8 of Fig. 5). When the physician agent gets all the data needed, it will return to the same agent server that creates it,

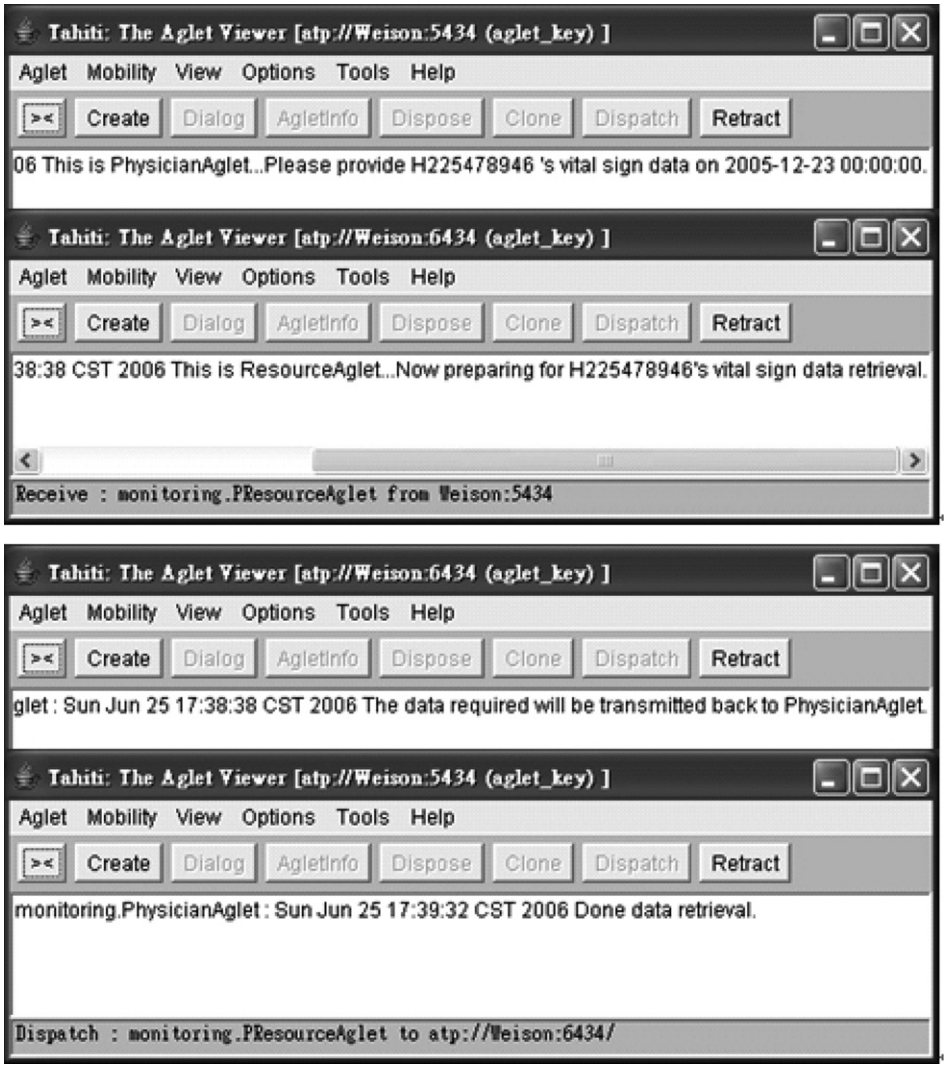

Fig. 10. The collaboration between the physician agent and resource agent.

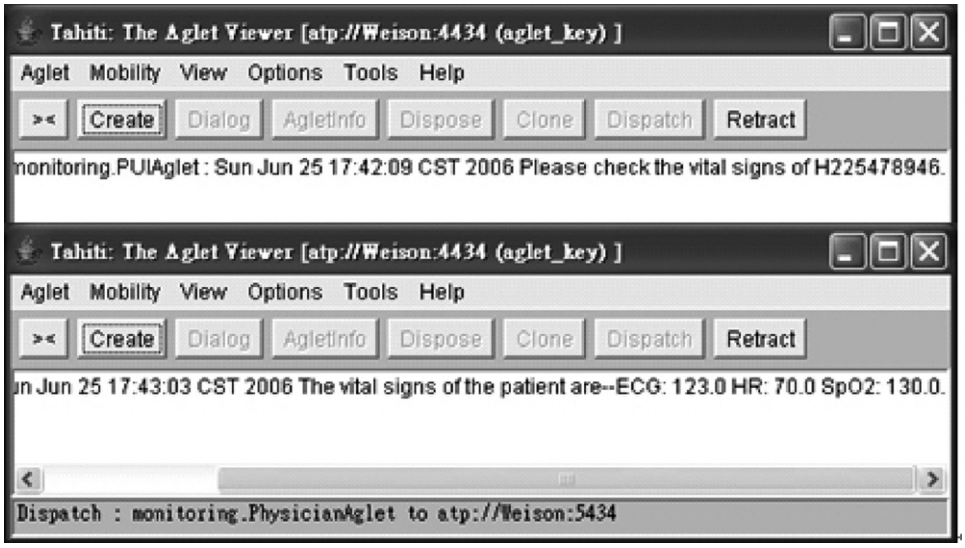

Fig. 11. The physician agent returned to the UIA where it was initiated with collected data.

and post the collected dataset to the blackboard (step 9 of Fig. 5). At the same time, it will notify the UIA that the data is ready (step 10 of Fig. 5). The UIA will retrieve the data from the blackboard and display the data on the application GUI (steps 11 and 12 of Fig. 5) when Dr. Wang is back on line with his device.

4.2. Scenario 1: Implementation

Within this implementation we created three Tahiti with three different port numbers, 4434, 5434, and 6434. The lowest Tahiti window indicates that it is a local host (with port 4434) while the other two Tahiti windows represent other hosts in a network as illustrated in Fig. 6.

When a physician desires to monitor the status of one of his patients, he/she firstly creates the PUIAglet as illustrated in Fig. 7.

The physician has the option of choosing "allpatients" or a specific patient as depicted in Fig. 8 for monitoring all the patients whom he/she is responsible for or a particular patient (e.g., H225478946) who needs to be concerned.

Upon pressing "Start monitoring," the Tahiti viewer displayed the current state of UIA and a physician agent was then initiated and dispatched to the network to perform delegated job as shown in Fig. 9.

Fig. 10 shows that the physician agent requested data and the resource agent verified the physician agent's identity while

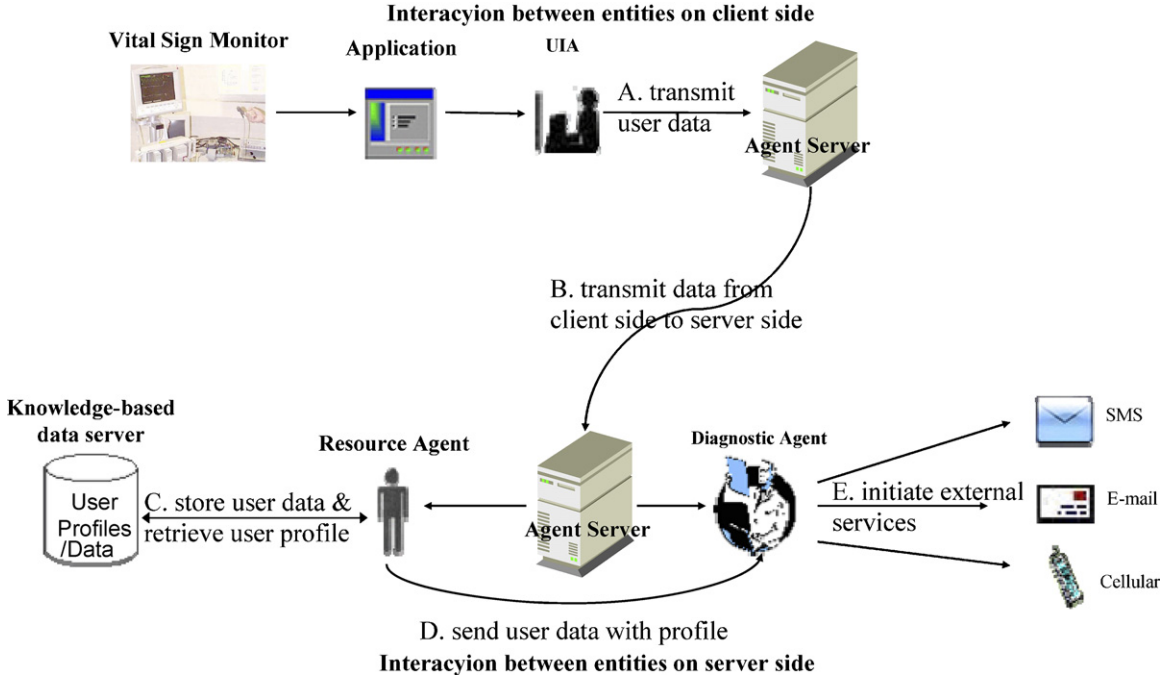

Fig. 12. Usage scenario from a user's perspective.

Fig. 13. The UIA main window in usage scenario 2.

Table 3
The database design of the prototype system

Table: Hospital

Field name	Type	Size	Description
HospitalID	Varchar	3	Key
HospitalName	Varchar	20	

Table: User profiles

Field name	Type	Size	Description
PatientID	Char	10	Key
PatientName	Varchar	15	
ECG	Float	8	
HR	Float	8	
SpO2	Float	8	

Table: Patient status

Field name	Type	Size	Description
PatientID	Varchar	10	Key
PatientName	Varchar	15	
MonitorDate	Small date time	4	
HostpitalID	Varchar	3	
ECG	Float	8	
HR	Float	8	
SpO2	Float	8	

checked the existence of requested data in database. If the identity possesses the clearance and data exists in database, the resource agent retrieved the data and passed it to the physician agent.

The physician agent carried the collected data and returned to the UIA where it was initiated.

The UIA then notified the physician that data has been prepared and asked him to double check. As soon as the physician is available for checking the patient's status, the vital sign information of the patient H225478946 will be displayed as illustrated in Fig. 11.

4.3. Scenario 2: User's perspective

While Mr. Smith is watching CNN news, he turns on his portable vital sign monitor to check his condition of health (blood pressure, temperature, pulse, heart rate, etc.). After checking, he registers with the local agent server by using his PDA via UIA. The physiological data is then transmitted to the resource agent in a local agent server (step A of Fig. 12). The resource agent stores the data to local data server, retrieves Mr. Smith's personal profile (steps B and C of Fig. 12), and subsequently sends a copy of the data along with the profile to the diagnostic agent (step D of Fig. 12). The data is then checked against the profile that stores Mr. Smith's criteria of abnormality. If abnormal had been detected, the diagnostic agent will notify the associated physician(s) immediately and suggest suitable follow-up procedures based on history of medical treatment by using external services such as SMS, cellular call, and/or e-mail (step E of Fig. 12).

4.4. Scenario 2: Implementation

Mr. Smith wanted to update his vital signs data that he just measured. By activating MADIP installed in his mobile device,

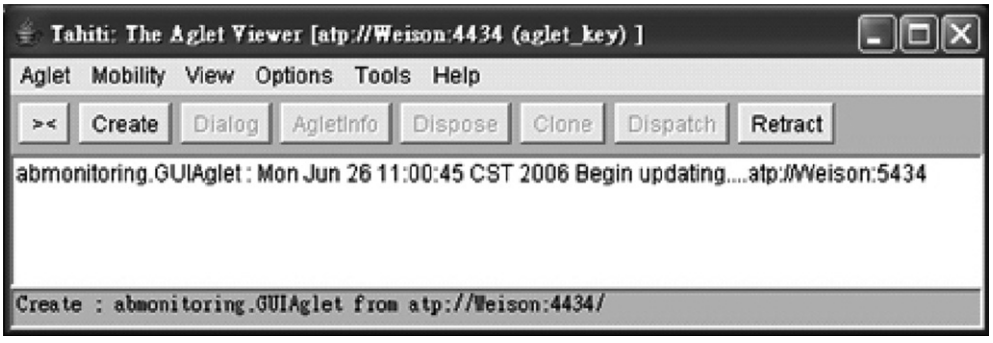

Fig. 14. Notifying resource agent for preparing to update data.

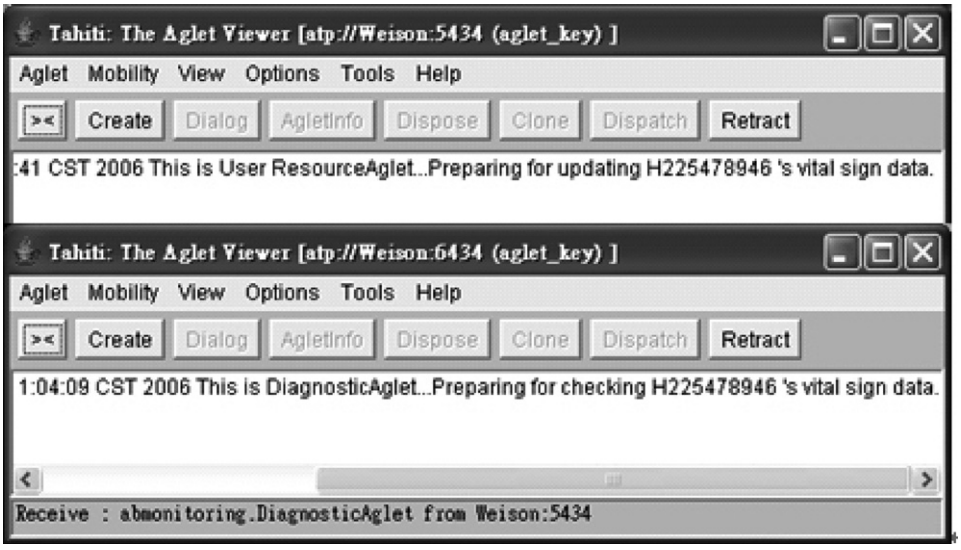

Fig. 15. The diagnostic agent prepares to examine the vital sign data.

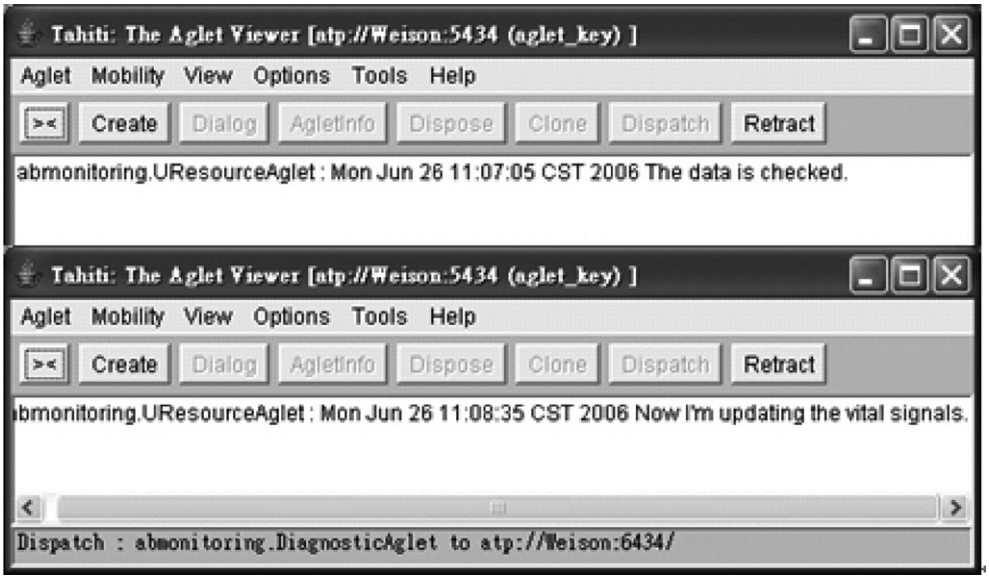

Fig. 16. The resource agent executes the data update.

the data was transmitted from vital sign monitor to the mobile device and displayed in UIA (GUIAglet) as shown in Fig. 13. The UIA subsequently started to inform the resource agent for preparing to update Mr. Smith's data, see Fig. 14. Upon receiving the notice from UIA, the resource agent renewed Mr. Smith's data and sent a copy to diagnostic agent with Smith's personal profiles simultaneously. The diagnostic agent then prepared to examine the vital sign data of Mr. Smith, as illustrated in Fig. 15.

In case an abnormality was detected, the diagnostic agent would instantaneously report the situation to corresponding physicians via external services. It was not until the diagnostic agent completed his job that the resource agent updated Smith's data. When diagnostic agent completed the data check, it communicated with resource agent. Eventually, the resource agent executed the data update, as represented in Fig. 16.

5. Conclusion remarks and future works

5.1. Conclusion

The development of MADIP or similar systems will be critical for the future of health care monitoring. From the medical staff point of views, the workload can be reduced by substituting routine patient supervision tasks. While from the patient point of views, patient care can be improved by immediate transit as well as communication between agents

with more reliable information and also it is delivered in a ubiquitous way. Efficiency can also be increased by readily exchange information as a matter of fact that permanent data is collected only once so less time is spent in looking for mislaid data. Ultimately, documentation is diminished as a consequence of the elimination of hand-written records, clinical reports, and medical errors, etc.

The MADIP takes a new opportunity to integrate and analyze the immense data encountered in patient monitoring. It could not only respond to inquiries for medical information on the Internet relevant to a person's medical history but also monitor the patient status to alert its owner about unhealthy trends in addition to the inconsistencies in the health record. It clearly presents an innovative way to assist health care practitioners through collecting, filtering and examining relevant information for a patient, providing basic diagnosis and suggesting actions in a powerful manner.

The trial implementation however reveals three issues which need to be addressed in our future works:

(1) Sophisticated diagnostic engine needed: the diagnosis is merely carried out in the pilot implementation by comparing vital sign monitoring data against a rather primitive guide line specified by physicians (e.g., systolic <120 and diastolic <80 mm Hg, pulse: 60–80 beats/min, etc.). Many false alarms were generated consequently.

    Frequently, the patient did not avoid coffee, smoking, unprescribed drug with sympathomimetic, or vehement sports activity on the day of the measurement. As a result many false alarms were sent to physicians. A more practical guide and sophisticated diagnostic engine need to be developed and integrated into MADIP.

(2) Security and privacy: When individuals conduct the transmission of health information over Internet there is a serious concern about that whether personal information would be divulged to others and would be protected from assault or corruption. Similar to the arena of Electronic Commerce, users would be hesitant using the system unless the issues of security and privacy are well addressed.

A basic security mechanism is introduced in the **ASDK** called capability-based mechanism that prohibits aglets link to any dynamic link library to provide some control over the security. More robust and comprehensive security mechanisms need to be incorporated in MADIP.

5.2. Future works

Mobile multi-agent systems have great potential to transform the process, and possibly even the outcome, of health care monitoring. It is necessary that innovation goes hand in hand with evaluation. The next reasonable step is to evaluate the pilot trial in designing an open and scalable monitoring platform that allows advanced modules to be directly incorporated in and expands coverage areas of health care smoothly. For example, the diagnostic agent plays a key role in MADIP and it is insufficient to perform the diagnosis simply based on some simple guidelines. By incorporating sophisticated diagnostic methodology into MADIP the functionality of diagnostic agent can be enhanced. As an essential extension of this work, robust diagnostic algorithms will be explored to achieve desired level of diagnostic accuracy and integrated into the proposed system to enhance the functionality of the diagnostic agent and consequently strengthen the entire system.

Another critical issue remained to be addressed is the system security. While the advantage of mobility in the agent paradigm helps us in developing distributed health care systems and reaching a large section of mobile clientele, it also opens the system and makes it vulnerable to attacks by malicious entities. We adopted IBM ASDK as our tool for prototyping, which equips with built-in basic security mechanism. Nevertheless, mobile agent is a trust-based paradigm; the security issue has far reaching implications and is a hindrance in its acceptance and uniform spread. Robust agent security frameworks [16–18] for controlling risk and mitigating the threat posed by malicious entities need to be explored and integrated in MADIP.

Acknowledgements

I wish to thank the Tao-Yuan Armed Forces General Hospital for their kind support of providing vital signs monitoring guide-line and device. My special thanks to many students in the ERP/MC laboratory for the implementation and testing the prototype system. The experiment of the proposed research cannot be done without their dedication.

References

[1] B. Stan, Evolution of the eHealth space, Pharmaceutical Executive 2000, ABI/INFORM Global 2000, pp. 8–14.
[2] M. Maheu, P. Whitten, A. Allen, E-Health, Telehealth, and Telemedicine. A Guide to Start-up and Success, Jossey-Bass W. Wiley Company, 2001
[3] G.P. Picco, Mobile agents: an introduction, Microprocessors and Microsystems 25 (2001) 65–74.
[4] S.T. Vuong, I. Ivanov, Mobile Intelligent Agent System: Wave vs. JAVA, etaCOM '96, in: The First International Conference on Emerging Technologies and Applications in Communications, Portland, May 1996.
[5] M.J. Mendes, F.M. Silva, Mobile agents in autonomous decentralized systems, ISADS (1999) 258–260.
[6] D.B. Lange, M. Oshima, Programming and Deploying Java Mobile Agents with Aglets, Addison Wesley Longman, Inc., 1998.
[7] M. Naylor, et al., Enhancing network management using mobile agents, in: Proceedings of the 7th IEEE International Conference and Workshop on the Engineering of Computer Based Systems.
[8] M. Naylor, The Use of Mobile Agents in Network Management Applications, Master thesis, Napier University, 2000.
[9] P. Braun, C. Erfurth, W.R. Rossak, An introduction to the Tracy mobile agent system, Technical Report No. 2000/24: Friedrich Schiller University of Jena, Computer Science Department, 2000.
[10] F. Hohl, The Mobile Agent List, Available at: http://reinsburgstrasse.-dyndns.org/mal/mal.html.
[11] W. Suwu, A. Das, An agent system architecture for e-commerce, in: Proceedings of the 12th International Workshop on Database and Expert Systems Applications, 2001, pp. 715–719.

[12] A. Wang, C.F. Sörensen, E. Indal, A mobile agent architecture for heterogeneous devices, in: Proceedings of the International Conference on Wireless and Optical Communications, 2003, pp. 14–16.
[13] A. Corradi, M. Cremonini, R. Montanari, C. Stefanelli, Mobile agents integrity for electronic commerce applications, Information Systems (1999) 519–533.
[14] M.J. O'Grady, G.M.P. O'Hare, Mobile devices and intelligent agents, Information Science (2005) 335–353.
[15] C. Wang, H.F. Leung, Mobile agents for secure electronic commerce transactions with privacy protection of the customers, in: Proceeding of the IEEE International Conference on e-Technology, e-Commerce and e-Service, 2005, pp. 530–535.
[16] N. Karnik, Security in mobile agent systems, Doctoral dissertation, Department of Computer Science and Engineering, University of Minnesota, 1998.
[17] R.B. Patel, K. Garg, Providing security and robustness to mobile agents on open networks, in: Proceedings of the 6th International Conference on 182 Business Information Systems (BIS 2003), Spring, CO, June 4–6, (2003), pp. 66–74.
[18] W. Jansen, T. Karygiannis, NIST Special Publication 800-19: Mobile Agent Security, NIST, Gaithersburg, MD, 2000.

Dr. Chuan-Jun Su is currently with the Department of Industrial Engineering and Management at the Yuan Ze University, Taiwan, ROC. His interests include Information Technology, Virtual Reality, and AI applications in design and manufacturing.