Universal Interaction with Networked Home Appliances Using Stateless Thin-Client Architecture

Tatsuo Nakajima and Atsushi Hasegawa
tatsuo@dcl.info.waseda.ac.jp
Department of Information and Computer Science
Waseda University
3-4-1 Okubo Shinjuku Tokyo 169-8555 JAPAN

Abstract

In ubiquitous computing environments, a variety of objects include computers to be intelligently behaved, and home computing is a typical instance of ubiquitous computing. Since networked home appliances contain powerful operating systems and middleware components, our daily lives will be changed dramatically due to the integration of these appliances. Also, the interaction between us and the appliances should change a way to utilize their advanced features dramatically.

A lot of research projects are working on new user interaction devices such as wearable user interface and PDAs as remote controllers. However, these systems require to build a completely new user interface systems. On the other hand, standard middleware components for networked home appliances assume to use traditional graphical user interface systems such as Java AWT or Swing. Therefore, it is necessary to combine traditional user interface systems with advanced user interaction devices.

In this paper, we propose universal interaction for networked home appliances, which is a simple mechanism to fill the gap between traditional user interface systems and advanced user interaction devices. The system allows us to select suitable input and output devices according to our preferences and situations. Our system has extended the VNC(Virtual Network Computing) system developed by AT&T Labolatories, Cambridge, which adopts the stateless thin-client architecture, and translates input and output interaction events according to user interaction devices. We also show the effectiveness of our approach in our home computing systems.

1 Introduction

Our daily lives will be dramatically changed due to a variety of objects embedding computers. These objects behave intelligently to extend our bodies and memories[11]. A lot of research projects are working on attacking various problems to realize these computing environments. These computing environments are widely called *ubiquitous computing*[2, 34]. Also, other researchers have proposed similar concepts called *pervasive computing*[5], *sentient computing*[12], or *things that think*[11]. In *ubiquitous computing environments*, a variety of objects are augmented by containing computers. Since any programs can be executed on the computers, there are infinite possibilities to extend these objects by replacing the programs. Also, these objects have networks to communicate with other objects, thus respective objects will behave more actively by replacing programs at each other according to surrounding situations. For example, our environments may memorize what is going on in the world instead of us, or each object tells us where it currently exists.

There are a lot of researches for realizing ubiquitous computing environments. Some research groups have been working on building prototypes of ubiquitous computing environments[7, 21, 33]. These projects show the impact of ubiquitous computing to our lives. Also, recently, several standard middleware specifications are proposed for realizing ubiquitous computing. Jini[14] and UPnP[30] provide mechanisms to discover a variety of services in distributed environments. Also, HAVi(Home Audio/Video Interoperability)[8, 28] enables us to develop networked home appliances. Therefore, ubiquitous computing in home environments will be realized in near future, and the ubiquitous computing environment is called home computing. This is a very interested field since home computing is directly related to our daily lives. Also, it is a good candidate to show the effectiveness of ubiquitous computing because every person expects radical advances in home computing environments.

In these environments, one of the most important problems is how to interact with a variety of objects embedding computers. The interaction between us and computers embedded in various objects has been developed by several research groups[4, 13, 17, 27]. These devices enable us to interact with embedded computer more naturally. However, current standard middleware components for networked home appliances have adopted traditional standard graphical user interface systems such as Java AWT or GTK+. Therefore, it is not easy to control home appliances from advanced interaction devices such as PDAs, cellular phones, or a variety of research prototypes described above. Also, natural interaction is changed according to a user's current situation. For example, if a user is cooking a dish. S/he likes to control appliances via voices, but if s/he is watching TV on a sofa, a remote controller may be better. This means that the most appropriate interaction device should be dynam-

ically chosen according to a user's current situation and preference, and the selection of interaction devices should be consistent whether s/he is living in any spaces such as at home, in offices, or in outdoor.

In this paper, we propose universal interaction for networked home appliances, which is a user interface system to fill the gap. Our system enables an application to use traditional standard graphical user interface systems such as Java AWT or GTK+, but a user can navigate the interface through a variety of devices such as PDAs, cellular phones, or advanced technologies. We show that it is possible to realize the goal very easily using the VNC(Virtual Network Computing) system[25], which is a stateless thin client system. We have built a prototype system, and show that a user can use a variety of interaction devices carried by him. The prototype system is currently integrated with our home computing system that have implemented HAVi(Home Audio/Video Interoperability)[8, 28], which is a standard distributed middleware specification for home appliances, and shows that our system is useful to control home appliances.

The reminder of this paper is structured as follows. In Section 2, we describe desirable user interface systems for home appliances. Section 3 presents the design and the implementation of our user interface system for home appliances. In Section 4, we show how our system works in our home computing system. Also, we describe a brief overview of our home computing system. Section 5 presents several experiences with building our system, and we describe related work in Section 6. Finally, in Section 7, we present future work and conclude the paper.

2 Interaction with Networked Home Appliances

Distributed middleware components such as HAVi enable us to control multiple networked home appliances in an integrated fashion. For example, HAVi defines API for controlling respective audio and video home appliances. A portable application is able to be created by using the standard API without taking into account the differences among vendors. However, the system gives us a traditional way to interact us with home appliances as shown in Figure 1. As shown in the figure, each home appliance has a different remote controller device to control itself. The approach is not good to support future home appliances since we may have a lot of appliances in our houses, and it will confuse us to use the remote controllers. For example, to control an appliance, we usually need a special remote controller, then we need to control several appliances independently, and we need to learn how to use appliances even if the same type of appliances is used. Also, when a user controls an appliance, a system does not understand the situation of a user. Therefore, a user needs to take into account which appliance he likes to control, where the appliance is currently located, and the differences among appliances such as vendors.

We believe that the following three issues are important to realize better interaction with future networked home appliances.

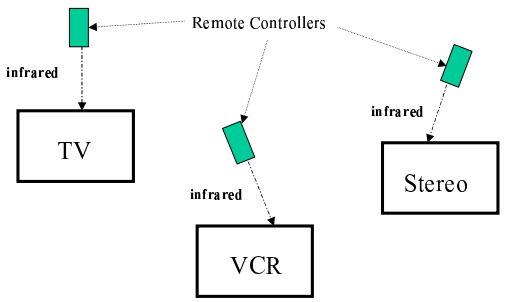

Figure 1: User Interaction with Home Appliances

- The appliances can be controlled with a variety of interaction devices such as PDA and cellular phone. Also, input events and output events may be processed in separated devices.
- It is possible to control multiple devices as one appliance.
- The user interface should be customized according to each user's preferences.

The most desirable interaction device to control home appliances should be changed according to a user's preference, currently available interaction devices from him, or his current situation as shown in Figure 2. For example, when both hands of a user are used for something, he cannot control appliances by traditional remote controllers, PDAs, or cellular phones. In this case, it is desirable that these appliances should be controlled by voice. Also, it is desirable to switch input and output devices according to a user's situation. While he sits on a sofa in a living room, he likes to use a universal remote controller to control TV, but while he is cooking, he likes to control TV via his voice. The situation is suddenly changed according to a variety of reasons such as a change of his feeling. This means that flexible interaction should be realized to control home appliances, for example, it is better to switch interaction devices dynamically according to a user's situation.

Also, the user interface system for networked home appliance should provide a mechanism to select input and output devices independently, and change them dynamically even if these devices are connected to different appliances. For example, a display device where user interface appears is dynamically changed according to the location of a user. If a large display device is not available in the current room, it is better to show the user interface on the screen of a user's PDA.

As the number of home appliances is increased, the control of these appliances will be difficult due to the complexities. In future, we will have a lot of home appliances in our houses. Therefore, a mechanism to virtually reduce the number of home appliances is required.

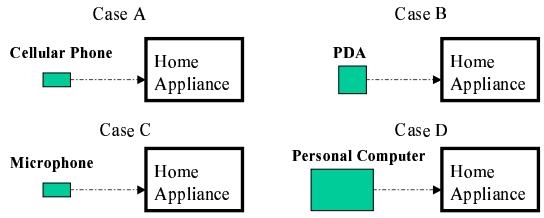

Figure 2: Controlling Appliances with a Variety of Interaction Devices

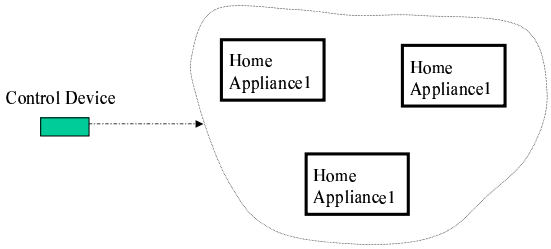

Figure 3: Composition of Appliances

One of the promising approaches is to compose multiple appliances into one appliance as shown in Figure 3. It makes possible to deal with these multiple appliances as one appliance. Currently, distributed middleware components like HAVi allow us to use respective functions such as tuner, amplifier, display and speaker in different appliances separately. Therefore, it is possible to compose these multiple functions to build a new appliance since a small function makes it easy to compose multiple functions.

Future home appliances will provide rich functionalities. However, if the number of functionalities is increased, this makes it difficult to use these home appliances by most of usual users. To support rich functionalities is important since each user's preference requires to use various optional functions. Therefore, it is important to customize user interface according to the user's preference[3]. Also, it is desirable to use context information of a user to customize user interface[6]. For example, the location information is useful to select a suitable appliance for the user as shown in Figure 4, and the information acquired by monitoring usual behavior of a user is useful to customize the user interface.

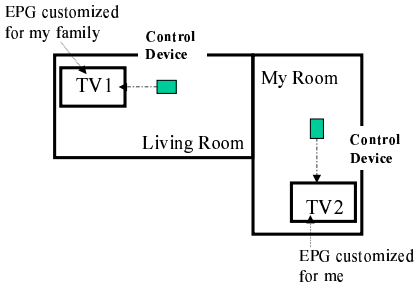

Figure 4: Context Awareness

3 Design and Implementation

In this section, we first describe several design choices for supporting the interaction with users in home computing environments. Then, we present the implementation of the current prototype in detail.

3.1 Design Issues

There are several ways to realize the goals described in the previous section. The most important issue is how to support a variety of interaction devices. Traditional user interface systems assume a few types of interaction devices. For example, mouses, track points, and touch screens are used as input devices. In this section, we describe three alternative approaches to build a flexible user interface system as shown below, and present why our approach is chosen.

- Builds a new user interface system from scratch.
- Builds a new device independent layer that invokes respective traditional user interface systems for a variety of interaction devices.
- Captures input and output events of traditional user interface systems, and transforms the events according to interaction devices.

In the first approach, a new user interface system that can be used with a variety of interaction devices from scratch. The approach is able to survive when a new interaction device appears because the system takes into account to accommodate new interaction devices. However, we need to modify existing applications and middleware components for incorporating the new user interface system, and the development needs to take a long time. Many home computing systems already have adopted existing solutions, then it may be impossible to replace the currently used user interface systems.

In the second approach, there are two alternative to support a variety of interaction devices by adding a new layer on traditional user interface systems. In the first alternative, an interaction device independent layer translates the standard requests of applications to the requests

for respective user interface systems. For example, in document based approaches[9, 21, 31], each application specifies a document containing how to interact between a user and an appliance. The device independent layer renders the document for respective existing user interface systems. In multiple user interface approaches, which is the second alternative, each appliance provides the *getUI()* method. The method returns a reference to a multiple user interface support service, and the service offers the most suitable user interface for respective interaction devices. The both approaches are very promising, but they require to modify existing applications like the first approach, and the device independent layers should be implemented for respective interaction devices. Therefore, the approach is not suitable for supporting existing standard specifications for home computing.

In the last approach, the bitmap images that are generated by the user interface system is converted according to the characteristics of interaction devices to control appliances. Also, input events from interaction devices are converted to mouse or keyboard events. The approach is limited not to be able to use the layout information in graphical user interface to convert the input and output events. However, most of audio and visual home appliances provide a display device to show a graphical user interface. In this case, the most important requirement is that we like to change a display device to show the graphical user interface and a input device to navigate the graphical user interface according to a user's situation or preference. The requirement can be achieved very well in this approach. On the other hand, the approach does not require to modify existing software for home computing. Also, most of popular user interface systems such as Java AWT/Swing, the X window system, Microsoft Windows, which have been adopted in traditional home computing standard middleware, can be adopted very easily in this approach.

In this paper, we have chosen the third approach since our home computing system implementing HAVi requires to use Java AWT, and we like to use various home computing applications developed on HAVi in near future without modifying them. We have adopted the VNC system that has been developed by AT&T Labolatories, Cambridge[25]. VNC already supports a variety of popular user interface systems such as the X window system, Microsoft Windows, and Macintosh. Especially, recent interests in the use of embedded Linux to build home appliances make our approach more practical[19] since Linux usually adopts the X window system as its basic user interface system, and our system enables a variety of applications on embedded Linux to be controlled by various interaction devices. The approach makes the development of a variety of embedded applications very easy since the applications can use traditional and familiar user interface toolkits.

3.2 Universal Interaction

In our approach, we call the protocol that can be universally used for communication between input/output interaction devices and appliances *universal interaction*. The output events produced by appliances are converted to *universal output interaction events*, and the events are translated that are appropriate for respective output interaction devices. Also, input events generated in input interaction devices are converted to *universal input interaction events*, and the events are processed by applications executed in the appliances.

A universal interaction proxy that is called the VNC proxy in the next section plays a role to convert between the universal interation protocol and input/output events of respective interaction devices in a generic way. The proxy allows us to use any input/output interaction devices to control appliances if the events of the devices are converted to the universal interaction protocol. This approarch offers the following three very attractive characteristics.

The first characteristic is that input interaction devices and output interaction devices are choosen independently according to a user's situation and preference. For example, a user can select his/her PDA for his/her input/output interaction. Also, the user may choose his/her cellular phone as his/her input interaction device and a television display as his/her output interaction device. For example, a user can control an appliance by his/her gesture by navigating augumented real world generated by wearable devices.

The second characteristic is that our approach enables us to choose suitable input/output interaction devices acoording to a user's preference. Also, these interaction devices are dynamically changed according to the user's current situation. For example, a user who controls an appliance by his/her cellular phone as an input interaction device will change the interaction device to a voice input system because his both hands are busy for other work currently.

The third charatceristic is that any applications executed in appliances can use the any user interface systems if the user interface systems speak the universal interaction protocol. In our approach, we currently adopt keyboard/mouse events as universal input events and bitmap images as universal output events. They enable us to use traditional graphical user interface toolkits such as Java AWT, GTK+, and Qt for interfacing with any interaction devices. For example, a lot of standards for comsumer electronics like to recently adopt Java AWT for their GUI standards. Thus, our approach will allow us to control various future comsumer electronics from various interaction devices without modifying their application programs. The characteristic is very desirable because it is very diffiucult to change existing GUI standards.

3.3 System Architecture

Our system uses the VNC system to transfer bitmap images to draw graphical user interface, and to process mouse/keyboard events for inputs as described in Section 3.1. The VNC system originally consists of a VNC viewer and a VNC server. The VNC server is executed on a machine where an application is running. The application implements graphical user interface by using a traditional user interface system such as the X window system. The bitmap images generated by the user interface system are transmitted to a VNC viewer that are

usually executed on another machine. On the other hand, mouse and keyboard events captured by the VNC viewer are forwarded to the VNC server. The protocol between the VNC viewer and the VNC server are specified as the RFB(Remote Frame Buffer) protocol. The system is usually used to move a user's desktop according to the location of a user[7], or shows multiple desktops on the same display, for instance, both MS-Windows and the X Window system.

In our system, we replace the VNC viewer with a VNC proxy that forwards bitmap images received from a VNC server to an output device. Also, it forwards input events received from an input interaction device to the VNC server.

Our system consists of the following four components as shown in Figure 5. In the following paragraphs, we explain these components in detail.

- Home Appliance Application
- VNC Server
- VNC Proxy
- Input/Output Devices

Home appliance applications generate graphical user interface for currently available home appliances to control them. For example, if TV is currently available, the application generates user interface for the TV. On the other hand, the application generates the composed GUI for TV and VCR if both TV and VCR are currently available. In our system, applications can adopt standard GUI libraries such as Java AWT/Swing, GTK+, or Win32 API running on a variety of popular window systems to write user interface components.

The VNC server transmits bitmap images generated by a window system using the RFB protocol to a VNC proxy. Also, it forwards mouse and keyboard events received from a VNC proxy to the window system. In our current implementation, we need not to modify existing VNC servers, and any applications running on window systems supporting a VNC server can be controlled in our system without modifying them.

The VNC proxy is the most important component in our system. The VNC proxy converts bitmap images received from a VNC server according to the characteristics of output devices. Also, the VNC proxy converts events received from input devices to mouse or keyboard events that are compliant to the RFB protocol. The VNC proxy chooses a currently appropriate input and output interaction devices for controlling appliances. Then, the selected input device transmits an input plug-in module, and the selected output device transmits an output plug-in module to the VNC proxy. The input plug-in module contains a code to translate events received from the input device to mouse or keyboard events. The output plug-in module contains a code to convert bitmap images received from a VNC server to images that can be displayed on the screen of the target output device.

The last component is input and output interaction devices. An input device supports the interaction with a user. The role of an input device is to delivers commands issued by a user to control home appliances. An output device has a display device to show graphical user interface to control appliances.

In our approach, the VNC proxy plays a role to deal with the heterogeneity of interaction devices. Also, it can switch interaction devices according to a user's situation or preference. This makes it possible to personalize the interaction between a user and appliances.

3.4 Implementation of VNC Proxy

The current version of VNC proxy is written in Java, and the implementation contains four modules as shown in Figure 5. The first module is the RFB protocol module that executes the RFB protocol to communicate with a VNC server. The module uses the same module implemented in a VNC viewer, then we do not explain the module in this paper. The second module is the plug-in management module that receives input and output plug-in modules from interaction devices, and dynamically links the modules in the VNC proxy. The third module is the plug-and-play management module that detects currently available input and output interaction devices. The last module is the plug-in migration management module that manages the migration of input and output plug-in modules between interaction devices and a VNC proxy.

3.4.1 Management for Available Interaction Devices

The plug and play management module detects the currently available input and output devices near a VNC proxy. In our system, a unique ID is assigned for each type of input/output devices. The VNC proxy broadcasts beacon messages periodically. In the current prototype, interaction devices are connected via IEEE802.11, Ethernet or infrared networks. When each interaction device receives a beacon message, it replies an acknowledgement message. The acknowledgement message contains the unique ID to identify the device type. If the VNC proxy receives several acknowledgment messages from multiple interaction devices, it chooses one device according to the preference determined by the VNC proxy. Also, when a newly detected device replies an acknowledgement message, the device may be chosen as a currently used interaction device, if the device is more preferable than the currently used device.

When a VNC proxy chooses a new device after the detection of the device, it sends an acknowledgement message before using the device. Then, the VNC proxy sends a terminate message to the device that is used until now. Finally, the VNC proxy waits for receiving a plug-in module from the new device. The preference to select input and output devices is registered in a VNC proxy for each user. If the system cannot detect who likes to use an appliance, a default preference is chosen. Also, each plug-in module supports an event to switch the currently used input and output devices. For example, a user can send a command to change a currently used output device to a VNC proxy. The VNC proxy switches the current output

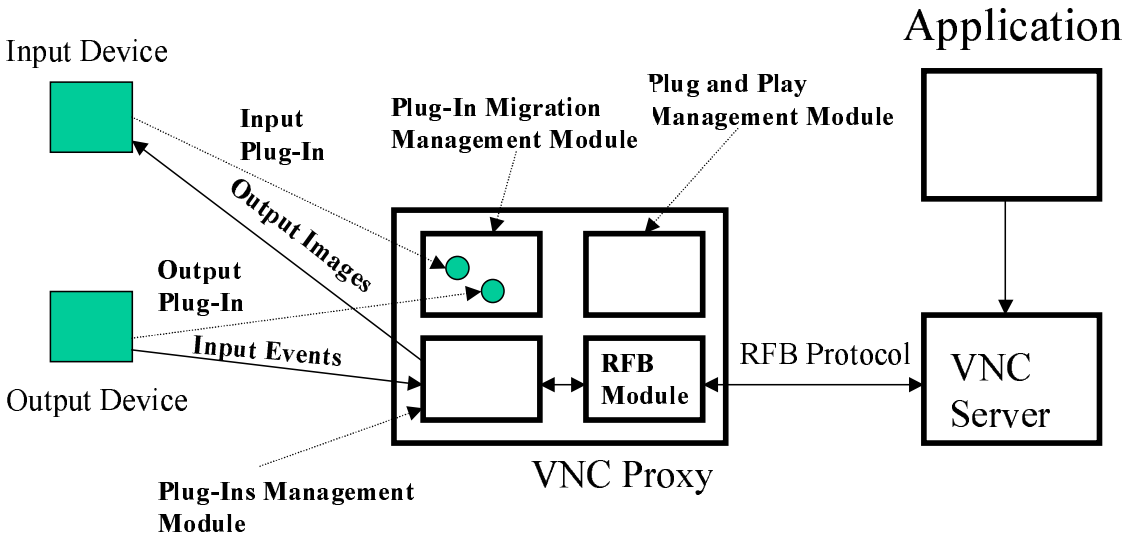

Figure 5: System Architecture

device to the next one until the user selects his favorite output device.

3.4.2 Migration Management of Plug-In Modules

When receiving an acknowledgement message from the VNC proxy, the input/output devices send plug-in modules to the VNC proxy. In our system, we are using the MobileSpaces mobile agent system[26] to transmit plug-in modules between input/output devices and a VNC proxy. After the plug-in modules are downloaded in the Java virtual machine executed in a VNC proxy, the plug-in migration module sends a migration complete message to the input/output devices. MobileSpaces supports hierarchical agents, and the agents can be communicated when they reside at the same hierarchical level. Therefore, we also implement a VNC proxy as a mobile agent, but the agent does not be migrated to other hosts. However, the feature may be used to move a VNC proxy to a near computer from input and output devices.

Currently, we are using a mobile agent system written in Java, but we consider that the assumption to use Java in every device has a limitation since some devices cannot have Java virtual machines due to several limitations. We are working on building a very small runtime written in the C language to send agents to other computers. The runtime does not execute agents, but can transmit the agents to other runtimes. We believe that such a small runtime is desirable to support a variety of interaction devices. Also, the approach allows us to universally adopt Java based mobile agent systems that provide rich functionalities. This means that it may be possible to build a standard mobile agent system that can be used in any devices that have extremely different requirements.

3.4.3 Life Cycle Management of Plug-In Modules

Figure 6 shows a plug-in management module. The module contains an input and an output plug-in modules. The input plug-in module receives events from an input device that is currently selected. The event is converted to a mouse or a keyboard event, and the event is transferred by the RFB protocol to a VNC server. For example, if a user touches a button drawing a right arrow, the event is transmitted to the input plug-in module delivered from a PDA device. The event is translated to a mouse movement event to the right, and the event is finally forwarded to a window system.

Also, after bitmap images are received by the RFB protocol module in a VNC proxy from a VNC server, an output plug-in module processes the images before transmitting to an output device. For example, a color image received from a VNC server is converted to a black and white image. Also, the size of the image is reduced to show on a PDA's screen.

The current version provides two output plug-in modules and three input plug-in modules. The first output plug-in module draws bitmap images on a standard VGA screen, and the second one is for a 3COM's PalmPilot, and the respective three input plug-in modules process events from a keyboard and a mouse, PalmPilot, and a NTT Docomo's i-mode cellular phone that has a Web browser supporting compact HTML[10].

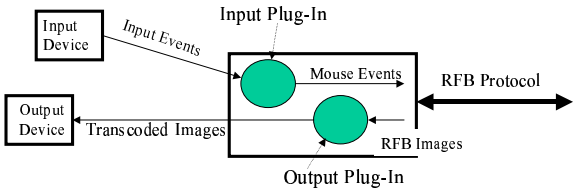

Figure 6: Plug-In Modules

3.5 Context-Awareness in Home Computing Applications

The role of home computing applications is to allow us to control home appliances easily. The interaction between a user and home appliances will become more complex since the functionalities of appliances will be richer and richer. Also, the number of appliances will be increased in future. Therefore, future home applications should support context aware interaction with a variety of appliances.

There are two types of context awareness that should be taken into account. The first one is to personalize the interaction. The interaction is also customized according to a user's situation. The second type is to deal with currently available multiple appliances as one appliance.

It is usually difficult to personalize the interaction with an appliance since a system does not know who controls the appliance. In our system, we assume that each user has his own control device such as a PDA and a cellular phone. If these devices transmit the identification of a user, the system knows who likes to control the appliance. However, in our system, there is no direct way to deliver such information to a home computing application from interaction devices since we assume that the application adopts traditional user interface systems that do not support the identification of a user. Therefore, in our current implementation, each user has a different VNC server that executes personalized applications for each user. The application provides customized user interface according to each user's preference. The VNC proxy chooses an appropriate VNC server according to the identification of a user acquired from an input device.

Our system needs to know which appliances are currently available according to the current situation. In the current system, we assume that an application knows which appliances can be available. For example, if the application supports three home appliances, the application needs to provide seven graphical user interfaces with the combination of the three appliances. The user interfaces are selected according to the currently available appliances. In our system, we assume that each home appliance is connected via IEEE 1394 networks. Since IEEE 1394 networks support a mechanism to tell which appliances are currently connected and whose power switches are turned on, it is easy that the application easily knows the currently available appliances, and selects the most suitable user interface.

3.6 Input and Output Interaction Devices

In our system, a variety of input and output devices are available, and the input devices and output devices may be separated or combined. For example, a graphical user interface can be appeared on the screen of a PDA or a large display device on TV. The user interface appeared on TV can be navigated by a PDA device. Therefore, a user can choose a variety of interaction styles according to his preference. Also, these devices can be changed according to the current situation. For example, when the currently used interaction devices are unavailable, another interaction device may be selected to control appliances.

We assume that each device transmits a plug-in module to a VNC proxy as described in Section 3.3.3. However, some input devices such as a microphone may not be programmable, and it is difficult to support the communication to a VNC proxy. In this case, we connect the device to a personal computer, and the computer communicates with a VNC proxy to deliver a plug-in module. However, it is difficult to know when a program on the personal computer returns an acknowledgement message when a beacon message is received from a VNC proxy since the computer does not understand whether a microphone is currently available or not. Therefore, in the current prototype, we assume that the device is always connected and available.

4 How does Our System Work?

In this section, we describe how our system works. First, we describe a brief overview of our home computing system[22], then we present several components in the system. Finally, we show how our system works in our home computing system.

4.1 Controlling AV Appliances with PDA

When an application recognizes that currently available appliances are a television and a video recorder, it shows a graphical user interface for controlling them. Since the current user does not interested in the TV program reservation, the application draws a user interface containing power control, TV channel selection, and VCR function to its VNC server. As described in Section 3.4, a VNC proxy selects a VNC server executing an application drawing a customized user interface for the user who likes to control these appliances, and the selected VNC server transmits bitmap images containing the interface to the VNC proxy.

Let us assume that the VNC proxy detects that the user has a PDA device. The PDA device delivers input and output plug-in modules to the VNC proxy. The VNC proxy transcodes bitmap images transmitted from the VNC server using the output plug-in module before transmitting the images to the PDA device. In this case, 8 bits color images whose image size is 640x480 are reduced to monochrome images whose size is 180x120. Also, input events on the touch screen of the PDA device are converted to mouse and keyboard events by the input

plug-in module. However, we assume that the user likes to see the graphical user interface on a bigger display now. The user transmits a command to the VNC proxy by tapping on the screen. When the VNC proxy detects it, the bitmap images containing the graphical user interface are forwarded to the display system, and the images are converted by the output plug-in module provided by the bigger display before transmitting them. Also, the user interface will be appeared again on the screen of the PDA device by tapping the PDA's screen.

The scenarios has already implemented in our system. In the system, we have adopted PalmPilot as a PDA device, and a VGA display as a display system.

4.2 Controlling CD Player with Cellular Phone and Voice

In this section, we show a scenario where a user controls a CD player using his cellular phone, but it will be controlled by voice when the cellular phone is turned off.

A VNC proxy makes the input plug-in module for a cellular phone activated if it recognizes that a user has a cellular phone. Also, an output plug-in module is downloaded to use a TV display for showing a graphical user interface. Therefore, bitmap images transmitted from the VNC server are appeared on the TV display. If a user pushes a button on the cellular phone, the event is transferred to the VNC proxy. The VNC proxy converts the event to a mouse movement event, and sends it to a VNC server to simulate the movement of a mouse cursor. When a user pushes another button, the VNC proxy translates it to a mouse click event. Then, the VNC server forwards the event to an appropriate application, and the application recognizes that a button such as a play button is pushed.

Now, let us assume that the user turns off the power switch of his cellular phone. In this case, a voice recognition software on his personal computer is selected as an input interaction device since the user does not carry other currently available input devices. Then, the VNC proxy needs to download the input plug-in module for voice control from the personal computer. The plug-in module translates texts converted by the voice recognition software to mouse and keyboard events to control the CD player.

4.3 Controlling a TV appliance using a Wearable Device

In this example, we like to show that our approach enables us to use advanced wearable devices to control various home appliances. Let us assume that a user wears a head-mounted display that cannot be distinguished from prescription lenses, and he/she wants to control a television. In this case, the graphical user interface of the television is appeared on the glass. The user navigates the graphical user interface via his/her voice.

The VNC proxy converts the image size that is suitable for displaying on the glass. If a user takes off the glass, the graphical user interface is automatically displayed on the display of the television. The voice is used to move a cusor on the graphical user interface. The voice is translated to keyboad/mouse events in the VNC proxy and these events are delivered to an application on a machine that runs with the VNC server.

5 Current Status and Experiences

In this section, we first describe the current status of our prototype system, then we discuss several experiences with building our prototype system to control networked home appliances.

5.1 Current Status

The current prototype in our home computing system emulates two home appliances. The first one is a DV viewer and the second one is a digital TV emulator. Our application shows a graphical user interface according to currently available appliances as described in the previous section. Also, the cursor on a screen that displays a graphical user interface can be moved from a PalmPilot. However, if the device is turned off, the cursor is controlled by the keyboard and the mouse of a personal computer. It is also possible to show a graphical user interface on the PDA device according to a user's preference. Currently, we are working on to integrate cellular phones in our system. NTT Docomo's i-mode phones have Web browsers, and this makes it possible to move a cursor by clicking special links displayed on the cellular phones.

Figure 7 contains several photos to demonstrate our system. In the demonstration, if both a DV camera and a digital TV tuner are simultaneously available, the control panels for them are combined as one panel. As shown in the photos, the control panels can be controlled by both a Palmpilot and a mobile PC.

In our home computing system, we have adopted Linux/RT[29] that extends a standard Linux by adding a resource reservation capability. The Linux also provides an IEEE 1394 device driver and an MPEG2 decoder. Also, IBM JDK1.1.8 for the Java virtual machine is used to execute the HAVi middleware component.

5.2 Experiences

In this section, we show three experiences with building the current prototype of a user interface system for networked audio and visual home appliances.

5.2.1 Limitation of Our System

In our system, a bitmap image that contains a graphical user interface is transferred from a VNC server to a VNC proxy. Since the image does not contain semantic information about its content, the VNC proxy does not understand the content. For example, it is difficult to extract the layout of each GUI component from the image. Therefore, it is not easy to change the layout according to the characteristics of output devices or a user's preference. Also, our system can deal with only mouse and keyboard events. Thus, the navigation of a graphical user interface can be done by emulating the movement of a cursor or pressing a keyboard and mouse buttons. If the

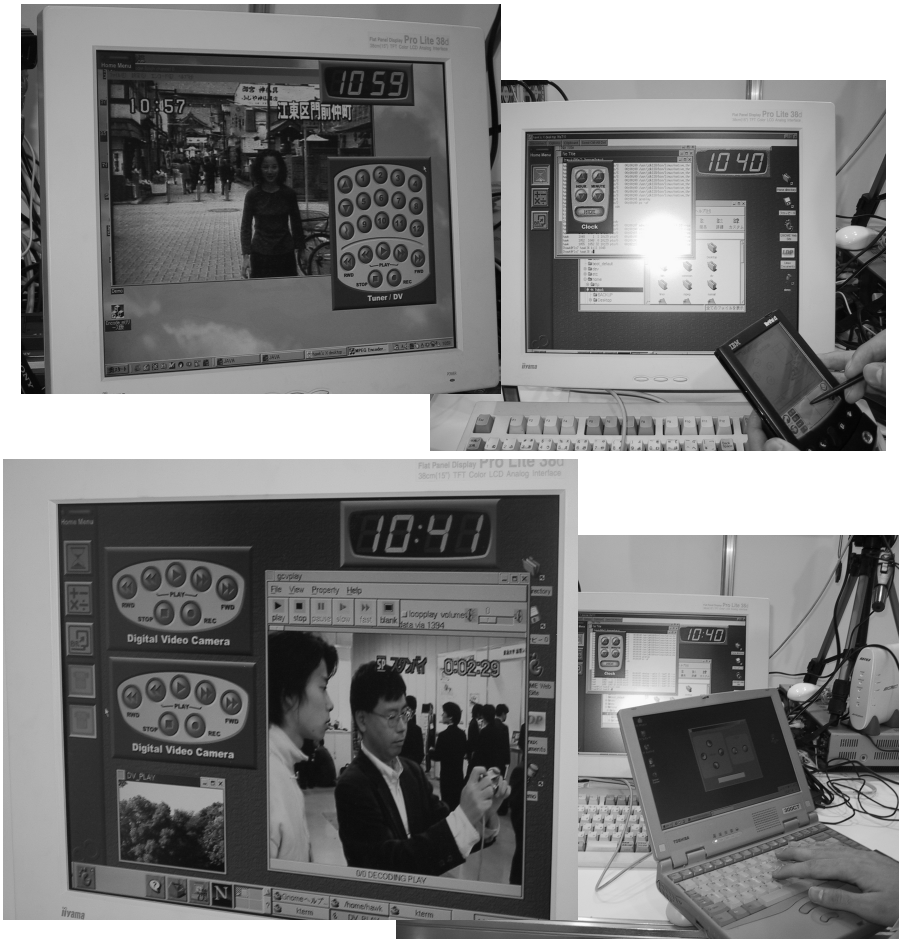

Figure 7: Current Status of Our System

limitation makes the usability of a system bad, other approaches should be chosen. However, navigating a graphical user interface from a PDA and a cellular phone provides very flexible interaction with home appliances. Our experiences show that home appliances usually allow us to use a large display and show graphical user interfaces on the display to control the appliances. Thus, we believe that our system has enough power to make future middleware components for home appliances flexible.

5.2.2 Better Control for Home Appliances

Our system can control any applications executed on standard window systems such as the X window system. In our home computing system, traditional applications coexist with home computing applications, and these applications can also be controlled in an integrated way by our system. For example, we can navigate an MP3 player or a Netscape browser running with home computing applications via our system. However, the overlapping window layout is painful to be navigated by our user interface system. We consider that the tiled window strategy is more suitable for controlling home appliances. Also, our experience shows that we can control both home appliances and traditional applications such as presentation software and web browsers by using our system in a comfortable way if our system supports the movement of a mouse cursor at a variety of speeds.

5.2.3 Importance of Context Awareness

Currently, applications in our home computing system provide very simple functionalities to customize user interface according to context information. Our system monitors which appliances are available using the IEEE 1394's plug and play capability. We also assume that our system needs to know all combinations of available appliances, and design user interface for respective combinations. Then, an appropriate user interface is selected according to the current configuration.

The prototype application is useful to demonstrate the effectiveness of our system. Especially, the composition of functions in respective appliances and the customization of a user interface according to a user's preference are useful for controlling networked home appliances. However, our current design methodology to build context-aware

applications is very ad-hoc, then it is necessary to investigate to build context-aware applications in a systematic way[23], and we need a variety of sensors to monitor a lot of context information such as location information[7] and a user's emotion[24]. Also, we need to investigate a technique to compose multiple functions in appliances in an automatic way even if a new appliance is appeared on an IEEE 1394 network.

For solving the problems, we are working on two issues. The first issue is to build context-aware applications in a systematic way. In our approach, we first create a base program that does not take into account contexts. Then, we add a new concern that deals with a context in an incremental fashion. In the program, a context is considered as a concern, and each concern is clearly separated from a base program. We believe that aspect-oriented programming[18] may help to solve the issue. The second issue is to specify contexts rigorously. In traditional system, the representation of contexts is treated in ad-hoc way. However, to build portable context-aware applications, the representation of contexts should be standardized.

6 Related Work

There are several approaches that are very close to our research efforts. In this section, we describe five systems, and compare the systems with our system.

The first system is the Pebble system[20]. The Pebble system enables us to control desktop applications on the MS-Windows operating system through PalmPilot which is the most popular PDA in the world. For example, a cursor on the desktop can be moved by touching a screen of the PalmPilot. The system is very close to our current prototype. However, the focus is different since Pebble focuses on the usability of a system. On the other hand, our system focuses on the system architecture. Our system enables us to use a variety of input devices to navigate a graphical user interface on an output display. Also, in our system, input and output interaction devices can be switched according to a user's preference. We think that the flexibility of our system is more suitable to support networked home appliances.

The second system is the UIML(User Interface Markup Language) system[31], which is an XML language that permits a declarative description of a user interface in a highly device-independent way. If an application writes a user interface as a UIML document, the document is rendered according to respective input/output interaction devices. For example, a UIML document is rendered on PalmPilot to use it as a input/output devices. Also, a UIML document can be rendered for VoiceXML[32] to support voice interaction. However, it is difficult to adopt the approach when an input and an output devices are separated. Also, there is no support to dynamically switch these input/output devices according to a user's situations. Since the approach is promising in future to support heterogeneous devices running on a variety of user interface systems, there are similar document based approaches described in [21, 9].

The CUES system[16] provides a framework to control a variety of appliances from a mobile device. In the system, each appliance has a Java bytecode that is transmitted to a mobile device. The code contains a graphical user interface, and it is appeared on the mobile computer. A user can control the appliance by the graphical user interface on the mobile computer. The approach enables us to use an appropriate graphical user interface for respective appliances. However, the approach assumes that the mobile computer has a medium size display to show a graphical user interface implemented by Java AWT/Swing, and should have a pointing device and a keyboard. Also, the approach does not support the customization of user interface and dynamic switching of interaction devices.

In [15], they have proposed Simja that is a middleware component supporting multi-modal interfaces to services. Simja translates a variety of media formats. Each translator contains a single function, and it can be connected to another translator. In Simja, the connection is called a path, and the path is set up among translators automatically according to the specification representing the requirements. In our current approach, the structure of plug-in modules is monolithic, and each interaction device needs to create an input and an output module independently, so that Simja's approach is useful to build a plug-in module in a composable way.

HAVi provides two ways to support the interaction with a user. The first way is *Data Driven Interaction*(DDI). This provides a declarative way to describe graphical user interface. In the approach, user interface is described as a document like UIML, and the document is rendered according to the characteristics of a display device, but UIML is more general than HAVi's DDI since UIML is based on XML. The second way is similar to the CUE system. The Java bytecode containing a user interface written in Java AWT is downloaded in a HAVi device, and the graphical user interface is appeared on the display of the device. The problem of the approach is also the same as CUES.

7 Conclusion and Future Work

This paper has described a new user interface system to fill the gap between traditional graphical user interface systems and advanced input/output interaction devices for networked home computing. We have also described the effectiveness of our system by demonstrating our system to control our home computing system.

Our system does not analyze the content of bitmap images containing graphical user interface. Therefore, it is not easy to use our system unless bitmap images are translated for output devices, or input events are translated to mouse or keyboard events, but we believe that our system is enough to control networked audio and visual home appliances since these appliances are usually used with a large display device. Also, our system can be used to control a variety of applications running on Windows and Linux like Pebble.

The other problem is how to select a suitable interaction device according to a user's preference. A user

likes to control a variety of appliances in a uniform way whether he is in offices, houses, stations, or airports. In the current environments, a user needs to take into account where he likes to control appliances due to a variety of physical limitations, but such mode changes caused by user mobility impose a severe burden on his mind. Thus, we require more intimate user interaction regardless of his location or situation, and the interface will connect a variety of our life spaces in a seamless way.

Another future topic is a user interface system in personal area networks. This is another typical example of ubiquitous computing environments. However, our approach is not appropriate for personal area networks since there is no large display in these environments. We also like to investigate simple user interface systems for the environments in future.

References

[1] 1394 Trade Association, "AV/C Digital Interface Command Set General Specification, Version 2.0.1", http://www.1394TA.org/, 1998.

[2] G.D. Abowd, E.D. Mynatt, "Charting Past, Present, and Future Research in Ubiquitous Computing", ACM Transaction on Computer-Human Interaction, 2000.

[3] A.K. Dey, D.Salber, M.Futakawa, G.Abowd, "An Architecture To Support Context-Aware Applications", GVU Technical Report GIT-GVU-99-23. June 1999.

[4] W. Ark, D. Christopher Dryer, D.J. Lu, "The Emotion Mouse", In Proceeding of the HCI International conference, 1999.

[5] G.Banavar, J.Beck, E.Gluzberg, J.Munson, J.Sussman, D.Zukowski, "Challenges: An Application Model for Pervasive Computing", In Proceedings of the Six Annual International Conference on Mobile Computing and Networking, 2000.

[6] K.Cheverst, N.Davies, K.Mitchell, A.Friday, "Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project", In Proceedings of the Six Annual International Conference on Mobile Computing and Networking, 2000.

[7] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, Paul Webster, "The Anatomy of a Context-Aware Application", In Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, 1999.

[8] HAVi Consortium, "HAVi Specification: Specification of the Home Audio/Video Interoperability(HAVi) Architecture, http://www.havi.org/

[9] T. Hodes, and R. H. Katz, "A Document-based Framework for Internet Application Control", In Proceedings of the Second USENIX Symposium on Internet Technologies and Systems, 1999.

[10] T.Kamada, "Compact HTML for Small Information Appliances", W3C Submission, http://www.w3c.org/TR/1998/NOTE-compactHTML-19980209.

[11] N. Gershenfeld, "When Things Start to Think", Owl Books, 2000.

[12] Andy Hopper, "Sentient Computing", The Royal Society Clifford Paterson Lecture, 1999.

[13] H. Ishii, B.Ullmer, "Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms", In Proceedings of Conference on Human Factors in Computing Systems,1997.

[14] Jini Technology, http://www.jini.org/

[15] A.Joseph, B.Hohlt, R.Katz, and E.Kiciman, "System Support for Multi-Modal Information Access and Device Control", WMCSA '99, 1999.

[16] Kangas, K., and Roning, J., "Using Code Mobility to Create Ubiquitous and Active Augmented Reality in Mobile Computing", In Proceedings of the 5th ACM/IEEE International Conference on Mobile Computing and Networking, 1999.

[17] N.Khotake, J.Rekimoto and Y.Anzai, "InfoStick: an interaction device for Inter-Appliance Computing", Workshop on Handheld and Ubiquitous Computing (HUC'99), 1999.

[18] G. Kiczales, et. al., "Aspect Oriented Programming", Xerox PARC Technical Report, SPL-97-008, 1997.

[19] R. Lehrbaum, "The Embedded Linux Market", http://www.linuxdevices.com/

[20] Brad A. Mayer, Herb Stiel, and Robert Gargiulo, "Collaboration Using Multiple PDAs Connected to a PC", In Proceedings CSCW'98: ACM Conference on Computer-Supported Cooperative Work, 1998.

[21] M. Munson, T. Hodes, T. Fischer, K. H. Lee, T. Lehman, B. Zhao, "Flexible Internetworking of Devices and Controls", In Proceedings of IECON, 1999.

[22] T. Nakajima, "System Support for Networked Audio and Visual Home Appliances on Commodity Operating Systems", Submitted for Publication.

[23] T. Nakajima, A Framework for Building Environment-Aware Software, The second IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 1999.

[24] R. Picard, "Affective Computing", The MIT Press, 1997.

[25] T.Richardson, et al., "Virtual Network Computing", IEEE Internet Computing, Vol.2, No.1, 1998.

[26] I. Sato, "MobileSpaces: A Framework for Building Adaptive Distributed Applications using a Hierarchical Mobile Agent System", In Proceedings of IEEE International Conference on Distributed Computing Systems, 2000.

[27] I.Siio, T.Masui, K.Fukuchi, "Real-world Interaction using the FieldMouse", In Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'99), 1999.

[28] R.Lea, S.Gibbs, A.Dara-Abrams, E. Eytchson, "Networking Home Entertainment Devices with HAVi", IEEE Computer, Vol.33, No.9, 2000.

[29] Timesys, "Linux/RT", http://www.timesys.com.

[30] Universal Plug and Play Forum, http://www.upnp.org/

[31] User Interface Markup Language, http://www.uiml.org/

[32] VoiceXML Forum, "VoiceXML Forum Version 1.0 Specification", http://voicexml.org/.

[33] R.Want, B.Schilit, N.Adams, R.Gold, K.Petersen, J.Ellis, D.Goldberg, M.Weiser, "The ParcTab Uniquitous Computing Experiment", Technical Report CSL-95-1, Xerox Palo Alto Research Center, 1995.

[34] Mark Weiser, "The Computer for the 21st Century", Scientific American, Vol. 265, No.3, 1991.