Personal Home Server:

Enabling Personalized and Seamless Ubiquitous Computing Environments

Tatsuo Nakajima[†] and Ichiro Satoh[‡]

[†]Department of Computer Science, Waseda University

[‡]National Institute of Informatics

tatsuo@dcl.info.waseda.ac.jp, ichiro@nii.ac.jp

Abstract

In this paper, we propose a *personal home server* that makes it possible to coordinate various home appliances. Since each person has a different personal home server, it allows us to personalize how to use the appliances according to his preference. Also, our personal home server can discover and configure various appliances at any places, such as railroad stations, cars, and streets as well as houses in a seamless way. We show the design and implementation of a personal home server, and discuss the current design and future work.

1 Introduction

Our daily lives will be changed dramatically due to the popularity of ubiquitous computing technologies. The technologies make us to use information about the real world, that could not be used in a cyber space before by using various sensor technologies. Also, we will be able to control various everyday objects that could not be controlled before by embedding small computers in our world. The technologies will make it possible to integrate various aspects in our lives seamlessly.

In future ubiquitous computing environments, we will have several personal devices carried with us everyday like a cellular phone or a wrist watch. These devices will contain more functionalities than now, and it is important to consider what these devices can do in the future.

We believe that the combination of devices embedded in our environments and a personal wearable computer is a key to reduce complexities in our lives. In our future daily lives, a variety of devices will be embedded in various places. However, currently, we do not know how to coordinate these embedded devices. The number of these devices will be increased dramatically, and it is not possible to adopt the current middleware infrastructures because they support spontaneous interactions in a very weak manner. The most important issue for a future middleware infrastructure for ubiquitous computing is to reduce complexities in our lives. In our case, it is important to reduce the number of home appliances that we can control according to the current situation. Also, the middleware should support the seamless movement of a user. This means that a way to control appliances should not depend on the place where he is.

We propose a *personal home server* to solve the problems. A personal home server is carried by each person, and configures home appliances near him in a spontaneous

way. The personal home server has the following characteristics.

- Our system allows us to configure to control home appliances according to each user's preference and situation.
- Our system adopts standard protocols as much as possible to incorporate commercial products.
- Our system can be used to configure various appliances at any places.

In this paper, we show the design and implementation of our personal home server platform.

2 Home Computing: Current Status

Many consumer electronic companies are recently interested in a home gateway to integrate home appliances. The gateway allows us to control home appliances from a variety of devices such as a universal remote controller, a personal computer and a cellular phone. Also, the gateway shows a graphical user interface to integrate several home appliances. There are several standard middleware infrastructures for implementing home gateways such as HAVi, Jini and UPnP, and several real products adopting UPnP appear in markets. However, we believe that the traditional approach has the following problems.

The first problem is that we need to buy a new device to access advanced services in each place. We believe that the functionality to coordinate appliances should not be embedded in environments because the approach requires expensive infrastructures to acquire context information, and it is not easy to install the infrastructures in existing houses. The second problem is that it is difficult for a home gateway to protect a user's privacy. Privacy information is required to provide better personalization services, but a home gateway needs to support a complex globally distributed authentication service to protect privacy. Also, even an owner of the home gateway should not allow to access the privacy information. We believe that the approach requires complex and expensive infrastructures to protect privacy information.

There are several approaches to use personal devices for supporting spontaneous interactions. For example, in [5], a traditional personal computer is decomposed into a server carried by a user and interaction devices embedded in our environments. SpeakEasy[1] is a software platform

to support spontaneous interaction between devices and services. The system adopts a component-based approach that allows us to collaborate among devices and services by connecting components. The personal home server can offer similar functionalities provided by their systems, but our approach takes into account personalization and security to provide more attractive services.

3 Design Issues

Personalized Services and Privacy: Our future daily lives will be more and more complex. The personalization of services is a key to reduce the complexities. The personalization requires to use personal information, and the information should be hidden from other persons to protect a user's privacy.

In our approach, a user can carry his/her own personal home server. Thus, the approach can maintain personal information within the user's personal home server, that is always bound to each user. The server does not leak the personal information about its user to other parties. The approach can introduce a mechanism for dynamically generating HTML-based user interfaces for controlling appliances according to the personal information.

In our approach, the personal information can be used to personalize services at any places since the information is always carried by a user. This means that we can use the same personalization policies whenever we like to access appliances. In the future, our social relationship will become complex, and it is better not to share each user's privacy information because the personal relationship will be dynamically changed according to each social situation. A personal home server can solve the problem, and complex infrastructures are not necessary to protect privacy.

Simplicity: Currently, generic appliances are popular, but it is necessary to decompose the generic appliances into specialized appliances to reduce complexities caused by the genericity. In the future, everyday objects may give various information through embedded sensors. Also, the objects should be integrated by middleware infrastructures with home appliances. Therefore, simple protocols are desirable for future devices embedded in our daily environments.

Our approach adopts SOAP to control home appliances. We believe that SOAP is one of the most promising candidate as a common protocol that various middleware support. The approach allows various middleware infrastructures like Jini and HAVi to communicate with each other through SOAP-based proxies in an easy way as described in [2]. Also, the approach can integrate Web services with our home computing service in an easy way by using the same protocol. We believe that integrating Web services and home computing services is useful to provide more pleasurable and useful services. Also, our approach offers a light-weighted security mechanism to operate appliances. In our approach, each personal home server can know the existence of appliances that have the permission to operate.

Naming Scheme: We use URL as an intermediate representation for not only specifying but also controlling ap-

Figure 1: Overview of Personal Home Server

pliances, because URL is one of the simplest representations to identify networked devices from a variety of existing user-side devices, such as smart cellular phones, PDAs, and PCs.

Today, it is popular that home appliances can be controlled from a Web browser. However, the POST request is used to transmit a command in usual cases, and the approach is not easy to transmit the command from various popular presentation documents such as Flash, MS Power Point or SMIL documents that will be widely used as user interface for future appliances. Also, it is difficult to automatically generate an HTML document containing commands for detected appliances near a user because each command requires a different HTML document to transmit a POST request. The fact limits the design of our user interface. Our naming scheme allows a user interface to transmit a GET request that is used in Flash, MS Power Point and SMIL. Also, our approach allows us to specify our preference in URLs as attributes for the appliances.

We believe that our approach allows interaction designers to design various home computing applications by using their favorite presentation design tools. Thus, the cost to develop an appliance will not become expensive.

4 Design and Implementation

Figure 1 presents an overview showing how a personal home server works. A personal home server is implemented in a personal and portable devices like a cellular phone, a wrist watch, or a jacket. Thus, the server can be carried by a user anytime anywhere.

4.1 Structure of Personal Home Servers and Appliances

In ubiquitous computing environments, it is not easy to specify which functionality is useful in advance, because the required functionality should be different according to a user's situation. In our approach, a server consists of several components and each component should be dynamically loaded or unloaded according to the current situation.

Our system has been written in Java, and we have adopted the OSGi(Open Service Gateway initiative) Service Gateway Specification as a component framework.

The system is currently running on Linux, and we are using the Blackdown's Java virtual machine. In the current prototype, we have adopted Oscar which is an open source implementation of the OSGi service gateway specification.

4.1.1 Structure of Personal Home Server

In the current implementation, we have constructed 12 components. *SOAP*, *GENA*, *SSDP*, *UPnP*, *HTTP Server*, *HTTP Clients* are components for implementing the UPnP protocol. Also, the protocol is used to offer services implemented in a personal home server. For example, a user can notify his/her friends by sending a command to an alarm service on the friends' personal home servers.

Database is a component to store information about appliances near a user. *ContextInference* is a component to support context-awareness in our system. *In-Forwarder* is a component to receive a command encoded in a URL and converts the command to SOAP. *Out-Forwarder* is a component to transmit a SOAP command to a target appliance. *Jini/SOAP* is used to send a command to Jini devices by converting a SOAP command to a RMI invocation. The component is used to show that our system can control various appliances supporting different home computing middleware infrastructures[2]. *UI* is a component that generates an HTML-based user interface document from the information specified in service specification documents received from respective appliances.

4.1.2 How Does a Personal Home Server Work ?

The personal home server automatically collects information about home appliances near a user by using the SSDP component, and creates a database storing information about these appliances in the Database component. Then, it creates a presentation document containing the attributes of appliances and the commands to control them by the UI component automatically. A display near the user also detects the personal home server, and retrieves the presentation document containing the automatically generated user interface. The display shows the presentation document. The document contains URLs defined by our naming scheme and embedding the attributes of appliances and their commands, that are retrieved from the collected information received from appliances near a user that is stored in the Database component. Thus, a user needs not to enter URLs by hands to control appliances, and the presentation document is customized according to a user's preference by using the ContextInference component. When a user touches the display, a URL containing the attributes of an appliance and its command is transmitted to his personal home server via the HTTP message. The In-Forward component translates the URL to a SOAP command by sending a query to the Database component and the query is customized by the ContextInference component. Finally, the SOAP command is forwarded to the target appliance for the Out-Forward component.

4.2 URL-based Naming Scheme

As mentioned previously, our approach allows us to specify commands for appliances within URL convention, because URL is supported by many user-side network-enabled devices and content documents. Since URL is inherently designed to identify resources available in a

network, it cannot operate appliances. Therefore, we extend URL with the ability to control appliances without modifying the syntax of URL. That is, the extension is defined within the syntax of the standard URL, but the path elements of the URL form can contain some additional information for managing appliances according to the following ABNF-like syntax:

```

path      = *(directory) request
directory = *(directory) / name
request   = *( "&" request | search | command
search    = "%3F" pair ;%3F = ?
command   = "!" pair | "!" name
pair      = name "=" value
name      = <HEX escaped string>
value     = <HEX escaped string>

```

In the above definition, the ? attribute specifies a query expression and is a field-value pair, where a field describes the property of the query (such as location, or owner), and a value is a string or an integer. The ! attribute specifies a command, where a field describes the property of the command, and a value is a string or an integer. The URL definition is very flexible because we can specify various attributes to identify a target home appliance. We can also use attributes that represent context information such as location.

```
http://A.B.C.D/room1/?function=light&?type=ceiling&!power=on
```

The above URL is a typical example of our naming scheme. The command is transmitted to a personal home server whose IP address is A.B.C.D. Since a presentation document is generated by a personal home server, its IP address can be included in the presentation document automatically. *room1* specifies a location in the house. This example assumes that a user wants to control a ceiling light. The *?function=light&?type=ceiling* element specifies an electric light on the ceiling in *room1*. The *!power=ON* element signals to the server and nominates a specific function, where the *power* term is an attribute corresponding to the power supply and the *ON* term is a command of the *power* term. The example shows that our scheme can identify an appliance that a user likes to control by specifying their attributes. We believe that the approach is simple and enough powerful to identify appliances than other directory services.

4.3 Appliance Discovery

In our system, a personal home server knows respective appliances via the SSDP protocol. Each home appliance transmits advertisement messages periodically. Also, a personal home server asks respective appliances to transmit advertisement messages by sending a SEARCH message. After receiving the advertisement messages, the personal home server transmits an HTTP GET request to get a service specification document to each appliance.

The Database component stores all service specification documents detected currently. The component asks the SSDP component to transmit the SEARCH message periodically, and the service specification document of an appliance is removed if there is no advertisement message within a fixed duration.

If the service specification is changed, the change is propagated quickly since the SEARCH message is transmitted periodically. However, if the number of SSDP messages is increased, the battery consumption of a personal

home server may be a problem. We need to investigate how to reduce the number of messages when the number of appliances is increased in the future.

It is not practical to control any appliances from each user's personal home server. Thus, we offer a security mechanism to protect appliances. However, the mechanism should be simple to be implemented in a small devices. Our security mechanism extends the above appliance discovery protocol, and the personal home server can know the existence of appliances that have permission to operate.

In our scheme, we assume that each appliance offers RF tags that contain an encryption key. The SSDP messages transmitted from an appliance are encrypted by the key. A user who likes to use the appliance needs to insert one of the tags into his/her personal home server. The personal home server has an RF tag reader, and the reader retrieves the key from the inserted tag. The key is registered in the database of the server.

When a personal home server receives an SSDP message, the message are decrypted by the key registered in the home server. If the decryption is successful, the personal home server transmits an HTTP request to get a service specification document. Currently, we have implemented our scheme by using the RF code Inc.'s spider system.

When a user likes to control multiple appliances, s/he needs to have tags for respective appliances. We assume that future tags will be very small, and they can be inserted in a personal home server in a convenient way. In our approach, the key is discarded when a corresponding tag is removed after its expiration time is passed.

Also, a user can give a key temporally to her friend to access home appliances in her home. In this case, she closes her personal home server to her friend's personal home server. The keys in the her personal home server are stored in her friend's personal home server. The keys will be discarded after the expiration time is passed. Also, we like to extend our approach in the future to discard the keys when her friend goes out from her home. If she likes to give the permission of a particular appliance, she closes her tag of the appliance to her friend's personal home server.

In public space, tags containing keys to operate appliances are embedded in the appliances. Also, the keys can be embedded in various object such as tables and walls. When a user closes to the appliances, tables or walls, his/her personal home server stores the keys to decrypt SSDP messages received from appliances.

4.4 Service Management

Our service specification document is represented as an RDF document, and contains attributes of the appliances. The attributes of the document are used to identify a target appliance. For example, if a URL for specifying an appliance, which is automatically generated by the UI component, contains `?function=light&?type=ceiling`, the `function` tag and `type` tag in the document are examined to find the target appliance. Some attributes can be filled by sensors attached to a home appliance. For example, if an appliance can know its location through an attached location sensor, it fills its location attribute in its service specification document. The approach can be

used to find appliances in a context-aware way. Also, the service specification document contains a link to a WSDL document identifying commands that can be accepted. If an appliance contains several functionalities, its specification document may contain several links to WSDL documents.

The Database component offers two functions to access information in the database. The first function invoked by the In-Forward component is used to transmit a command. When a personal home server receives a command, it extracts the attributes of a function for a target appliance like `?func=television` in the URL. The Database component returns an IP address of the appliance. Thus, the In-Forward component converts the command to a SOAP request. Let assume that a URL contains `!power=on`. The command is converted to `power("ON")` that is transmitted on a SOAP message. Then, the Out-Forward component transmits the SOAP command to the target appliance specified by the IP address. The second function invoked by the UI component is used for generating a presentation document. Currently, we support an HTML based presentation document. The function returns an array containing information of respective appliances. The information of the appliances includes the names, command names, and the attributes of the appliances. These information is converted to URLs, and they are embedded in an HTML document.

4.5 Context Management

The ContextInference component allows a personal home server to process commands in a context-aware way. When an In-Forwarder component receives a URL, the component finds an IP address of a specified appliance. Let us assume that the URL contains `"?function=TV"`, and the statement is passed to the ContextInference component. The component expands the statement by using rules specified in a personal home server. The rules are stored in the context database managed by the ContextInference component. We assume that a server contains the following rules.

```
Function(x, TV) -> Location(x, living-room), Support(x, BS).
Function(x, light)
-> Type(x, ceiling), Location(x, y), Location(me, y), Room(y).
```

In this case, `?function=TV` is expanded to `?function=TV&?support=BS&?locations=living-room`. The example shows that our system selects an appliance according to a user's preference specified in rules stored in each person's personal home server. Thus, a television that supports the BS digital (one of Japanese digital satellite broadcasting methods) and resides in a living room is selected.

Also, ContextInference customizes a presentation document according to a user's preference. For example, let us consider the above rules. When a personal home server detects various types of light appliances, the second rule filters light appliances whose type is not a ceiling light, and the presentation document contains only ceiling lights that reside in a room where a user is. A personal home server omits information about other types of light appliances. Also, the presentation document contains information about televisions whose location is a living room and they support the BS digital according to the first rule.

The example illustrates how a personal home server can generate a customized presentation document according to a user's preference.

4.6 User Interface Generation

This section presents an example showing how the UI component automatically generates HTML documents for controlling home appliances. In the example, we assume that there are a television and two lights in a room. After detecting these appliances, a user's personal home server generates three HTML documents. The following shows the first and third documents.

```
<HTML>
<HEAD>
<TITLE>A Document for Controlling Detected Appliances</TITLE>
</HEAD>
<BODY>
<LI> <a href='http://host/?func=TV&power=on/'> TV Turn On </a>
<LI> <a href='http://host/?func=TV&power=off/'> TV Turn Off </a>
<LI> <a href='http://host/?func=TV&channel=1/'> NHK</a>
<LI> <a href='http://host/?func=TV&channel=3/'> CNN</a>
<LI> <a href='http://host/tmp/123.html'> Light Power On</a>
<LI> <a href='http://host/tmp/124.html'> Light Power Off</a>
</BODY>
</HTML>

<HTML>
<HEAD>
<TITLE>A Document to Turn Off Lights</TITLE>
</HEAD>
<BODY>
<IMG src='http://host/?func=light&place=left&type=ceil&power=off'>
<IMG src='http://host/?func=light&place=right&type=ceil&power=off'>
</BODY>
</HTML>
```

The first HTML document contains commands for controlling the television and URLs to the second and third documents. The second document includes URLs to turn on two lights simultaneously, and the third document contains URLs to turn off the lights. The information in the service database of the Database component is used to generate URLs in the documents. When the last URL in the first HTML document is accessed, the third HTML document is retrieved in a browser, and two URLs in the document are accessed simultaneously to turn off the lights.

In the documents, the IP address in the URLs is the personal home server's IP address. Also, attributes in the URLs are retrieved from the service specification documents for light and television appliances stored in the Database component. Also, the command names in the URLs are retrieved from the WSDL documents of these appliances. In the example, the *func*, *place*, and *type* attribute can be retrieved from the service specification documents, and the *power* and *channel* command are retrieved from the WSDL document.

Our personal home server need not to offer a display. The server can use a display near a user. The server transmits an advertisement message to a display device. Then, the display near him transmits a GET request to retrieve a presentation document from the personal home server. Finally, the document is rendered on a display by a Web browser. The approach enables us to create new attractive services. For example, an approach to deal with lights in the example allows us to trigger commands automatically when a presentation document is rendered. The service can present information implicitly similar to ambient displays. Also, as described before, our approach can use various presentation documents. For example, a Power Point document that contains URLs for controlling

an appliance allows us to show a sequence to access the appliance, which provides a new approach that a user can access the manual of an appliance. Each page contains simple instructions for an appliance, and a user controls the actual home appliance by clicking a part of the document.

5 Current Status and Future Work

We have implemented the personal home server, and developed several home appliances to control from our personal home server. Our personal home server detects a public display near a user, and shows user interface to control appliances. Our current prototype system supports an HTML-based user interface, but we are working on a Flash-based user interface to generate more sophisticated user interface. Also, we are currently working to personalize to control appliances by embedding personalization rules in RFIDs[4]. The approach makes our personalization scheme more practical and more pleasurable.

We have found that in the future we would need to solve the following two issues, and we are working on to attack the issues. The first issue is to dynamically load and select components that are suitable for the current resource limitation. Currently, we are working on a new component framework. In the framework, each component specifies a protocol to define the abstract behavior of the component. The protocol is represented as a state machine, and the state transition describes constraints of the transitions. The constraints are used to check whether the component can satisfy the requirements of a system before running it. The state machine can also show the dependencies among components and assumptions of an underlying platform. Also, a meta protocol specifies tacit dependencies among components. For example, a component that contains threads needs to say which scheduling policy is assumed to execute the component. The second issue is to offer various applications by extending our software infrastructure. In our vision called the *pervasive server architecture*[3], each object contains a small computer. The object may be an appliance, a sensor, or a daily object like a mug cup, and it contains a component to return its service specification. Also, a personal server can download various components to access the object. Each service is constructed by connecting these components. Thus, a personal home server can be built as one of applications of the system.

References

- [1] W.K.Edwards, M.W.Newman, J.Sedivy, T.Smith, S.Izadi, "Challenge: Recombinant Computing and the Speakeasy Approach", In Proceedings of Mobicom '02, 2002.
- [2] T.Nakajima, D.Ueno, I.Satoh, H.Aizu, "A Virtual Overlay Network for Integrating Home Appliances", In the Proceedings of the 2nd International Symposium on Applications and the Internet, 2002.
- [3] T.Nakajima, "Pervasive Servers: A Framework for Building a Society of Appliances", In Proceedings of 1st International Conference on Appliance Design, 2003.
- [4] T.Nakajima, "A Personalization Framework in a Personal Home Server: System Infrastructure for Designing Pleasurable Services", Technical Report, Waseda University, 2003.
- [5] R.Want, T.Pering, G.Danneels, M.Kumar, M.Sundar, J.Light, "The Personal Server: Changing the Way We Think About Ubiquitous Computing", In Proceedings of Ubicomp2002.