Connecting Object-Oriented Middleware for Home Computing with Virtual Overlay Netw orks

Daiki Ueno, Eiji T okunaga, Hiro Ishikawa, Tatsuo Nakajima
Departmen t of Information and Computer Science
W aseda University
3-4-1 Okubo Shinjuku T okyo 169-8555 JAPAN
tatsuo@mn.w aseda.ac.jp
Ichiro Satoh
National Institute of Informatics /
Japan Science and Technology Corporation

Abstract

The paper proposes an approach to solve the situation. In our approach, we have developed a middleware component to connect several middleware implementing these object-oriented standard specifications for home computing. Actually, our middleware component makes it possible to connect home appliances adopting HAVi, Jini, or UPnP. Therefore, various home appliances can comm unicate with each other.

1 Introduction

Our future home will have a lot of home appliances such as TV and VCR, and these appliances will be connected with each other. Several standard specifications for home computing ha ve been developed recently, and a lot of people believe that future home appliances will implement these specifications. However, for example, Jini has been developed several years ago, and we had thought that several real appliances implementing Jini appear soon. Wh y aren't these appliances available now ? One of the reasons is that if an appliance chooses one standard specification, it is not easy to comm unicate with other appliances implementing different standard specifications.

There are several approaches to solve the problem. For example, a Jini and HAVi gateway has been developed to connect Jini and HAVi devices. However, the approach is very ad-hoc, and many gateways should be implemen ted to connect different specifications. W e need a more systematic approach to solve the connectivity problem.

This paper proposes a middleware component to connect various appliances implemen ting standard specifications for home computing in a systematic w ay. Actually, we have developed a middleware component connecting HAVi[2], Jini[4], and UPnP[3]. Our approach adopts a virtual overlay network using the HTTP protocol, which builds an application specific network on the Internet.

2 Design and Implementation

A key goal motivating the design of a virtual overlay network is to provide an application-specific abstraction for home appliances overlaid on heterogeneous networks such as IEEE 1394, Bluetooth, and the Internet. In this section, we describe the design and implemen tation of our current prototype system of the virtual overlay network. First, we present design issues and the system overview of our system, then we show some components that we have currently implemen ted. The components show the effectiveness of our approach.

2.1 Design Issues

The end-to-end argument is a system design principle in the Internet. That is, sophisticated network functions are assumed to be pushed to the edge of the network. However, all the edges in the current Internet are sophisticated computing devices, because some of them are designed for only their own special purposes, for example home appliances, embedded computers, and PDAs, and thus may not handle standard protocols in the Internet. Therefore, some enrich network functions should be supported or enhanced inside the network, rather than such stupid edges, and provide a seamless perspective for all edges and sub-networks, even when some of the edges and sub-networks are not ready for the Internet. Our virtual overlay network framework supports the following further functions.

- The virtual overlay network framework is built around a new naming system. W e believe that appliances should be looked up according to their intentional information such as functionality and physical locations, rather than to things like hardcoding hostnames to addresses. Indeed, some appliances, whic h do not support TCP/IP, cannot have any unique addresses available in the Internet and thus are required to be encapsulated by and identified through other servers, which can directly connected with the appliances.

Proceedings of the Fifth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'02)
0-7695-1558-4/02 $17.00 © 2002 IEEE

- Web-based computing is performed through HTTP and widely used in the current Internet. HTTP is the protocol that underlies the World Wide Web, and so any web browser, by definition, can communicate using it. This communication occurs by the browser using URLs that conform to a predetermined naming convention. Our virtual overlay network framework should easily and naturally control access to, management of, corporate various home appliances through an ordinary web browser. Therefore, it needs to enable home appliances to work together within the Web space and to be specified at URLs.

- Some home appliances may often be managed under existing service discovery systems for managing devices and services such as Jini and UPnP. These systems are proposed to facilitate dynamic cooperation among devices/services with minimal administration and human intervention. In order to be able to support the impromptu community, they should provide the means to announce its presence to the network, to discover services in the neighborhood, and to access to services. Our virtual overlay network framework offers a unified perspective for not only appliances but also existing service discovery systems for managing appliances.

- The virtual overlay network framework should make extensive use of transcoding to perform transformations on data in the network. This is because some home appliances often have serious limitations in their computational resources such as processors, and output and input devices. They may not be able to directly handle enriched data, which they should receive from and send to the network, without any assists for transcoding that data from external systems.

2.2 Architecture

The virtual overlay network framework consists of home appliances/services and application-level gateways, where some appliances are just application-specific devices and thus do not support TCP/IP. Each application-level gateway is implemented as a HTTP-based proxy server, and can control more than one appliance, with which it can communicate directly. Also, the gateway can be treated as an access point which interfaces to more than one appliance in an access network and provides a translation service from HTTP to the corresponding commands of its target appliances. Also, it can export a generic session interface of various appliances to the Internet. That is, it acts as a client when it establishes sessions to the Internet and as a server when it accepts incoming sessions from the Internet. Also, the application-level gateway can support application-specific services, for example naming, transcoding, and packet forwarding. A variety of home appliances can be connected to application-level gateways and the ways of controlling these appliances are not unified yet. Therefore, we introduce the notion of extensibility to our gateways to handle the variety and changes of appliances. An application-level gateway consists of two layers. The top layer consists of the functionality to satisfy HTTP requests and various modules that provide the logic to perform web based tasks such as authentication, and so on. When it receives an HTTP request from a browser, one of the most appropriate modules is selected to serve the request. This modular approach means that it is possible to install additional modules to extend the gateway's functionality. The lower layer uses our framework for accessing to the actual appliance and supporting services.

2.3 Accessing Home Appliances from Web browsers

Although our final goal is to offer URL based interfaces to arbitrary applications and devices, we cannot currently turn all network appliances into HTTP servers. Instead, each application-level gateway controls more than one appliance through their particular networks by offering a translation service from URL-based expressions to corresponding commands designed for the appliances. Typically, the web-browser accesses remote hosts identified in a URI by submitting a query request using the standard HTTP GET mechanism, and receives the response from the URI, or submitting data to the URI via an HTML form using the standard HTTP POST mechanism, and receives the response from the URI also in HTTP. The gateway then carries out the necessary operations or queries with the home appliance. When the gateway has finished talking to the appliance, the resulting data is passed back to the browser.

```
http://some.where.com:8080/CEIL-LIGHT/!power=ON
```

The above URL is a typical example of our naming convention. It invokes the gateway named as `some.where.com`, which is listening on port 8080 by prior agreement. The `CEIL-LIGHT` element specifies an electric light controlled under the gateway. The `!power=ON` element signals to the gateway and nominates a specific function, called `power` provided with the `ON` value by the appliance. The URL is a request to turn on the electric light named as `CEIL-LIGHT`.

2.4 Integrating Home Networks

In this section, we describe how to connect several home networks that provide different home network protocols by using our virtual overlay networks. As shown in Figure 1, we assume that there are three home networks that are connected to the Internet via respective application-level gateways. These home networks respectively support UPnP, HAVi, and Jini protocols.

Application-level gateways in our virtual overlay networks consist four components. The first component implements home network protocols such as HAVi, Jini, or UPnP. The component behaves as one of home network devices. For example, if a component implements the Jini protocol, it behaves as a Jini device. The second component implements the HTTP protocol. The component behaves as a Web server.

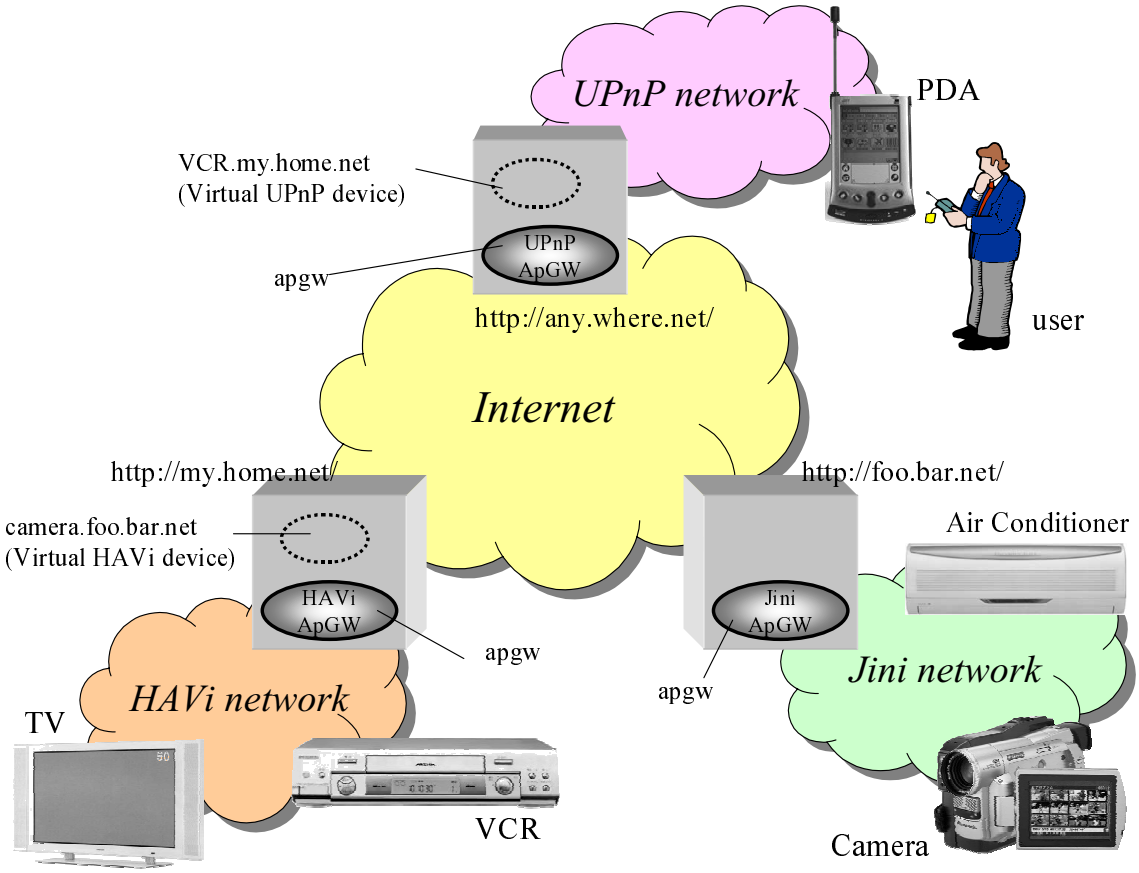

Figure 1: Connecting Home Networks

The third component implements a protocol translator. The translator converts between the HTTP transaction described in the previous section and home network protocols. For example, the HAVi translator converts both from the HTTP protocol to the HAVi protocol and from the HAVi protocol to the HTTP protocol. We have also implemented the Jini translator and the UPnP translator in our current prototype implementation.

The last component is the registration manager. The registration manager can be accessed by the HTTP protocol from a Web browser or a home network protocol from some appliances that implement the protocol. If an appliance on a home network likes to be accessed from my home network, the name of the appliance that we like to access should be registered on my application-level gateway. The name can be registered via either the HTTP protocol or a home network protocol. After the registration, a new pseudo home network device is created in the application-level gateway. The pseudo device registers its name in a lookup service. The role of the pseudo device is to covert control commands that are delivered on the home network to the HTTP protocol.

For example, let us assume that we like to control a VCR device on a HAVi network from a PDA connecting to a UPnP network. In the HAVi network, the VCR device is registered as "VCR", and its application-level gateway(HAVi ApGW) is registered as "apgw" in the HAVi lookup service. In the UPnP network, the PDA devices is registered as "PDA" and its application-level gateway(UPnP ApGW) is registered as "apgw" in the UPnP lookup service.

Now, we like to register the HAVi VCR device in the UPnP network for controlling the HAVi VCR device from the UPnP PDA device. From the PDA device, a user registers the HAVi VCR device in UPnP ApGW as a pair of "VCR" and "http://my.home.net/VCR" from the Web browser on the PDA device. UPnP ApGW creates a pseudo UPnP device on UPnP ApGW, and registers "VCR" in the UPnP lookup service. When the PDA device likes to control the HAVi VCR device, the PDA device tries to find the VCR device via the UPnP lookup service. The lookup service returns a reference to the pseudo device that is created on UPnP ApGW. Then, a control command is delivered to the pseudo device via the UPnP protocol. The pseudo device converts the UPnP request to the HTTP protocol, and forwards

the request to HAVi ApGW according to the host name part of "http://my.home.net/VCR/". HAVi ApGW receives the HTTP request, and finds the VCR device with the name "VCR" that is specified in the file name part of the URL via the HAVi lookup service. Finally, the command is converted to the HAVi command, and it is delivered to the target VCR device.

Also, let us assume that we like to control a Jini camera device from a HAVi TV device. In this case, a user registers a pair of "Camera" and "http://foo.bar.net/Camera/" on HAVi ApGW using the HAVi protocol, where the TV device finds HAVi ApGW via the HAVi lookup service with the name "apgw". Similar to the previous example, a HAVi command is converted to a HTTP command at HAVi ApGW, and the HTTP command is converted to a Jini command at Jini ApGW.

There is a proposal to connect HAVi and Jini via the HAVi-Jini bridge. However, the approach needs to implement a lot of bridges if there are several home network protocols. Moreover, when a new home network is proposed, it is not realistic to create a lot of bridges to convert among protocols. In our approach, a new home network protocol requires to implement a converter to translate between the HTTP protocol and the new home network protocol.

2.5 Registering Remote Devices

One of the most serious problems of the current architecture is the registration of remote devices. When a user moves to other people's homes, it is necessary to register the name of an appliance to use it from the other people's homes. For example, if I like to access my television from my friend's home, my television should be registered to the gateway server of my friend's home. It seems that the approach is not practical. Moreover, if a user' home requires to use multiple home network protocols, it is difficult to use appliances that support different home network protocols.

Our solution to solve the problem is to use a personal device to make the registration easy. In our approach, each appliance returns a URL when a personal device sends a request. Also, the personal device can transmit the URL to a gateway server when a user visits to other people's homes. In our system, we are using infrared to receive/transmit URLs. We believe that our approach is very practical and makes the registration of appliances very easy. Also, the approach can be used as a generic method to move a small amount of information between two appliances.

3 Current Status

Currently, we have built a prototype system implementing the proposed architecture. The prototype system has implemented in Java, and is running on Linux.

The system contains the following components.

- Jini-based client program
- UPnP-based client program
- HAVi-based client program
- Jini-based TV control system
- UPnP-based Light control system
- UPnP-based MP3 player
- HAVi-based digital camera system
- Jini-base gateway
- UPnP-based gateway
- HAVi-based gateway

The following system allows us to control the UPnP-based Light control system, Jini-based TV control system, and HAVi-based digital camera from any client systems that supports UPnP, Jini or HAVi. Also, these systems can be controlled from a standard Web browser. Figure ?? shows our current prototype system.

4 Conclusion

In this paper, we have proposed a way to connect object-oriented middleware for home computing with virtual overlay networks. The virtual overlay network determines the routes to deliver messages and converts protocols and messages according to the characteristics of networks and appliances. We have described several scenarios to show the effectiveness of the proposed virtual overlay network and have presented the implementation of the current prototype system. The detailed description of our work can be found in [1]

References

[1] T. Nakajima, H.Aizu, I.Sato, D.Ueno, "A Virtual Overlay Network for Integrating Home Appliances", In Proceedings of the International Symposium on Applications and the Internet, 2002.

[2] HAVi Consortium, "HAVi Home Page", http://www.havi.org/.

[3] Univeral Plug and Play Forum, http://www.upnp.org/

[4] Jim Waldo, "Alive and Well: Jini Technology Today", IEEE Computer, Vol.33, No.6, 2000.