

UFC: A Ubiquitous Fashionable Computer

Kyu Ho Park , Seung Ho Lim, , Yong Song, Daeyeon Park
UFC Project Group*
Korea Advanced Institute of Science and Technology

Abstract-- The ubiquitous fashionable computer(UFC) is a human centered fashionable computer with which we exploit the ubiquitous computing environment by our intuitive interface such as our hand motions. The appearance goal of UFC is to design it aesthetically so that we can wear it like our fashionable accessories or clothes. The main features of our interoperable wearable computer, UFC, can be summarized as follows: (1) the interoperability in CDMA, WLAN, BlueTooth and ZigBee communication environments in which multimodal communication is implemented, (2) the ubiquitous computing services based on our original middleware, (3) a new human computer interfaces called *i-Throw* by which file transfers are possible between two persons just taking a ball throwing motion by one's hand and (4) an aesthetic and ergonomic appearance design for the wearability. Besides these features, we have also implemented a testbed using sensor boards which support ZigBee protocol. The ZigBee protocol stacks are ported for the testbed. A voice recognizer is implemented for UFC and 100 voice commands are possible with 98% accuracy. VoIP phone is also developed for UFC with image transfer.

It is the first year of total 3 year projects started last March, 2005. The main functions of the prototype are described in this paper but its quantitative evaluation is now under progress. The performance improvement is our further work for the remaining two years

Keywords – Fashionable Computer, Ubiquitous Fashionable Computer, UFC, intuitive interface, information throwing, ZigBee.

I. Introduction

In the ubiquitous computing environment, we can get necessary information anywhere and anytime without any constraints. This new computing paradigm influences almost all areas of our society. To exploit the ubiquitous environment efficiently, an interoperable wearable ubiquitous computer is required. Furthermore, It is generally thought that the value of a wearable computer is largely determined by how easily and effectively it allows target users to perform the tasks in everyday life. The success of UFC will heavily rely on good wearability, usability, aesthetic appearance, and social acceptance. We defined the target users as young university students and drew design concepts by analyzing their activities in everyday life and fashion trend.

We have been developing the UFC since last March, 2005. Its CPU is ARM9 and it runs on embedded Linux. It supports CDMA, WLAN, BlueTooth and ZeeBee communication and the multimodal communication is also possible for the interoperability. For the wearability, it is designed to have the low power consumption, and small size. It has very convenient features such as the voice command system, multimodal communications and service recovery through our own middleware that will be explained later in this paper.

The most important contribution of UFC can be summarized as follows;

1. A ubiquitous fashionable computer is implemented based on ARM9 processor and embedded LINUX.
2. The Multimodal communication with CDMA, WLAN, BlueTooth, and ZigBee communication modules.
3. A new human computer interface called *i-Throw* is invented to throw information in one's UFC to the other UFC(s) or Objects when a throwing action is taken by his hands. Using the *i-Throw*, we can turn-on/off devices, volume-up/down audio devices, and scroll-up/down pages of computer screens remotely.
4. The Middleware is developed to support the service discovery and resources discovery.
5. An original Java virtual machine for embedded computers, TVM, is developed for the UFC.
6. A testbed implemented using ZigBee sensor nodes and stacks for the location based service(LBS).
7. To reduce the execution latency of necessary programs, a look-ahead-execution(LAE) of required programs in each location is developed and the seamless I/O for the UFC and the testbed is also provided.

The whole architecture of UFC is described in Fig.1 .

*UFC Project Group: It consists of 10 research teams including Kyung Ki University, Ajou University, Won Kwang University, Tmax Software Co., At Phone Co., and KAIST. This project is supported by the Ministry of Information and Communication, Korea.

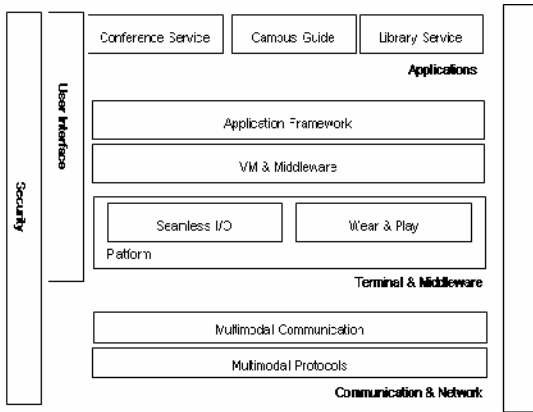


Fig.1 Architecture of *UFC*

The *UFC* consists of communication and network, terminal and middleware, security and application parts. At the same time we are developing its testbed which supports ubiquitous computing environment. In communication and network parts, the major theme is to implement multimodal communication method among CDMA, wireless LAN, BlueTooth, and ZigBee RF communication. The BlueTooth is used for wireless microphone, wireless earphone, wireless mouse and wireless keyboard. The ZigBee is adopted for the location based service which is implemented in testbed of the *UFC*. In the terminal and middleware part, a wearable terminal with low power consumption, seamless I/O, wear and play characteristics is now being developed. The middleware is required to support each *UFC* to be able to discover other resources and communicate with them. Middleware APIs are defined and implemented for this purpose.

Several applications are implemented to demonstrate the functions of *UFC* and its testbed. A new user interface called 'Intuitive Interface' is implemented to send and receive information in the mode of human gesture like a pitcher and a catcher in a baseball game. If a user with a *UFC* take an action to pitch a ball with his arm, some information will be transmitted through wLAN in the testbed to a user who is located at the position where the user had taken a throwing action directed. For this interface, we implemented a small device which can detect the throwing action and its direction.

This paper is described in the following order. In section II, the details of *UFC* platform including the appearance design are explained. The user interfaces including the voice recognizer and *i-Throw* interface are described in the next section. The testbed implementation, and the middleware design are presented in section IV and V respectively. It is concluded in section VI.

II. The *UFC* Platform

A. Platform Architecture

The basic design concept of our *UFC* is the modularity and easy extensibility. It consists of 5 modules; CPU, Communication, Harddisk, Communication, Battery, and I/O modules. They are easily extensible by USB ports. The implemented hardware is shown in Fig.2.



Fig.2 Implemented H/W of *UFC*

The hardware specification is given in Table 1.

Table 1. Hardware Specification

CPU	ARM-9 624Mhz
Memory	256 DRAM
Communication	WLAN, BlueTooth, ZigBee, CDMA
Video I/O	VGA/SVGA
I/O Devices	HMD, Earphone, Microphone, Keyboard, Mouse
I/O Module Interface	USB, RS232C

The platform software is implemented as shown in Fig.3.

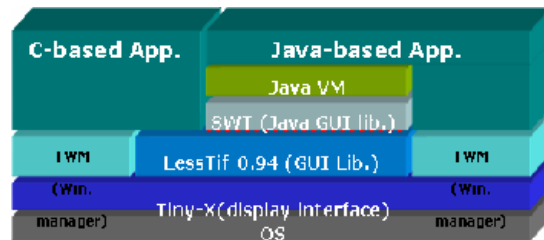


Fig. 3 Platform Software

B. Appearance design of *UFC*

It is generally thought that the value of a wearable computer is largely determined by how easily and effectively it allows target users to perform the tasks in everyday life. The success of *UFC* will heavily rely on good wearability, usability, aesthetic appearance, and social acceptance. We defined the target users as young university students and drew design concepts by analyzing their activities in everyday life and fashion trend. The process of designing *UFC* is an iterative one, in which we tried to find the best solution to meet detailed design requirements and to fulfill perceived needs of the target users through repetitive "prototyping-bodystorming"

sessions. *UFC* should be a attractive fashion as well as a high-functional computing device.

In address this problem, we have been trying to make a new design form factor for *UFC* that combines the characteristics of product and fashion.

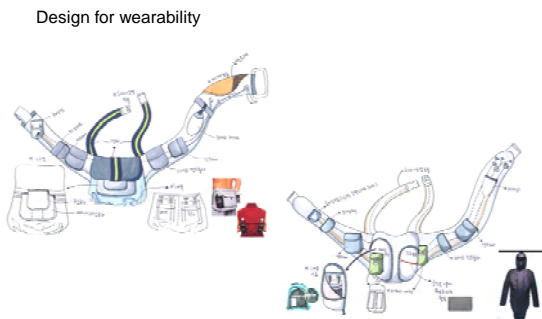


Fig. 4.1 Design for Wearability

Final study model

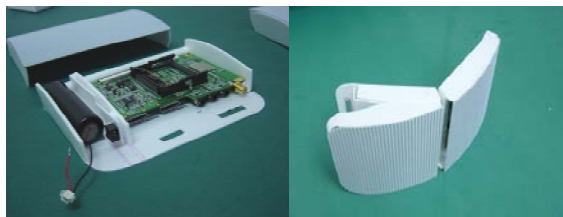


Fig. 4.2 Appearance of *UFC*

III. User Interfaces

Besides a head-mount display(HMD) unit, head-phone set , microphone, keyboard, and mouse, we have developed a voice recognizer for voice command and *i-Throw* device for an intuitive interface like human.

A. Voice Recognizer

A speech recognizer for *UFC* needs two or three microphones as its input device. Observing the waveforms which are received at each time, speech detector always checks whether they are potential speeches or not. When an utterance is detected, feature extractor extracts high-dimensional features such as cepstral coefficients. Finally, we get to obtain a recognized word through calculating the probability of input waveform.

The block diagram of speech recognizer is shown in the following figure.

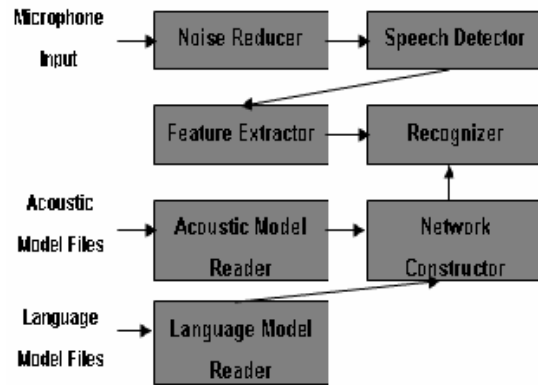


Fig.5 Voice Recognition Method

The recognized speeches consist of a string of characters, which is an encoded string. This string of characters is transported to the pseudo console driver in the kernel, which is responsible for transmitting the input string to keyboard input strokes. Those key input strokes will be considered as if someone strokes keyboards, so the recognized speech can be efficiently transmitted to each application requiring input information

B. *i-Throw*: Information Throwing based on 3D input device

We have implemented an interesting interface called *i-Throw* based on an input device consisted of a 3D direction sensor and an accelerometer. The basic idea is that we can send information like a ball throwing. For example there are two person several meters apart, and one person want to send a ball to the other he will throw the ball to the other person. Some years ago, MIT researchers showed that when two person shake hands , each persons' information at each person's PDA is exchanged using each human's skin as communication media. But in our *i-Throw*, information can be thrown to human or objects like display devices, or other computers. To implement the *i-Throw*, an 3D magnetic sensor , HMC10523[7] , and a 3D low-g accelerometer, MMA7260Q [8]are used as in Fig. 6.



Fig. 6 *i-Throw* Device

Besides the throwing interfaces, we also implemented a hand-right-rotation, hand-left-rotation, hand-scroll-up, hand-scroll-down and hand-erase gestures. Using the *i-Throw*, we also have implemented the device on-off, volume up-down, and page up-down of Power-point display. The current hand motions of *i-Throw* is summarized in the Table 1.

Table 2 Hand Motions and Services

Hand Motions	Command Description
Throwing a ball	File Transfer
Turning left/right a dial	Intensity Control
Waving a hand	Scroll Up/Down
Wiping a blackboard	Erasing

IV. Testbed Implementation

The interoperability of wearable computers is an important issue especially in the ubiquitous environment. The wireless sensor network based on IEEE 802.15.4[3] is one of the possible technologies to provide the location sensing and mobility to mobile users with wearable computers.

In this paper, we propose the architecture of the testbed as shown in Fig. 7. We assume the campus-wide testbed which includes both the street and indoor room. The sensor nodes are installed in a regular manner for the location sensing of the mobile users. For the management of the testbed, three different servers are installed: location management, 6lowpan network management, and application management. The connection to the servers could be done via multi-hop sensor nodes and wireless LAN.

If the connection requires high speed, the wireless LAN is used, while sensor network is for the low-power connection.

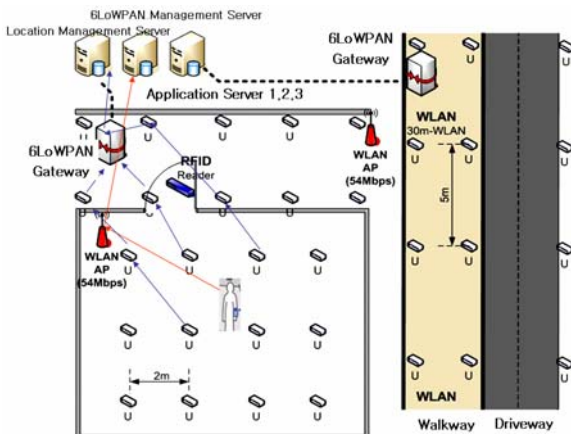


Fig. 7 Interoperable Testbed for UFS



Fig. 8 Testbed sensor node

Fig. 8 shows the sensor node with IEEE 802.15.4. The sensor node broadcasts beacons periodically for location sensing of mobile users. It uses 2.4GHz as a physical channel. The sensing of the location of mobile users utilizes the RSSI (received signal strength indicator) value of IEEE 802.15.4 networks. When a mobile user will receive multiple beacons from the sensor nodes around the user, it computes the RSSI values of them and then calculates the location.

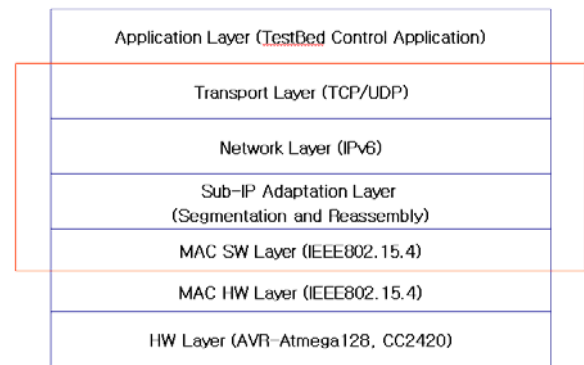


Fig. 9 Network stack of the sensor node

Fig. 9 shows the network architecture of the sensor node with 6lowpan. The MAC layer is the IEEE 802.15.4. For the network layer, we currently develop both the IPv6[6] and ZigBee network. Using IPv6 can offer the internet connectivity naturally, while ZigBee is a mature network technology relatively compared to the 6lowpan of IETF.

V. Middleware

We expect that any users wearing our *UFC* can communicate with the ubiquitous computing environment¹ and can be provided with various helpful services, like location-based services, wherever the ubiquitous infrastructure exists. For supporting this near-future scenario, we design an efficient middleware platform and implement it with adding essential functions. The *UFC* should perform services with low energy as possible as it can. Hence, especially on the *UFC*-side

¹ In this project, a room server takes a role of the ubiquitous infrastructure in the scope of a room.

middleware platform, it had better need minimal functions for exploiting the same ubiquitous services

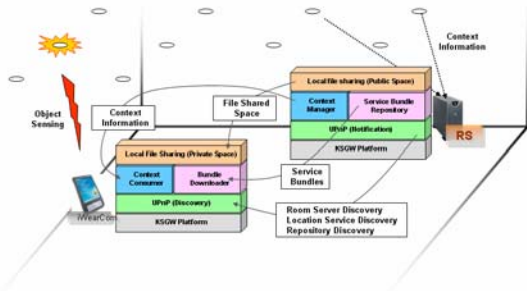


Fig.10 Middleware Functions

Our middleware implementation is divided into some components as follows. KSGW(Kaist Service Gateway) Platform is a core component and application framework that provides the execution environment with various applications and helps them interoperate with each of them. Service discovery mechanism following the UPnP specification helps *UFC* discover new ubiquitous services of new places. Various applications can be more intelligent with context information provided by the context manager. When a user with *UFC* comes to new place, a space-specific application (service bundle) of the place is downloaded from the bundle repository of a room server to the *UFC* by the bundle downloader. Additionally, we implement local file sharing application on our middleware platform through which users in the same place can share their own files with others. The functions are shown in Fig. 10.

A. KSGW (Middleware core, Application Framework)

As shown in the next Fig.11, the KSGW platform is based on OSGi specification[1] and it will operate on TVM which is developed by Tmaxsoft. On a KSGW platform, various applications (or bundle) can share their codes(classes) with others and their own service can also be referenced by others. This platform provides application framework for ubiquitous application developers. Moreover, on this architecture, user can compose optimal applications on his *UFC*. The essential components, Context management and Service Discovery is also installed on the platform and provides its own functions which other application can exploit. Especially, our toy application, local file sharing application, exploits user’s location information from context manager. It is the important thing that based on our middleware platform we can compose the *UFC*-side system with the minimum operation for exploiting the services provided by the infrastructure and we can also compose the infrastructure system providing rich services.

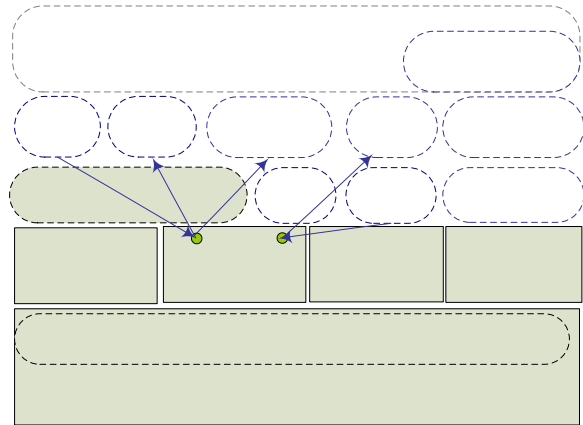


Fig.11 Middleware Core, Application Framework

B. UbiSpace

Many applications communicate each other by user defined java object or XML documents in distributed computing. We built object sharing space – UbiSpace – to exploit easy development and lightweight communication. As the applications on our framework built on KSGW platform, user defined java object can be used as a message without parsing or transformation to other native object.

Socket programming or other network programming models are built on client/server model. The sharing objects are taken by given channel name to take off user involved server/client socket programming. The sharing applications make a channel to maintain sharing space. UbiSpace contains each object as a tuple which can be queried or inserted as publish/subscribe model.

C. Context Manager

We developed context manager which handles context information. Context manager has a space allocated to each user. User specific context information is kept separate private space in UbiSpace repository. Context producer generate context information and publish the information to the UbiSpace. Room server or location based system infrastructures use this API to generate context information such as user location, proximity to the service area. Context consumer takes context information by pull mode or subscribe mode.

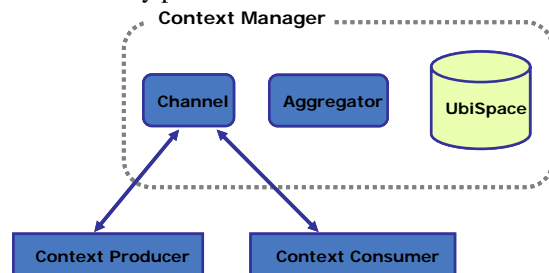


Fig.12 Context Manager

The scenario of the context manager is built on building rooms. A room server-context producer manage user information and publish the event of room entrance/exit or service/device in out to the room, which event is generated from sensor network.

The user can set the handler of the event of context information. When a user receives the event, the use can refresh the user list or current location by the event handler.

D. Service Discovery (UPnP)

As mentioned before, *UFC*'s service discovery is implemented with UPnP[2], standard, to satisfy these fundamental properties. UPnP find device and service, set up and make it ready automatically and user don't know about these process. Because UPnP is originated from home network service discovery, we expect that *UFC* can interact not only home network device easily but also simple server lookup using the UPnP protocol.

E. Local File Sharing Service (LFSS)

We design a toy application on our middleware platform, which exploit the ubiquitous services. The application called LFSS is a service for sharing files among multiple *UFC* in the same location. The service provides the ability for *UFC* to host files that is shared with other *UFC* and to locate and retrieve files from others. Additionally, the service authenticates *UFC* with location information provided by Context Manager. From the location information, the only system can get the list of every *UFC* in the same room. Hence, the outdoor *UFC* can't use LFSS.

LFSS operation is presented in below figure. LFSS uses a centralized peer-to-peer file sharing mechanism. That is, *UFC* in a room sends its own file information list to share with a room server and receives other file information lists from the server. Using these file information lists, *UFC* in the room can share files mutually.

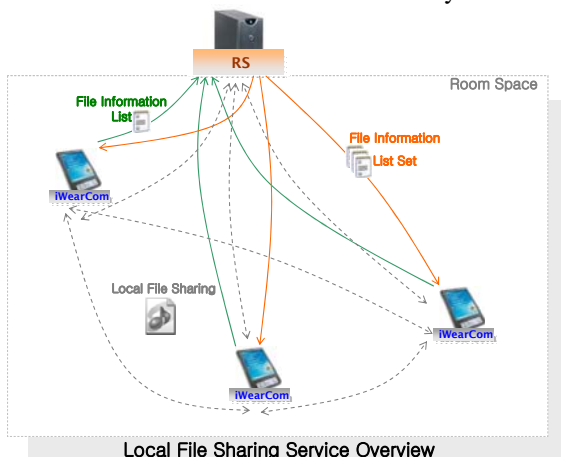


Fig. 13 Local File Sharing Overview

VI. Conclusion

In this paper, we have described the main features of our ubiquitous fashionable computer *UFC* that is an evolved form of wearable computers with the aesthetic design and the intuitive interface. The first prototype of the hardware and software of *UFC* have developed and will be improved progressively by its evaluation. The testbed has been also implemented with ZigBee based sensor network which supports the location based service and its middleware is also developed for the service recovery in the testbed. Our testbed will be expanded to the campus-wide area and *UFC* will be a real companion to each students for their convenient campus lives .

References

- [1] OSGi Service Platform Release 3 (<http://www.osgi.org/>), March 2003, IOS Press
- [2] UPnP™ Device Architecture 1.0 (http://www.upnp.org/download/UPnPDA10_20000613.htm), December 2003, UPnP™ Forum
- [3] IEEE Standard 802, part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs), 2003.
- [4] <http://www.figure8wireless.com/Zstack.html>
- [5] http://www.atmel.com/dyn/products/product_card.asp?part_id=2018
- [6] S. Deering and R. Hinden, "Internet protocol, version 6 (IPv6) specification," IETF Internet draft RFC 1883, Dec. 1995.
- [7] HMC1053, Honeywell, 3-axis magnetoresistive sensor <http://www.ssec.honeywell.com/magnetic/datasheets/HMC105X.pdf>
- [8] MMA7260Q, Freescale, 3-axis accelerometer http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MMA7260Q&nodeId=01126911184209