Application Specific Optimization and Local Resource Availability With Mobile Agent Enabled Middleware

O.P. Akomolafe
Computer Science Department
University of Ibadan
Akomspatrick@yahoo.com

ABSTRACT
Middleware solutions for mobile computing should give application designers full location visibility to perform application-specific optimizations and to adapt to local resource availability. This will enable application developers in patient monitoring domain to be able to concentrate their resources and time on the service logic of the problem they are solving which will invariably reduce the total time. Also, efforts expended by software development firms and IT departments on patient monitoring application development will be minimal if such middleware exist. This paper produced a context-aware mobile agent middleware on top of which healthcare service programmers and developer can run other plug-in applications that allow mobile devices and biomedical sensors to be used for monitoring patients remotely using agent technology.

Keywords: Mobile agents, Middleware, Remote patient monitoring.

African Journal of Computing & ICT Reference Format:

1. INTRODUCTION
Remote patient care and telemedicine platforms have been proved during the last years to be significant tools for the optimization of patient treatment in isolated areas [10][6]. Transport, accommodation and medical personnel related costs are reduced, and a full time 24 hours per day, 7days per week patient status monitoring is provided [13]. Health monitoring may be delivered not only in a hospital environment but at home as well, through the establishment of modern patient telemonitoring systems. The recent improvement in medical devices, wireless communication and software technology offers the opportunity for more sophisticated and user-friendly digital services which can be harnessed for better remote monitoring. Mobile software agents represent one of the recently emerged paradigms related to pervasive computing that carry a tremendous prospect in solving some of the real world problems and enabling “anytime, anywhere and with any device within any context" access to digital information services. Context-awareness and context management are in the focus of intensive on-going research efforts by the pervasive computing research community.

Pervasive computing is a computing paradigm incorporated in a variety of devices (computers, cars, entertainment units, appliances, etc.), which can carry out computing in a relatively non-intrusive manner and can impact and support many aspects of work and everyday activities [1][7][15].

In addition, mobile computing can greatly benefit from the possibility of a synchronicity between user/terminal requests of operations and their execution. For instance, wireless connections impose strict constraints on available bandwidth and on communication reliability and minimize the connection time of the wireless client device. The mobile agent (MA) paradigm does not need continuous network connectivity, because connections are required only for the time needed to inject MAs from mobile terminals to the fixed network infrastructure, for instance through the current wireless point of attachment. MAs are autonomous and can carry on services even when launching users/terminals are disconnected, by delivering service results back at their reconnection. Mobile agents are software abstraction that can move from host to host on a network to perform specialized services.

Apart from providing mobility, agents possess the unique characteristics of adapting to changes in their execution environment and hence have a higher chance of survival and achieving application objectives over a large, distributed and heterogeneous network when compared against traditional techniques which make its adoption for the present research a viable option in this local context [5]. Mobile agents have also proved very effective in supporting asynchronous execution of client’s request and weak connectivity [11] and especially in developing countries.
Despite these laudable advantages mobile agents paradigms, its adoption however comes at the cost of an increase development effort on the side of application developer compared to the other technology like client server despite its advantages. There are several issues ranging from unpredictable modification in accessible resources, wide heterogeneity of access terminal to security and trust. Therefore there is a need for technologies that will bridge the gap thereby making the development of a remote patient monitoring application easier while still harnessing the intrinsic advantages of mobile agent system [15][6].

The outline of this paper is as follows: Related works are discussed given in section 2. Section 3 discussed the requirement for the middleware framework design while Section 4 gave a layered view of the middleware while Section 5 discussed a model for the middleware. Section 6 discussed the application of the mobile agent middleware with a scenario. Concluding remarks was given in section 7.

2. RELATED WORKS

[14] proposed and consequently implemented a policy based architecture that allows autonomous and continuous monitoring of patient thereby providing continuous necessary medical information to hospital personnel by utilizing software agents and wireless sensor technologies. [8] presented a telephone care system using mobile telephony for remote patient monitoring. Their system takes advantage of the serial port available in new mobile phones to implement a generic interface for patient monitors. Vital signals are acquired from electro medic devices using RS232 interface and transmitted through the internet. This work also does not adopt the mobile agent technology. A mobile agent framework for telecardiology was proposed by [4], they combined both mobile agent and object request broker mechanism in their framework so that it can support interoperability and optimize monitoring process. [12] also produced a patient vital signal measuring devices called Tyndal mole, a non-intrusive patient monitoring equipment that does localized processing. Other similar were done by [3] listed several multi agents projects and initial for e-health. Of much interest is the work of [9] called Ubimedic.

3. MOBILE AGENT MIDDLEWARE FRAMEWORK

At an abstract level the most basic functional requirement of the middleware is to serve as an effective conduit for transferring physiological and contextual data from any application sitting on it to specified server location, therefore template of contextual data must be defined completely from the abstract definition provided. The middleware promises to help simplify remote patient monitoring application development. Based on analysis of many different RPM applications, the following requirements were identified.

- **Authorisation and Authentication**: It is vital that the middleware accurately authenticate a patient or a care giver
- **Context Requirement**: The middleware should provide a high degree of context awareness by automatically inferring patients’ locations and time without the user intervention.
- **Data capturing and delivering**: In remote patient monitoring, physiological and contextual data collected through a body sensor must be transferred to medical personnel for proper diagnosis and record purpose. Different data type must be captured, both discrete and continuous data.
- **Data Transmission**: Data must be transferred to a remote central server in real time and also to subscribed care givers, strict measures must be in place to protect unauthorised access to patient data.
- **Reliability issues**: Failure of a node should be properly managed so that data integrity can be maintained.
- **Messaging**: Notifications should be generated appropriately in case of occurrences that are of interest to caregivers.
- **Portability**: Although the monitoring is targeted towards mobile devices developers can also target computers systems.
- **Integration**: The middleware should easily integrate with an application developer's code, it should use well known coding styles and conventions.
- **Performance**: The middleware should perform its operation in standard time with minimal memory footprint.

4. MODEL OF THE CONTEXT AWARE MOBILE AGENT MIDDLEWARE

The Model of the developed middleware is shown in Figure 1. It shows the internal modules of CPMF and how they interact to form a unified whole. The different components in the model are explained below:

1) **Behaviour Stack/Behaviours**: Behaviour represents a task that an agent can carry out. It extends predefined behaviour in JADE and implements an atomic task or functionality. The CPMF engine on initiator sides of the system (SPDA/Reader) selects the behaviour that implements a requested task and schedules it for onward processing by a task agent, these task agents communicate with the server base agent by sending various messages and getting responses as described by the running behaviour. On the responder side (Server) all behaviours are added to the base agent and each of them respond to incoming requests as appropriate.
2) **Base Agent**: This is an agent that is started when a session starts and runs continuously until the session stops. They perform tasks that are to be repeated and processed continuously. Most typical of such task is the patient sensor data processing (forwarding/receiving). On the SPDA, the base agent exhibit the SubmitContextValue behaviour to continuously send patient context values to the server as soon as it is passed to CPMF. On the server, the base agent exhibits all the behaviours that respond to initiators.

3) **Task Agent**: These agents are started, to perform a specific one-off tasks. On starting, they pick up the behaviour that defines the required task, execute it and are destroyed when the task completes. Task agents exist only on the initiator side of CPMF, they send requests to the base agent on the server as specified by the behaviour assigned to them and are destroyed when a response is received, the response is handed over to the CPMF engine for onward delivery to the third party application.

4) **Agent Content Manager**: Handles the wrapping and unwrapping of agent communication concepts (Communicative acts, performatives, actions, expressions etc.) into or from a string. It converts between internal objects and FIPA compliant agent message content string. It serves as an interface between the CPMF Engine and its associated entities and the underlying agent-based framework.

5) **CPMF Codec**: This is the class that defines the language used by messages sent by CPMF. It understands and translate internal objects into/from

---

**Figure 1 Model of the Context Aware Mobile Agent Middleware**

---

79
string tokens as requested by the Agent Content Manager.

6) **CPMF Ontology**: This class defines the ontology used by CPMF. It understands the different terms and concepts in the framework and converts them to a standard format understood by the CPMF Codec. Uses defined schemas to interpret the symbols and expressions in agent messages sent in the framework.

**Schema**: Schemas provide definition of how internal objects will be converted to or reconstructed from the generic representation. Schemas are defined for concepts, predicates and agent actions to define how they will render themselves to the internal agent message representation.

Typical schemas defined in CPMF include Concept Schemas e.g. PatientSchema, AuthorizationTokenSchema, contextValueSchema, RelativeSchema etc. Predicate Schemas e.g. :HasStringIdOfSchema, ContextValueIsSchema, OwnerStringIds, ContextIsSchema etc. Agent Action Schemas e.g. PatientLoginAgentActionSchema, RegisterContextAgentActionSchema

7) **Concepts**: These model the entity classes, they provide a wrapper around them to make them conform to the JADE concept system. Some entities in CPMF are: Patient, Context, Address, ContextValue etc.

8) **Predicates**: These are used to indicate that a specific condition is true. They often wrap concepts. An example is indicating that a Patient Concept that has an Id of “id” is something. Some examples include: ContactWithIdOf, ContextIs, ContextValueIs, HasStringIdOf

9) **Agent Actions**: These model requests that perform specific actions. Example is Login to server, object creation on server, e.t.c typical agent actions in CPMF are: PatientLoginAgentAction, RegisterContextAgentAction

10) **CPMF Vocabulary**: It is an interface that serves as a central dictionary for Concept, schema and agent action names. Names used in the CPMF Ontology are declared as constants in the interface and reference all over CPMF.

5. **LAYERED VIEW OF THE MIDDLEWARE IN A REMOTE MONITORING SYSTEM**

Figure 2 shows the layered diagram for the middleware on the source personal digital assistance (SPDA) and the server which communicate back to back from the SPDA using the Context aware Middleware Framework Engine CPMF Engine (SPDA) /JADELEAP to the central server using the CPMFEngine(Server)/JADE. Some of the components in the diagram are described below.

11) **JADE-LEAP** is the java API used to develop the agent platform. JADE-LEAP is a modified version of the JADE platform that can run not only on PCs and servers, but also on resource constrained devices such as mobile phones provided that they are Java enabled (MIDP 1.0 is the minimal requirement). It is important to note that JADE-LEAP provides the same APIs with respect to JADE so that an agent developed to run on JADE can run on JADE-LEAP with NO modifications at all. The JADE-LEAP api runs on the java virtual machin instatled on the mobile phone.
Figure 2 shows the layered diagram for the middleware on the source personal digital assistance (SPDA) and the server which communicate back to back from the SPDA using the Context aware Middleware Framework Engine (CPMF Engine) /JADELEAP to the central server using the CPMFEngine(Server)/JADE. Some of the components in the diagram are described below. JADE-LEAP is the java API used to develop the agent platform. JADE-LEAP is a modified version of the JADE platform that can run not only on PCs and servers, but also on resource constrained devices such as mobile phones provided that they are Java enabled (MIDP 1.0 is the minimal requirement).

It is important to note that JADE-LEAP provides the same APIs with respect to JADE so that an agent developed to run on JADE can run on JADE-LEAP with NO modifications at all. The JADE-LEAP api runs on the java virtual machine installed on the mobile phone. As shown in Figures 2 both Jade- Leap and Jade runs on the JVM Layer which in converts Java byte code into machine language which is understood by the mobile device and executes it which in turn runs on the host Operating system. Entity Classes are classes that model the entities in the problem space, they model the actors, actions and data that move around the system.
Entity class model actors such as patient, medical personnel, relative etc. actions such as policy actions, and data such as place, address, context, context value etc. The classes have a consistent definition across all modules of the framework. The application developer uses it to define scenarios, requested actions and processes for a Remote Patient Monitoring (RPM) application using the middleware, that is Context aware Patient Monitoring Framework (CPMF) Engine. It makes API calls and it is called back with responses as appropriate.

Two basic agents that are doing all the plumbing work inside CPMF are Task agent and Base agent. A complete process of sending data is put into a session initiated by Task Agent, using a transport service that is built on top of Jade Message Transfer Service (MTS). Base agent to consistently transport contextual values from the source PDA to the server after a session has been successfully initialized by a task agent. As we much data across PC and mobile devices data are persisted using JPA Data Persistence on server and RMS Data Persistence on J2ME devices. Persistence of some information is needed on the Reader devices(mobiles devices) so that continuous reading won’t required continuous login procedure.

Notification services facilitates easier exposure automatic SMS or email sending capabilities to subscribers that wish receive patient sensor data while replicator agents on alternate server does automatic synchronization of data.

6. APPLICATION OF THE MIDDLEWARE

A simple scenario was implemented using the developed framework. Figure 3 shows the activity diagram for the patient registration process on the source PDA. A sample medical patient to be monitored is registered first on the server front end which mimics the process of registering a patient by for example, a health insurance firm. On successful registration, the patient is given a unique id that will be used for future correspondence. Next, on the PDA that will be used for receiving sensor data, an application that uses the middleware is installed. The application on startup, checks if a patient has been previously registered on the phone. If no, the id of the patient is requested and the full information for the patient is retrieved from the central server. After confirmation of the retrieved information, the Contexts registered for the patient is retrieved and persisted. After successful execution of these processes, the patient and his registered contexts are persisted on the phone and monitoring can commence. A reader, who could be the personal medical personnel or a next of kin, also installs an application that was developed based on the middleware, supplies the id of the patient he wishes to get reading from, the patient info is retrieved, he confirms and can then get live readings from the patient.
Figure 3  Activity Diagram for Test Scenario
7. CONCLUSION

We presented a framework that allows for easy development of healthcare monitoring applications. Using this framework, we successfully implemented a middleware platform for healthcare monitoring that allows application developers to develop remote monitoring applications embellish advantages mobile agent. Our example applications in activity monitoring and rehabilitation show how body sensor network resources can be dynamically allocated based on application requirements.

REFERENCES


AUTHOR’S SHORT BIOGRAPHY

Dr. Akomolafe Oladeji Patrick is a Lecturer at the Department of Computer Science, University of Ibadan. He obtained a Bachelor of Technology (B.Tech) in Computer Engineering at Ladoke Akintola University of Technology, Ogbomosu(LAUTECH) in 1999, a Master of Science Degree (M.Sc) in Computer Science at the University of Ibadan in 2004 and a Ph.D Degree in Computer Science from Ladoke Akintola University of Technology, Ogbomosu, Nigeria in 2014. His research interests include Pervasive and Mobile Computing, Mobile Agent Technology, Context Aware Computing and Software Engineering. He can be reached at Akomspatrick@yahoo.com