Proceedings of the 4th WSEAS Int. Conf. on Information Security, Communications and Computers, Tenerife, Spain, December 16-18, 2005 (pp21-24)

Implementation of Ubiquitous Digital Systems and Services*

Kyo-Sun Song, Gyeo-Re Park, Yu-Raak Choi, SeSangPyoungHwa Lee, Min-Young Kim, Yun-Sam Kim, Eun-Sun Cho
School of Electrical & Computer Engineering
Chungbuk University
12 Gaeshin-dong, Heungduk-gu, Chungbuk
Korea

Abstract: - In this paper, we propose a simple middleware for ubiquitous computing systems. Developed with UPnP in Java, it efficiently organizes runtime interactions of devices. To show the feasibility of this system, we implemented a set of customized services based on this middleware. This prototype allows users to enjoy services at various spatial points even while moving around.

Key-Words: - Ubiquitous, Java, UPnP, RMI

1 Introduction

'Ubiquitous computing' is computing environment that everything is connected for user service and get intellectual. Moreover computer should be provided to be invisible in physical space and service depending on user's condition. Although there have been many related research, few case which build and experiment this environment have been held in Korea.

In this paper, a simple middleware for ubiquitous computing system will be proposed and an example using this will be introduced. Proposed ubiquitous middleware was composed through Java and UPnP to service with connecting devices which located apart in different place for given user. And using this, '1 Inch' system which provide that when user move to another place, he or she can be served continuously services such as a favorite movie or music which was provided in previous space. Software robot and display devices which are available for user's convenience were installed in each space and the moving of user was sensed by cameras and RFID.

2 Middleware

2.1 Implementation Model

The basic implementation model of middleware is that when event occur by all devices is forwarded to the EventManager by the Event Deliver, proper action to that event will take place through Ctrl. Fig. 1 explains this structure.

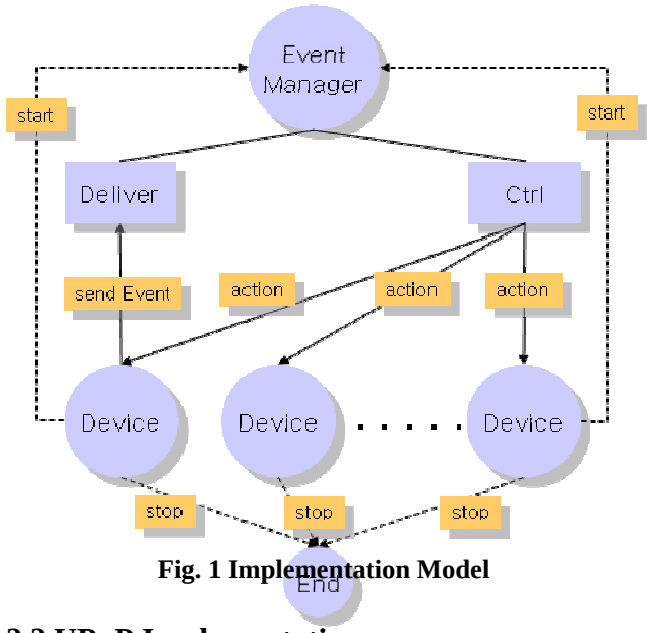

Fig. 1 Implementation Model

2.2 UPnP Implementation

UPnP is the abbreviation of Universal Plug and Play.

* This research is supported by the ETRI, Korea.

It can control or view the status of devices in local network. And it is the concept extended to network of plug and play that if connected, it will be recognized. It uses CP (Control Point) to control devices.

UPnP protocol is based on many standards. (GENA, SSDP, SOAP, HTTPU, HTTP) Hence we need to understand and implement these protocols to make UPnP devices. CyberLink provides Java Package for UPnP development [1]. If we use UPnP to implement above model, there would be a few weak points.

First, although EventManager and devices should communicate bidirectional, in UPnP, it is impossible. The reason is that control from CP to devices is available, but control from devices to CP is unavailable. Because a device provide action and an attribute description to CP and CP controls a device according to provide an attribute description. For this reason, we need another way to pass the event which occurred in a device to EventManager and there are two solutions.

2.2.1 Checking method

Devices have an event attribute which remember the events occurred and CP checks each devices' event attribute periodically, if there is event to run, it will do appropriate action and then initialize Event Attribute.

But UPnP has much delay relatively caused by network communication. Therefore if new event happens before CP check event Attribute, previous event will be able to disappear.

2.2.2 Using CP

We provide CP controlled by device for sending event to EventManger. The CP called Deliver has a 'sendEvent' method. When devices generate an event, it sends the event to EventManger through Deliver's sendEvent method. This way doesn't occur above problem. But the works of CP- find a device, manage a device list- are overlapping and a device needs occasionally more than one CP. If the device has more than one CP, its CPs doesn't correctly work at all time because their basic data duplicates.

2.3 RMI Implementation

The Java Remote Method Invocation (RMI) system allows an object running in one Java Virtual Machine (VM) to invoke methods on an object running in another Java VM. RMI provides for remote communication between programs written in the Java programming language [2]. before starting RMI server, we need to start RMI's registry, using the rmiregistry command. If you change RMI's object, you've to recompile and distribute the file to clients.

We can improve a performance and a function more than UPnP. But pre-setting annoy us before starting RMI .

We developed both ways in the beginning. UPnP's performence is not enough to hope in test. But it has some merits. Despite it allows standardization and connection control, we choose the RMI platform for good performance at last.

3 Service Implementation

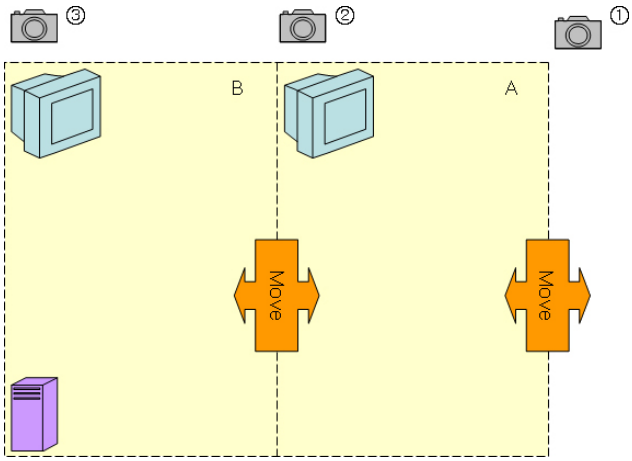

Fig. 2 '1inch' service room

'1 Inch' service consists of Fig. 2 with purposed middleware. When nobody stays in the room, ① web cam captures visitors and saves the picture. ② and ③ web cam checks where person is. The monitor installed in each room is displayed 1Inch service screen. When the person sees a movie or listens to the music by 1Inch service in the A's room, if he moves to B's room, he can be served continuously services.

In this section we explain how to recognize a movement and 1 Inch service.

Fig. 3 move B's Room to A's Room

3.1 Motion Detector

② and ③ web cam can capture enough pictures to check a room state. It is important that dectecting a difference between previous picture and present picture in real-time. The picture consists of a variaty of pixels. We convert them into a binary data to process efficiently.

3.1.1 Binary Conversion Algorithm

A pixel of an image has four information. It is an index, red value, green value and blue value. The index is a location of pixel and red, green and blue is the three primary colors. Those pixels make one of a picture in the know. Each pixel has a unique value so that we can reorganize the image. Binary Conversion reads the values in order and makes a histogram of them. And then we calculate an average of them. So we get a binary data accoding to the average.

3.1.2 Image comparison

A web cam creates 320x240 resolution images. There are 76800 pixels and each pixel has only 0 and 1 value through binary conversion. The binary value depends on whether the original value is greater or less then the average. We compare each pixel and express the result numerically. The numerical value of result is normally 100~9,000. When there is a movement, it increases more than 15,000 so that we can recognize there are something moving.

Fig. 4 '1 Inch' service screen (GuestBook, Movie)

3.2 '1 Inch' Service

'1 Inch' service is displayed full-screen and serviced basically four services - movie, music, weather information and guestbook - with character. The resource for service - movie clips, music clips and pictures - is shared in the network.

There are two ways to play movie and music in Java. First, it is low-level programming to use java.io. It is powerful but not convenience. Nowadays, Java supports a multimedia programming by Java Multimedia Framework (JMF) [3]. It is convenience to use and enough function to control. Movie and music service are played by JMF. The screen display slideshow in music service.

Weather information service is a weather forecast. We get the data provided by kweather site. It is a simple text file.

Guestbook service can check who invited here while user goes out of the room.

4 Related Work

4.1 Middleware

Project Aura is a ubiquitous architecture in Carnegie Institution [4]. The Aura span is every system level: from the hardware, through the operating system, to applications and end users. The example application of Aura Architecture is Portable Help Desk (PHD). PHD is a context-aware application built on two fundamental services: spatial awareness and temporal awareness. Spatial awareness includes a user's relative and absolute position and orientation. Temporal awareness includes the scheduled time of public and private events. Aura is specifically intended for pervasive computing environments involving wireless communication, wearable or handheld computers, and smart spaces. So Aura concentrates upon a problem such as intermittent and variable-bandwidth connectivity, concern for battery life, and the client resource constraints that weight and size considerations impose.

Unlike Aura, PICO's main components are software entities, called delegents (intelligent delegates), and hardware entities, called camileuns (connected, adaptive, mobile, intelligent, learned, efficient, ubiquitous nodes) [5]. Its novel features include creating mission-oriented dynamic communities of software agents, just-in-time communication and proactive collaboration among these communities, and adapting hardware and software changes as well as application requirements.

We research how to manage efficiently a variety of context information like Aura architecture, unlike software smart agent, PICO.

4.2 Service

MIT's smart room acts like invisible butlers [6]. And they use cameras, microphones, and other sensors to try to interpret what people are doing in order to help them.

The example of GAIA system is an active space [7]. It is a physical space coordinated by a responsive context-based software infrastructure that enhances mobile users' ability to interact with and configure their physical and digital environments seamlessly.

Smart Room, Active Space and 1 Inch Service Room is a similar service to interact with user in physical space. But both systems consider an interaction in one service space. 1 Inch service considers an interaction between service spaces.

5 Conclusion

In this paper, a simple middleware for ubiquitous computing system proposed and an example using this introduced. Proposed ubiquitous middleware was composed through Java and UPnP to service with connecting devices which located apart in different place for given user. And using this, 1Inch service system which provide that when user move to another place, he or she can be served continuously services such as a favorite movie or music which was provided in previous space. We will improve the middleware to control a variety of events and process. So the service will be changed as VOD, MOD streaming service (Not sharing the resource in the network.).

References:

[1] CyberLink for Java, http://www.cybergarage.org/net/upnp/java/

[2] Java Remote Method Invocation (Java RMI), http://java.sun.com/products/jdk/rmi/index.jsp

[3] Java Media Framework API, http://java.sun.com/products/java-media/jmf/index.jsp

[4] David Garlan, Daniel P. Siewiorek,Asim Smailagic, and Peter Steenkiste Aura: Toward Distraction-Free Pervasive Computing

[5] Kumar, M. Shirazi, B.A. Das, S.K. Sung, B.Y. Levine, D. Singhal, M PICO: A Middleware Framework for Pervasive Copumting

[6] Alex P. Pentland, Head of Perceptual Computing Section at the Media Lab of MIT Smart Room 1996

[7] M. Roman, C.Hess, R. Cerqueira, A. Ranganathan, RH Campbell, K. Nahrstedt. Gaia: A Middleware Infrastructure for Active Spaces. IEEE Pervasive Computing Vol. 1, No. 4, p. 74-83. October - December, 2002