Thin Client Based User Terminal Architecture for Ubiquitous Computing Environment

Tatsuo Takahashi[1], Satoshi Tanaka[1], Kenichi Yamazaki[1], and Tadanori Mizuno[2]

[1] Network Laboratories, NTT DoCoMo, Inc., 3-5 Hikari-no-oka, Yokosuka 239-8536, Japan
{tatsuo, satoshi, yamazaki}@netlab.nttdocomo.co.jp
[2] Faculty of Information, Shizuoka University, 3-5-1 Jo-hoku, Hamamatsu 432-8011, Japan
mizuno@cs.inf.shizuoka.ac.jp

Abstract. In this paper, the authors examine the use of thin client based user terminals to realize the RFID tag based ubiquitous computing environment. The ubiquitous service targeted is not information retrieval via RFID but the user observation service based on environment perception. Thus the user terminal must ensure service consistency even when the communication link to the server is disconnected. In order to achieve this, the authors propose an event cache mechanism that stores predicted event conditions and the corresponding reactions. A prototype and evaluation results are also described.

1 Introduction

In the ubiquitous computing environment advocated by Marc Weiser [1], various objects surrounding us will have computing ability. They will recognize the situation of the user and his/her surroundings, select the best service, and offer it without error or intrusion. In such an environment, the user will not have to know how to use a computer nor recognize the existence of the system. Therefore, the conventional wisdom is that user-carried terminals such as note PCs and PDAs will lose their significance in the ideal ubiquitous computing environment. However, the authors consider that the user carried terminals (hereafter called user terminal) will be needed for the time being, in order to evaluate and collect the ubiquitous services provided, offer an exclusive use actuator (which prevents interference by a third party), and follow the moving user to collect his/her situation continuously.

From such a standpoint, the authors examine the architecture of the user terminal and computing device controlling it with regard to implementing the ubiquitous computing environment.

The authors propose a thin client based user terminal that is realized as a cellular phone with an RFID tag reader. No specific RFID format is assumed.

2 The User Terminal for the Ubiquitous Computing Environment

2.1 Assumed Ubiquitous Computing Environment

Figure 1 illustrates the ubiquitous computing environment assumed in this paper. In this environment, computing devices in cyber-space capture the real-space from RFID

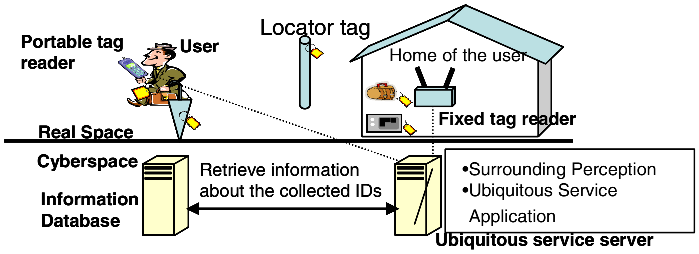

Fig. 1. Assumed ubiquitous computing environment

tags as detected by tag readers. RFID tags are attached to various objects. Locator tags are special RFID tags that identify places not objects. Locator tags will be attached to train stations, major buildings, and intersections. Each RFID tag stores just a unique identifier. Tag readers collect the IDs of the RFID tags in their range, and forward this information to the computing devices (hereafter called server) via the Internet and/or mobile communication networks. The server passes the ID to an information database, and retrieves information (name, role, color, owner, and other attributes) of the object. The server infers the user's surroundings from the collected information and offers various ubiquitous services as appropriate.

Though the simple information search service based on RFID tags is also called ubiquitous service, the authors address the observation and life assistance services based on RFID-based environment perception. Also, in this paper, the authors focus on services for the mobile user. Examples are Lost Property Notification service[2] which gives real-time notification when user drops a carried object or it seems to have been stolen, and Shopping List service [2] which is a reminder service for when the user seems to have forgotten to purchase something (or goes to the shop).

2.2 User Terminal Requirements for the Assumed Environment

In the ubiquitous computing environment assumed, each user should carry his/her user terminal in order to satisfy the following requirements.

(1) Platform for the portable tag reader

Because the detection range of most tag readers is limited to about ten meters even if the UHF band is used, it is difficult to cover all areas completely by using fixed tag readers. If the server cannot detect the event that should trigger the service, user satisfaction will be degraded. Thus, in the assumed environment, each mobile user always carries a portable tag reader to provide real-time the event detection around him/her; the collected RFIDs are transferred to the server over the mobile communication network.

In order to control the portable tag reader, collect the tag information, and transfer the information via the mobile communication network, some kind of platform is required. In the assumed environment, the user terminal acts as this platform.

(2) Subordinate control for the server

It is difficult to provide the service that the user desires all of the time even if the server perceives the surrounding from a huge amount of information and an advanced inference engine is used. Therefore, it is necessary to provide a service evaluation function, and feed the user's feelings back to the server. Moreover, it is considered that the user must make the final decision on important matters such as those related to the user's life and property. The user terminal must offer support functions to achieve these goals.

(3) Actuation function
In order for the ubiquitous service to reach the user, the mechanism of service actuation is required. If shared terminals and robots, etc. are used as the actuator, information related to the user's privacy and security will be leaked in public spaces. Therefore, the user's terminal should become an exclusive actuator that provides adequate privacy.

2.3 General Requirements for the User Terminal

As described in the previous section, the user terminal is needed in the environment considered here. In this section the general requirements for this user terminal are described.

- **Portability:** Because each user always has to carry the user terminal, it should be light and small.
- **Low power consumption:** Because the user terminal must always be active when the ubiquitous service is required, it should have low power consumption and should run for long periods without battery change.
- **Low cost:** This seems to be a fundamental requirement for every user.

2.4 Other Work Related to the User Terminal

In this section the authors describe the previous research on user terminals for the ubiquitous computing environment.

Project Oxygen of MIT uses the user terminal called H21 [3]. Unlike the environment assumed in this paper, Project Oxygen uses image and voice to gauge the user's surrounding. Accordingly, H21 has a camera and a microphone. Additionally, the perception and offering service is provided by H21 itself, so a huge amount of processing performance is required which increases the electric power consumption and cost.

The ubiquitous communicator [4] is a small, light user terminal like a PDA, which has an RFID tag reader. It can read ucode-compliant RFID tags and display the linked information. However, it does not target the RFID-based environment perception services for mobile user described in Section 2.1.

3 Thin Client Based User Terminal

As described above, existing research does not satisfy the requirements for the user terminal in the assumed environment. Accordingly, the authors propose the thin client based user terminal; it is a small, light, and low cost terminal (e.g. cellular phone) that realizes RFID-based surrounding perception and acts as the actuator for the ubiquitous services.

3.1 Basic Architecture

The thin client [5][6]is an architecture that concentrates execution, storing, and management of the application on the server, and the client function is limited to HMI(Human Machine Interface) and some part of I/O. Therefore, a simple terminal can realize the functions and the performance of a PC (Personal Computer). In particular, previous research, called mobile thin client [7], introduced a cellular phone-based thin client system that enables PC applications (e.g. Word, Excel, and PowerPoint) to be used through cellular phone.

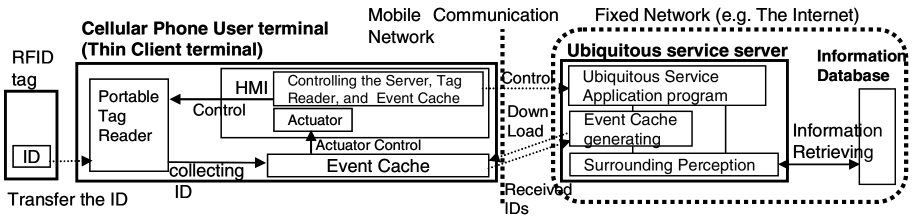
Mobile Communication Network
Fixed Network (e.g. The Internet)
Cellular Phone User terminal (Thin Client terminal)
RFID tag
ID
Transfer the ID
Portable Tag Reader
HMI
Controlling the Server, Tag Reader, and Event Cache
Actuator
Control
Actuator Control
collecting ID
Event Cache
Control
Down Load
Received IDs
Ubiquitous service server
Ubiquitous Service Application program
Event Cache generating
Surrounding Perception
Information Database
Information Retrieving

Fig. 2. Architecture of proposed user terminal

In this paper, the authors extended the mobile thin client to realize a cellular phone-based user terminal. Figure 2 illustrates the architecture of the proposed terminal. The HMI function for controlling (start, terminate, correct, selection of critical choices) server ubiquitous service is simply inherited from the mobile thin client. All other functions such as attached portable type tag reader, collecting the RFIDs function, and actuator control functions (BEEP, vibration, backlight blinking, and messaging to the user via the display of the cellular phone) are extensions. The collected RFIDs are transferred to the server via the mobile communication network. The server retrieves the information about the objects from the information database, perceives the user's surroundings from this information, and generates actuator control instructions for the cellular phone. In addition, the proposed architecture offers the function of continuing the ubiquitous services even when the mobile communication network is temporarily disconnected. This mechanism, called event caching, is described in detail in the following section.

3.2 Event Caching

The major problem of the thin client system, developed in the authors' previous research, is that when the communication link between the thin client and the server is disconnected, the thin client fails to work. Because the population cover rate of mobile communication network has already reached 100% in Japan, it has far better connectivity compared to the range of fixed tag readers. However, temporary disconnection is common in some environments such as inside subway trains and areas of heavy network traffic. Because the targeted services described in Section 2.1 are related to the user's life and property, these interruptions should be offset.

The event cache guarantees service continuity. It is achieved by both event prediction in the server and event detection and caching the reaction (i.e. actuator control) in the thin client. After the server perceives the user's current surroundings, it predicts the next event, and then generates the appropriate reaction that should be done when the event occurs. This information is downloaded to the event cache in the user terminal and held until needed or replaced by newer information.

3.2.1 Event Prediction

In this section, event predictability is described using the example of the "Lost Property Notification service", a typical ubiquitous computing application described in Section 2.1. It seems easy to judge that the user still has his/her property (object) from the reception by his/her tag reader of the object's RFID tag signal. When the object's signal is no longer received, the ubiquitous application should judge whether the user put the object somewhere on purpose, or he left it somewhere in error, in order to generate the correct reaction. Such processing has to be performed by the server because it requires much processing power and access to a comprehensive information database on the fixed network.

The authors note that the number of objects that should be observed is limited in several ubiquitous applications including the Lost Property Notification service. The server recognizes the current state of each object and decides what kinds of events are possible in the near-term future (e.g. if the user picks up an object, the server predicts that he/she may drop it as a possible next step). Moreover, by using the locator tag described in Section 2.1, the server can recognize the current place of the user; moreover, it is considered that the places that the user will pass are limited and predictable in the short term. It is considered that by using the features of each object (role, price, etc.), the kinds of possible events, and features of the places where an event may occur (public area or private area), the content of the reaction (level and content of warning) that should be executed when the event occurs, can be predicted. The number of events for which reactions are needed is limited because all of the above parameters are limited. After the server identifies the events that are possible in the near-term future and generates the corresponding reactions, it downloads this information into the user terminal. It is a relatively simple task for the thin client terminal to observe the event and execute the appropriate reaction quickly even if the mobile communication network is disconnected.

3.2.2 Event Classification

In this section the authors define the states and events. The term *state* means the relationship between the user and the object, and the term *event* means a state transfer. Because of the assumption that the user always carries a tag reader, it is possible to simplify the definition to three states and three events as described below.

WAITS_FOR state is the state in which the RFID reader cannot receive the signal from the target RFID tag. In the real space, this state corresponds to the situation in which the user is not carrying nor encounters the object. When the reader finds the targeted RFID, it raises the FOUND event and the DETECTED state is entered.

DETECTED state is the state in which the RFID reader first receives a signal from the target RFID tag. In the real space, this state means that the object and the user have approached each other. When the reader keeps receiving the signal for some time, the KEEP event is entered, and then the HAS_A state.

HAS_A state is the state in which the RFID reader continuously receives the signal from the target RFID tag. In the real space, this state means that the user is carrying the target object. When the signal from the tag is broken (LOST event), the WAITS_FOR state is entered.

3.2.3 Event Cache Construction

Figure 3 illustrates the construction of the event cache. Each entry in the event cache is composed of an entry identifier part, target RFID part, event detection condition part, reaction definition part, and linked entry definition part. The entry identifier part is the entry identification number in the event cache. The target RFID part describes the RFID value to be observed.

In some kinds of services, such as the shopping list service described in Section 2.1, the user terminal should be in the WAITS_FOR state where the RFID values are unknown. That is, the user has decided the kind of object to be bought, but does not know the entire RFID value. In order to support such situations, the authors propose to introduce the standardized category field that identifies the kind of the object into the RFID format. This allows the event cache to detect the event by the category field value (a part of the RFID) instead of the entire RFID value. In Figure 3, entry #02 WAITS_FOR any object belonging to category “beef (indicated by the code C)”.

The Event detection condition part defines the detection condition of the event according to the tag state condition and the time condition. The tag condition can be specified by not only the state of the observed RFID tag state but also the related RFID tag state. The related tag state is assumed to define the place condition retrieved by the locator tag. The reaction definition part defines the reaction that is to be executed by the user terminal actuator and the execution priority of the reaction when the event detection condition is satisfied. If the communication link is available, when the event occurs, the execution priority provides the definition of whether priority is given to the execution of the reaction (P (A) in Figure 3) or to communication with the server which will then execute advanced inference including the situation around the user (P (B) in Figure 3).

Entry ID	Target RFID		Event detection condition				Linked entry def.	Reaction definition
	tag ID	Cate gory	State of tag	Related tag		Time		
				tag ID	State			
#01	101	-	HAS_A	-	-	-	-	B, M ("You dropped Purse!"), P(A)
#02	-	xCx	WAITS_ FOR	303	DETE CTED	-	-	L, M (“Please buy beef at super”), P(B)
#03	301	-	DETECTED	-	-	-	#04	L, M (“Please go to the super”), P(B)
#04	-	-	-	-	-	18:30	#03	L, M (“Please go to the super”), P(B)

xCx: x means ignored field, C is category identifier for the object “beef”. B, L, M (), and P () are flags of Beeping, Lighting, Messaging on display, and priority.

Fig. 3. Construction of the event cache

It is considered that there will be cases in which several events are linked, so if one specific event occurs then the other will no longer be possible or required. For this case, if the first event cache entry is hit, the second should be deleted automatically. The linked event part provides support for these linked events. In Figure 3, if entry #03 (#04) hit, entry #04 (#03) is automatically deleted.

3.3 Service Scenarios Using Event Cache

In this section, the authors describe an example of event cache activities in the case of the Lost Property Notification service.

The user starts the service through the user terminal when he/she leaves home. The server recognizes the user-carried objects by observing the RFID tags collected by the user terminal over some period. If the user terminal receives a signal from an RFID constantly, the corresponding entry of the event cache moves to the state of "HAS_A". As a result of this process, entries #01 and #02 in Figure 4 are generated and downloaded to the user terminal. If the server detects the possibility of the user's visiting his/her friend's home from the tracking information from the received locator tag, the server adds entry #03, see Figure 4.

While the user is in a subway car and the mobile communication network cannot be used, if object "Purse" is dropped, event cache #01 is activated and the user terminal issues a maximum strength warning (Beeping, Vibrating, Lighting, and Messaging) to the user at once. On the other hand, if the place where the event occurs is the friend's house and if the object is not a purse but an umbrella, event cache #03 is activated and the user terminal tries to communicate to the server prior to notifying the user, because the execution priority of #03 is B. If the mobile communication network cannot be used, the user terminal issues a warning via the message display. If the user terminal can link to the server, it transfers all logged IDs to the server. The server then retrieves information about these objects. If there are many objects owned by the friend where the event occurred, the server recognizes that the user left it at the friend's home intentionally, and changes the state of the umbrella to WAITS_FOR.

Entry ID	Target RFID		Event detection condition				Linked entry def.	Reaction definition
	Tag ID	Cat.	State of tag	Related tag		Time		
				tag ID	State			
#01	101	-	HAS_A	-	-	-	-	B, V, L, M ("You dropped Purse!"), P(A)
#02	201	-	HAS_A	-	-	-	-	V, L, M ("Didn't you leave your umbrella?"), P (B)
#03	201	-	HAS_A	302	DETE CTED		-	L, M ("You left or mislaid your umbrella at Mr. A's home"), P(B)

B, V, L, M (), and P () are flags of Beeping, Vibrating, Lighting, Messaging on display, and priority. tag101 is the user's purse, 201 is the user's umbrella, and 302 is the locator tag near the friend's home.

Fig. 4. Event cache transfer in the Lost Property Notification Service

4 Implementation

The authors implemented a prototype. The RFID tag/reader is based on the SPIDER V system [8]. SPIDER V is an active type tag system, so each tag transmits its ID periodically for the reader to capture it. The event cache was implemented by i-appli (DoJa2.1) [9]. Because it is not possible to connect the cellular phone directly to the SPIDER V reader, the authors used a cellular phone emulator on a note PC. The SPIDER V tag reader was connected to the note PC via a serial interface.

Generally speaking, even if the tag is within the reader's range, the tag reader can not receive 100% of the transmitted signal because of signal failures caused by the characteristic of electric wave propagation and collision with other tag signals. Therefore, the user terminal stores the history of several signaling periods, and the KEEP and LOST events are defined by whether the received signal number exceeds some threshold. This means that there is some delay in detecting the corresponding event. Maximum delay value of the LOST event is as follows.

$$(T\text{-}m) \times t+t \text{ [seconds]} \tag{1}$$

where,

T is the number of stored periods, m is a threshold number, and t is tag signaling period.

5 Evaluations and Consideration

(1) Service consistency in the disconnected situation

Figure 5 illustrates the basic operation sequence in the Lost Property Notification service. Here, t=1, T=10, and m=3. First, service start is acquired and the list of carried objects is displayed (step 1). Next, the communication link between note PC and the server was cut (step 2). The tag corresponding to "Purse" was put into a shielded box (step 3), The Lost event was detected nine seconds after step 3, the cache was executed, and the warning was displayed on the emulator screen (step 4). Finally, the communication link was recovered and the new cache was downloaded by the server and the display was updated (step 5, 6). The results confirmed service consistency in the disconnected situation.

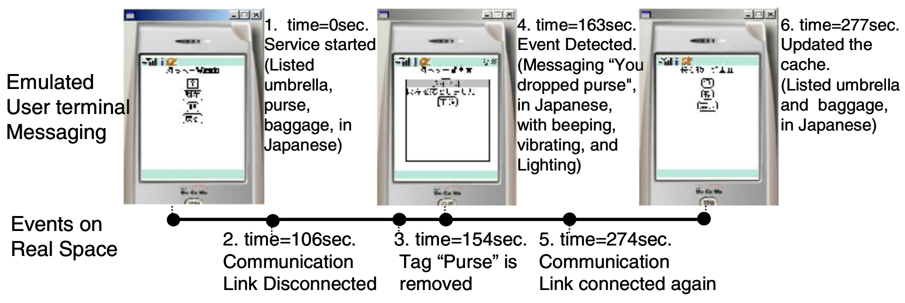
Emulated User terminal Messaging
1. time=0sec. Service started (Listed umbrella, purse, baggage, in Japanese)
4. time=163sec. Event Detected. (Messaging "You dropped purse", in Japanese, with beeping, vibrating, and Lighting)
6. time=277sec. Updated the cache. (Listed umbrella and baggage, in Japanese)
Events on Real Space
2. time=106sec. Communication Link Disconnected
3. time=154sec. Tag "Purse" is removed
5. time=274sec. Communication Link connected again

Fig. 5. Basic operation sequence in the Lost Property Notification service

(2)Event detection delay

From equation (1), the event detection delay can be reduced if m is increased, or T or t is lessened. Because excessively small t causes tag signal collision and shortens the life of the tag battery, the authors fixed t to 1 and tried to enlarge m and lessen T. Table. 1 shows the experimental results (ten times average and standard deviation) and the ideal value from equation (1) of event detection delay when T=10/m=3, T=10/m=7, and T=6/m=3. The difference, a few seconds, between the ideal value and the experimental value is due to the communication delay of the tag collecting function and event cache, thread switching timing, etc. In addition, when execution priority is B, another few seconds of delay is added for communication to the server.

Table 1. Evaluation results of event detection delay

	Average detection delay (standard deviation) [seconds]		
	T=10/m=3	T=10/m=7	T=6/m=3
Ideal value	8	4	4
execution priority A	9.3(0.63)	5.3(0.64)	5.1(0.66)
execution priority B	11.1(0.58)	7.0(0.54)	6.5(0.57)

The results of Table 1 show that actually the delay was lessened when the T=10/m=7 and T=6/m=3. However, too large m and too small T causes another problem. Figure 6 illustrates the 100 signaling period monitoring results of the event cache internal parameter for event detection (i.e. when this value becomes less than m the event cache detects the LOST event) when m is set to 7. The Y axis shows how many signals from the tag were received in the last T (in this case T=10) periods, and the X axis shows the number of periods. Figure 6 shows that the false event of LOST was detected even though the tag - antenna distance was 1 meter. This is considered to reflect the influence of signal failure described in Section 4. Similarly, too small T values (T=6/m=3) caused the false detection as in the other experiment. The user will not adopt the ubiquitous service if there are many annoyances like false detection. On the contrary, too large T and too small m enlarge the delay. The authors think that m

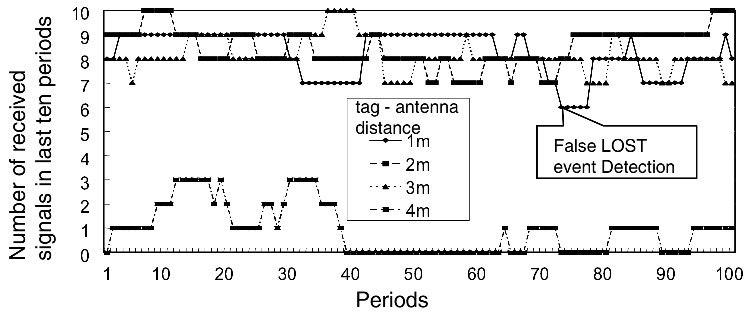

Fig. 6. Tag detection history: ten cycle windows over 100 cycles

should be 3 and T should be set to 10 according to the results of Figure 6. The average delay is 9.3 seconds from Table 1. The authors consider that this result satisfies the real-time requirements for mobile user observation ubiquitous services described in Section2.1.

6 Conclusion

In this paper the authors examined the user terminal needed for realizing ubiquitous computing services. Our terminal is based on the thin client architecture and uses event caching in order to provide service continuity even if the communication link between the client and the server is disconnected. A prototype and evaluation results were described. The results indicate that the proposal is sufficiently practical. Detailed evaluations such as event prediction and field experiments are being planned.

References

1. Weiser, M.: The Computer for the 21st Century. Scientific American (1991) 415-438
2. Takahashi, T., Mizuno, T.: Thin Client-based Handheld Device Architecture for Ubiquitous Computing. Proceedings of Workshop on Informatics 2004 (2004) 330-334, in Japanese
3. MIT PROJECT OXYGEN. http://oxygen.lcs.mit.edu/
4. Ubiquitous ID Center. http://www.uidcenter.org
5. Sinclir, J., Merkow, M.: Thin Clients Clearly Explained. Morgan Kaufmann, San Francisco (2000)
6. Kanter, J.: Thin Clients/Server Computing. Microsoft Press, Washington (1998)
7. Takahashi, T., Takahashi, O., Mizuno, T.: A Study of a Thin Client System for Mobile Computing. IPSJ Journal, Vol.45, No.5 (2004) 1417-1431, in Japanese
8. SPIDER V System. http://www.nextcom.co.jp/solutions/rfidrfid/spiderv.htm
9. http://www.nttdocomo.co.jp/p_s/imode/make/java/index.html