View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Waseda University Repository

A Virtual Overlay Network for Integrating Home Appliances

Tatsuo Nakajima,
Daiki Ueno, Eiji Tokunaga, Hiro Ishikawa
Department of Information and Computer Science
Waseda University
3-4-1 Okubo, Shinjuku, Tokyo 169-8555, Japan
tatsuo@mn.waseda.ac.jp

Ichiro Satoh
National Institute of Informatics /
Japan Science and Technology Corporation
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
ichiro@nii.ac.jp

Hiroyuki Aizu
Toshiba Corporation
1, Komukai Toshiba-Cho, Saiwai-ku, Kawasaki 212-8852, Japan
hiroyuki.aizu@toshiba.co.jp

Abstract

Various home appliances and personal appliances are expected to be connected to the Internet in the near future, and the number of such appliances will become extremely large. The Internet will provide connectivity between nodes running various services and applications, in order to provide new advanced functionality. However, the conventional IP protocol is not expected to be sufficient for connecting these appliances and services, because these future appliances are likely to require their own standardized protocols such as Jini and HAVi. Thus, connectivity cannot be achieved without protocol translation. Also, these appliances may require their own naming scheme, routing algorithm, and mobility/multicast supports. Therefore, it is preferable for these functionalities to be customized according to their respective applications. Since adding new functionalities to the IP protocol is too generic for most applications, we believe that future networks should be customized according to the characteristics of the respective applications.

In this paper, we propose a virtual overlay network for integrating networked home appliances. Virtual overlay networks enable new application-specific networks to be built on existing networks. The proposed virtual overlay network is designed to access a variety of appliances on home networks from personal or home appliances on other home networks. The paper describes the motivation of our work, and the design and the current status of the proposed virtual overlay network.

We think that the approach proposed in the paper will also be useful for building virtual overlay networks for other applications such as continuous media streaming or peer-to-peer storage systems.

1 Introduction

Various home appliances such as TVs and VCRs are connected to home networks, and all of the home networks can be integrated by the Internet to exchange various information or to control appliances from other appliances. The future Internet will be able to connect an extremely large number of appliances, and so these appliances should be integrated as one appliance from the viewpoint of the user. This kind of integration is likely to be very important in the near future since an extremely large number of appliances are expected to connected to the Internet.

A computing environment where a variety of appliances are connected with one another, is called an *ubiquitous computing environment* [31]. The traditional idea of an ubiquitous computing environment is somewhat limited in scale when compared to the Internet, however, and so it is necessary to consider some additional issues when trying to design *Internet-scale ubiquitous computing environments*[21]. In this paper, we discuss the issues that arise when considering such environments.

A variety of networks and protocols have been proposed for respective application domains. For example, various protocols such as Jini[30] and UPnP(Universal Plug and Play)[28] have been proposed for connecting various appliances without their configuration before using them. HAVi[12, 17] has been proposed for connecting various audio and video home appliances. On the other hand, various network systems such as ATM(Asynchronous Transfer Mode) networks, IEEE 1394[13], Bluetooth[18], and VIA[7] have been developed to connect a variety of appliances. Also, in the research community, PEN[16, 8], networked surface[25], and CLAN[23] have been proposed for connecting future advanced appliances.

However, various useful characteristics of these underlying protocols and networks are hidden from applications if an IP layer is inserted on the networks. For example, the plug and play functionality provided by Bluetooth and the network bandwidth reservation functionality provided by ATM are not usually available on the top of the IP layer[1]. Also, some appliances do not support the IP layer for connecting to other appliances and services. Moreover, each appliance may assume different control protocols and data formats to communicate each other. Therefore, a new approach is needed to realize network supports for future home appliances.

In this paper, we propose a virtual overlay network for integrating networked home appliances. The concept of virtual overlay networks is not new. It has been used in the past to solve various problems on the Internet. However, there has been no previous work on applying virtual overlay networks to the integration of networked home appliances. Our approach is designed to fully exploit the concept of virtual overlay networks. In addition, the aim of our research is to build customized virtual overlay networks for establishing *Internet-scale ubiquitous computing environments* in a systematic way.

The remainder of this paper is structured as follows. In Section 2, we explain what a virtual overlay network is, and how virtual overlay networks solve problems to which the current Internet is subject. Section 3 presents the design and implementation of a virtual overlay network for integrating home appliances. In Section 4, we discuss an application in which environment information is used. In Section 5, we describe related work, and Section 6 concludes this paper.

2 Virtual Overlay Networks

As mentioned in the previous section, an extremely large number of appliances are expected to be connected to the Internet. A mobile appliance may move with the person carrying it. Also, some information may be delivered to a large number of appliances simultaneously. For solving the issues respecting the Internet, various protocols such as Mobile IP[22] and IP Multicast[24] have been proposed. However, these protocols are not widely available now although they were first proposed a long time ago[26]. We believe that these functionalities such as mobility and multicast support should be tailored to respective application domains, and not be employed to provide generic supports that can be used for most applications. Similarly, other protocols such as DHCP, NAT and IPv6 will partially solve the problems previously mentioned, but we believe that the problems should be solved on the top of the IP layer, and not by offering generic solutions by extending the IP protocol.

An overlay network can be defined as a configuration within which a base network is used to support some upper level networks, and it provides a sufficient set of base properties for realizing virtual overlay networks. A virtual overlay network extends the base properties of overlay networks by adding a software layer to provide new abstract properties. The implementation of the properties on behalf of the user is customized for respective applications. The idea seems to be similar to that of *active networks*[27]. Active networks enable customization of existing networks

[1]There are several proposals to extend the IP protocol to support QOS guarantee, but the abstraction provided by the proposals does not offer the full power of the ATM's bandwidth reservation capability.

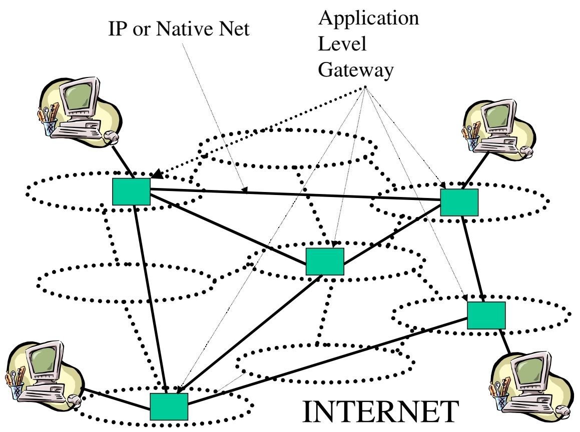

Figure 1. Virtual Overlay Network

so that application-specific codes can be injected without modifying existing properties provided by networks. On the other hand, virtual overlay networks add new abstract properties that are suitable for respective applications, and do not extend existing networks. We believe that virtual overlay networks are more practical because they do not require modification of existing network infrastructures, and they are easily customized for respective applications.

In this paper, we define a virtual overlay network as a network that is built on existing networks such as the Internet as shown in Figure 1. In virtual overlay networks, a lot of application-level gateways are installed in the network, and they are connected by some network protocols such as the IP protocol. The role of the application-level gateway is to route messages to destinations. Also, messages and protocols may be translated according to various requirements offered by environments and users. The application-level gateways are considered as routers of virtual overlay networks. They are usually connected by the Internet, but underlying datalink networks can be directly used if the communication between application-level gateways requires satisfaction of special QOS requirements.

Virtual overlay networks will solve a wide range of problems to which the current Internet is subject. For example, application-specific naming can be adopted for routing messages although the existing Internet has adopted FQDN(Fully Qualified Domain Name) and the IP address. Also, connections among application-level gateways are tailored according to respective QOS requirements between the gateways by using native networks directly or by choosing the configuration of application-level gateways carefully. On the other hand, it is very difficult to realize end-to-end QOS guarantee in the Internet. By choosing the functionalities of application-level gateways, virtual overlay networks can be customized for respective applications. In view of our experience in building mobility supports[20], we think that solutions for realizing high level functionalities should be customized for respective applications.

As mentioned in Section 1, the concept of virtual overlay networks is not new. There are some proposals concerning the building of virtual overlay networks for several applications. In [14], a virtual overlay network for multicasting continuous media data such as video and audio has been proposed. In [15, 6], virtual overlay networks for reliable multicasting have been proposed. In [5], a virtual overlay network for a global event service has been proposed. In [3], a router partition mechanism that is used to divide resources for respective virtual overlay networks has been proposed. The solutions are not generic for satisfying all requirements of the Internet, but they are sufficient to solve the problems of respective domains.

In traditional networks, the concept of virtual overlay networks has been partially adopted. For example, MBone[24], a multicast-capable IP network on the current Internet, has been built, and 6Bone[11] has been used to develop an IPv6 network on the current Internet. Moreover, Domain Naming System(DNS)[19] and web caching systems[32] are also considered to be virtual overlay networks, and they have solved various problems in an application-specific manner.

The above proposals have greatly increased scalability by adopting the concept of virtual overlay networks, but these proposals do not use native underlying data link networks directly and protocol and message translation explicitly. In our system, we endeavor to fully exploit the concept of virtual overlay networks.

One of the most important aims of our research is to develop virtual overlay networks customized for each application in a systematic way. However, achievement of this goal is predicated on our gaining a great deal of experience. Therefore, the present paper, concerns the building of a virtual overlay network for integrating networked home appliances, work undertaken in order to gain experiences. Networked home appliances provide a variety of protocols to control these appliances. For example, Jini, UPnP, HAVi, and SOAP are provided to control different home appliances. A client appliance cannot control these home appliances unless it supports all the protocols. However, the assumption is unrealistic because supporting multiple protocols requires large memory footprint. Also, new software should be added in the client if a new protocol is proposed for future home appliances.

Virtual overlay networks solve the problem by converting a protocol supported by a client appliance to the protocol supported by a home appliance. The conversion is executed by respective application-level gateways according to the characteristics of appliances and networks. The next section discusses the effectiveness of virtual overlay networks for integrating networked home appliances in detail, and propose a network architecture for integrating home appliances in the Internet.

3 Design and Implementation

A key goal motivating the design of our 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 implementation of our current prototype virtual overlay network. First, we present design issues and an overview of our system, and then we show some components that we have already implemented. The components show the effectiveness of our approach. Lastly, we discuss some problems of the current prototype system.

3.1 Design Issues

The end-to-end argument is a system design principle for 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 not 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 enriched network functions should be supported or enhanced inside the network, rather than at such unsophisticated 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 additional functions.

- The virtual overlay network framework is built around a new naming system. We believe that appliances should be looked up according to their intentional information such as functionality and physical locations, rather than according to attributes such as hard-coding hostnames to addresses. Indeed, some appliances, which 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 connect with the appliances.

- 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. Not only does our virtual overlay network allow easy access and control of various home appliances from an web-brower, but it also enables appliances to communicate with each other and work in cooporation.

- 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 services. Our virtual overlay network framework offers a unified perspective for not only appliances but also for 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 assistance for transcoding that data from external systems.

IEEE COMPUTER SOCIETY

3.2 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. Therefore, an application-level gateway needs to have a protocol translation function from appliance specific protocol to HTTP (and vice versa). In our proposal, each application-level gateway is implemented as an HTTP-based proxy server. It accepts an HTTP-based home appliance control protocol, and translates it into an appliance-specific protocol. Also, we propose to use the gateway as an access point from the appliance-specific protocol to the Internet. It provides a translation service from HTTP to the corresponding commands of its target appliances. Also, it exports a generic session interface of various appliances from/to the Internet.

Also, the application-level gateway should support application-specific services, for example naming, transcoding, and packet forwarding, to various kinds of appliances. Therefore, we introduce extensibility to our gateways. 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, appliance control, and so on. 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 the actual appliance and supporting services.

Another point we should consider is appliance controller. The controller sends the HTTP-based appliance control request to the gateway. We can use a legacy web browser for the controller, but with some modification to the browser, some new functions can be provided.

3.3 Access to Application-level Gateway

We have implemented several components that show the effectiveness of our approach. In this section, we describe the components by using two scenarios. The first scenario is to access a home appliance from a web browser, and the second example is to connect two home networks that use different protocols. Although our final goal is to offer URL [4] 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.

Attributes for URL: Our framework allows the user (or other gateways) to access one or more appliances through a few gateways given as HTTP-based proxy servers. However, it is difficult for the user to identify one of the most appropriate appliances among them, even when the addresses of gateways are known. Therefore, we introduce a URL-based naming convention for specifying and controlling appliances. The convention is defined within the standard URL but the path elements of the URL form can contain some additional information according to the BNF-like syntax shown in Figure 2:
where ϵ is an empty string and <string> corresponds to a sequence of alphabetical characters. 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.

```
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 `CEIL-LIGHT`. We show another example below:

```
http://some.where.com/?location=
livingroom&?function=light/!power=ON
```

where the `?location=livingroom` element specifies the location called `livingroom` in a building or floor controlled under the gateway server. The `?function=light` element specifies, one of the most suitable appliances for supporting the functionality named `light` in the `livingroom` location.

Variables for URL: To specify such suitable appliances dynamically, our URLs can contain the form `$(variable)`, where `variable` denotes a variable name whose value is a string.

These variables can be associated with environment variables provided by the shell program or the operating system running on application-level gateways. The notion of variables for URL-based notation is originally inspired by the dynamic URL approach studied in [29]. whereas variables in the dynamic URL must be always associated with environment variables provided by the shell program or the operating system running on client-side computers, our variables can be dynamically interpreted at application-level gateways. We show an extended URL including variables as follows:

```
http://some.where.com/$(LIGHT)/!power=ON
```

where `$(LIGHT)` is a variable for specifying electric lights. When the value of the `LIGHT` variable maintained in the application-level gateway is `CEIL-LIGHT`, which specifies an electric light mounted on a ceiling, the above URL is interpreted as `http://some.where.com/CEIL-LIGHT/!power=ON`. When a new light is mounted

```
      <path>  ::=  ε  |  <string>  |  <search>  |  <command>  |  <path> / <path>
    <search>  ::=  ? <pair>  |  <search> & <search>
   <command>  ::=  ε  |  ! <pair>
      <pair>  ::=  <filed-name> = <value>
<field-name>  ::=  <string>
     <value>  ::=  <string>
```

Figure 2. Syntax

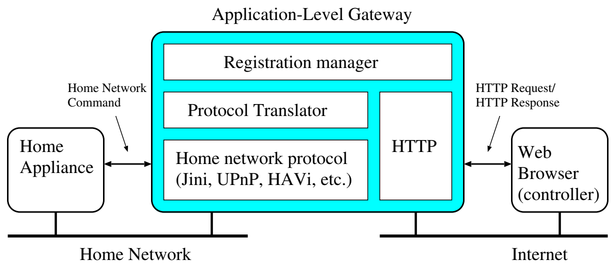

Figure 3. Structure of Gateways

in the room, the operating system may update the value of the `LIGHT` environment variable so that the value refers to one of the most suitable agents in the current environment. Consequently, agent migration can be dynamically adapted to the current environment.

3.4 Components

This framework consists of three elements Application-Level Gateways, Controllers and Appliances/Services. Since the Appliances/Services are application-specific, we will not discuss them here. This subsection describes Application-Level Gateways and Controllers.

3.4.1 Application-Level Gateways

An Application-Level Gateway works as a gateway between Appliance-specific networks and the Internet, as shown in Figure 3. The Application-Level Gateway bridges between HTTP-based control methods explained in the previous subsection, and various middle-ware-appliance control methods.

Application-Level Gateways in our virtual overlay network consist of four components. The first component implements home network protocols such as HAVi, Jini, or UPnP. This component behaves as a home network device. For example, if a component implements the Jini protocol, it behaves as a Jini device. The second component implements the HTTP protocol. This component behaves as a web server.

The third component implements a protocol translator. The translator converts between the HTTP transactions described in the previous section and a particular home network protocol. 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 a Jini translator and a 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 by a home network protocol from an appliances that implements the protocol. If an appliance on a home network is to be accessed from outside the home network, the name of that appliance should be registered on the application-level gateway. The name can be registered either via the HTTP protocol or a home network protocol. After 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.

Next, we describe the startup procedure. When an application-level gateway starts up, it will try to join the surrounding network. Using methods provided by the middleware particular to the protocols used in this network, it will investigate what devices or services are available. In a Jini network, for example, the gateway will use the lookup service of Jini. The results accumulated by this probing will be stored in a database, that lists the attributes used by the home network protocol, and the corresponding commands used in the HTTP-based method.

If the Application-Level Gateway receives an HTTP-based command from the Internet, it parses the command by referring to the database, and after determining what command should be sent to which device, applies the necessary protocol translation and sends out the corresponding middle-ware-specific command. On the other hand, pseudo-devices created on the gateway by the registration manager act like appliance-specific network devices. Thisallows devices on the application-specific network side to control devices on the Internet side of the gateway. In this way, the gateway appears to devices within the home network as if it provides the functions or services that are actually provided by devices on the Internet side.

3.4.2 Controller

Our Application-Level Gateway provides its control interface as an HTTP server, enabling any application-specific network device to be controlled using our HTTP-based control protocol. Therefore, it is acceptable to use an existing web browser as the controller. In recent years, web browsers have become embedded in various devices such as PDAs, cellular phones, and digital TV sets, not to mention PCs. All of these devices can be used to control other devices through the Application-Level Gateway.

When a generic browser is used as the controller as described above, the gateway must generate a web page containing command buttons, which are linked to each command the gateway will accept. On the other hand, if a controller is built to generate our specific HTTP-based commands by itself, it may have certain advantages over a generic browser, such as customizability, or adaptability to the environment. We describe the details of such a controller in the next section.

3.5 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 4, we assume that there are three home networks that are connected to the Internet via their respective Application-Level Gateways. The three networks, whose gateways have URLs `http://any.where.net`, `http://my.home.net` and `http://foo.bar.net`, each use one application-specific protocol, respectively UPnP, HAVi, and Jini.

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

First, we must register the HAVi VCR device in the UPnP network in order to control the HAVi VCR device from the UPnP PDA device. From the PDA, the user registers the HAVi VCR device in UPnP ApGW as "http://my.home.net/VCR" from the web browser on the PDA device. This uses "register" command, like this:

```
http://any.where.net/!register
  =''http://my.home.net/VCR''
```

If the UPnP ApGW receives a registration request, and then it sends a query to the HAVi ApGW using the HTTP-based protocol, to request information about the "VCR" device. The HAVi ApGW returns information about the "VCR" indicated by the URL. On the UPnP ApGW, a pseudo UPnP device of the "VCR" is created, using information that return from HAVi ApGW. The pseudo device joins the UPnP community with the name "VCR". When the PDA device wishes to control the HAVi VCR device, the PDA device tries to find the VCR device via the UPnP service discovery mechanism. The service discovery mechanism returns a reference to the pseudo device that is created on the 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/". The 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 a HAVi command, and is delivered to the target VCR device.

3.6 Prototype Implementations of Gateways

In the current implementation of the Virtual Overlay Network framework, each application-level gateway is implemented as a Java application running as a stand-alone process and offers HTTP-based interfaces to home appliances and service discovery systems for managing appliances. It has a simple repository database for interpreting attributes including its receiving URLs. Furthermore, it can handle unknown attributes by cooperating with its underlying service discovery systems such as Jini and UPnP.

4 Discussion

In this section, we describe extensions to the controller that enable it to send additional information, such as user preferences, personal identification, and environment information, along with the commands it sends to a gateway.

A controller that can generate HTTP-based control commands by itself can be used to send various bits of information to the gateway, thereby improving cooperation between the controller and the gateway. Examples of such information include user customization settings, and environment variables.

We assume that the controller is an extended web browser that can generate commands such as those proposed in this paper. An extended browser can send environment variables to a gateway, by embedding them into the HTTP-based commands it sends. An HTTP-based command before environment variables are embedded will look like this:

```
http://%(HOME)/?owner=%(USER)&?location=
      %(LOC)&?function=tv/!power=on
```

This command means to power on a TV set where owner is "USER" and its location is "LOC" at "HOME".

Let us assume environment information the controller has about itself and the user is like this.

```
USER = foo
HOME = http://my.home.net/
LOC = living-room
```

The controller treats the "%(" as the beginning of an environment variable. The string enclosed between "%(" and ")" is the name of the environment variable.The "%( )" part will be replaced with the value of the corresponding environment variable. The command sent in this case will be as follows.

```
http://my.home.net/?owner=foo&location
  =living-room&function=tv/!power=on
```

The environmental information the controller posseses will be updated according to the controller's location, the user currently using the controller, and other environmental parameters. Even if someone pushes the same buttons on the same controller, the emitted command may be different, depending on the circumstances in which the buttons were pressed. Other examples of environmental information that may be used to influence the control of home appliances, may be weather information, emotions of the user, and so on. The relationship between this user-agent functionality of the controller, and the proposed application control framework, is the subject of further study.

5 Related Work

Although most of the standards are based on the object model, there are many differences between each of the standards.

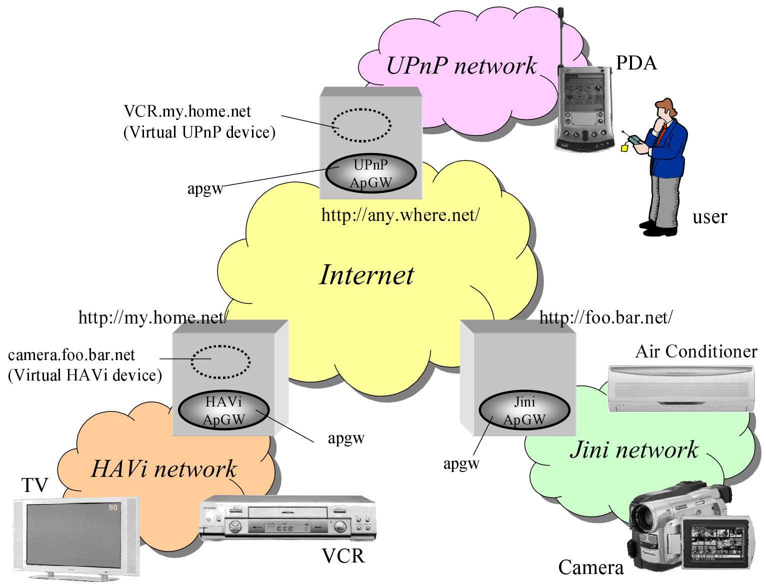

Figure 4. Connecting Home Networks

Jini[30] is a Java-based infrastructure for coordinating remote services and appliances. The Jini system provides mechanisms for service construction, lookup, and communication in a distributed system. However, Jini is heavily dependent on the Java language, whose runtime system requires a large amount of computational resouces, and such resources are usually very limited in a home appliance.

UPnP (Universal Plug and Play)[28] is an architecture for providing peer-to-peer network connectivity between PCs and appliances, UPnP allows a device to dynamically join a network, allocate itself an IP address, announce its presence and its capabilities upon request. UPnP is based on TCP/IP and web technologies to enable seamless proximity networking in addition to control and data transfer among networked devices in the home and office. However, UPnP is intended to profile features and capabilities of each appliance in XML files. Although XML-based profiles allow complex, powerful description of appliance capability as opposed to Jini's simple service attribute, it is difficult for home appliances to interpret XML-files.

HAVi (Home Audio Video Interoperability)[12, 17] provides a home networking standard for connecting digital audio and video consumer devices seamlessly. HAVi also provides a Java API for controlling A/V appliances, which enables control of appliances from within their programs. HAVi assumes that each appliance has an IEEE 1394 network interface, that has enough bandwidth to transmit multiple digital audio and video streams simultaneously. However, this requirement prevents HAVi from being widely used in various types of appliances.

It is expected that in the future home appliances will support different standards depending on their manufacturers' policies. However, these standards do not support connection of multiple home networks. Our virtual overlay network allows all home appliances to be connected and to control one another.

OSGi (The Open Services Gateway specification)[9] is a Java-based application layer framework that gives service providers, network operators, device makers, and appliance manufacturers vendor-neutral application and device layer APIs and functions. This strategy enables virtually all emerging home networking platforms, protocols and services to seamlessly inter-operate with back-end services, using existing residential telephone lines, cable TV, or electrical wiring. The technology needs to be platform-independent so that it can be implemented on a variety of computing environments, communications environments, consumer electronics and household products and platforms. Because the Open Services Gateway specification focuses exclusively on providing an open application layer and gateway interface for Services Gateways, it complements and enhances virtually all current residential networking standards and initiatives. Some of these include JINI, Bluetooth, CAL, CEBus, HAVi, Home API, HomePNA, HomePnP, HomeRF, and VESA.

The Open Services Gateway specification also preserves consumer investments in future smart home devices. For example, today, when a consumer switches from one home security provider to another, the entire internal network has to be replaced. By choosing devices compatible with the Open Services Gateway specification, consumers can switch between various vendor offerings, without having to replace virtually any of the networking infrastructure.

The advantage of OSGi is that it provides a framework for building extensible home gateways. This standard may be adopted to implement extensible application-level gateways.

6 Conclusion and Future Work

In this paper, we have proposed a virtual overlay network that integrates networked home appliances. The virtual overlay network determines the routes for delivery of messages and converts protocols and messages according to the characteristics of network and appliance. The virtual overlay network is very useful in implementing *Internet-scale ubiquitous computing*. We have described several scenarios to show the effectiveness of the proposed virtual overlay network, and have presented the current prototype system.

The thrust of our research is that the IP layer should provide functionalities that are necessary for building virtual overlay networks customized for respective applications that require complex communication. Most of the functionalities should be implemented on the top of the IP layer, and should be customized for respective applications. Therefore, supporting functionalities in the IP layer should be minimized, and providing some advanced functionalities in the IP layer in a generic way should be avoided because the functionalities implemented in the IP layer may be too generic and provide insufficient power for building realistic advanced applications.

From now on, we intend to work on several issues in order to extend the current prototype system. The first issue is building a more practical system integrating HAVi appliances, Jini appliances, and other appliances that support other protocols. Also, we would like to support conversion of streaming media depending on the characteristics of each appliance[2, 1]. Another aim of this work is to compose several home appliances on the Internet. The second issue is how to build a virtual overlay network customized for respective applications in an automatic way because there are expected to be numerous applications in future and it is not realistic to build a different virtual overlay network for each new application that appears in the future. Finally, the URL-based description of the current system is not yet satisfactory. We plan to design a more elegant and flexible description incorporating interfaces.

References

[1] E.Amir, S.McCanne and H.Zhang, "An Application Level Video Gateway," in Proceedings of ACM Multimedia '1995.

[2] E. Amir, S. McCanne and R. Katz, "An Active Service Framework and Its Application to Real-Time Multimedia Transcoding," in Proceedings of SIGCOMM'98, 1998.

[3] K.P. Birman, "Technology Challenges for Virtual Overlay Networks," in Proceedings of the 2000 IEEE Workshop on Information Assurance and Security, 2000.

[4] T. Berners-Lee, "Universal Resource Identifiers in WWW," IETF, 1994.

[5] L.F. Cabrera, M.B. Jones, M. Theimer, "Herald: Achieving a Glocal Event Notification Service," in Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII), 2001.

[6] , Yang-hua Chu, Sanjay G. Rao and Hui Zhang, "A Case For End System Multicast," Proceedings of ACM Sigmetrics, 2000.

[7] T. von Eicken, W. Vogels, "Evolution of the Virtual Interface Architecture," IEEE Computer, Vol. 31, No. 11, 1998.

[8] G. Girling, J. Li Kam Wa, P. Osborn, R. Stefanova, "The Design and Implementation of a Low Power Ad Hoc Protocol Stack," in Proceedings of the IEEE Wireless Communications and Networking Conference, September 2000.

[9] Li Gong, "A Software Architecture for Open Service Gateways," IEEE Internet Computing, Vol.5, No. 1, 2001.

[10] M. Gritter, D. R. Cheriton, "An Architecture for Content Routing Support in the Internet," USENIX Symposium on Internet Technologies and Systems, 2001.

[11] I. Guardini, P. Fasano, G. Girardi, "IPv6 Operational Experience within the 6bone," INET 2000, 2000.

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

[13] IEEE Std 1394-1995, Standard for a High Performance Serial Bus.

[14] "The Inktomi Overlay Solution for Streaming Media Broadcasts," Technical Report, Inktomi 2000, http://www.inktomi.com/products/media/docs/whtpapr.pdf .

[15] J. Jannotti, D.K. Gifford, K.L.Johnson, M.F. Kaashoek and J.W. O'Toole Jr., "Overcast: Reliable Multicasting with an Overlay Network," in Proceedings of the Fourth Symposium on Operating Systems Design and Implementation, 2000.

[16] A. Jones, A. Hopper. "The Prototype Embedded Network (PEN)," AT&T Laboratories, Cambridge, Technical Report 2000.15.

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

[18] B.A. Miller, C. Bisdikian, "Bluetooth Revealed," Prentice Hall, 2000.

[19] P. Mockapetris, K. Dunlap, "Development of the Domain Name Systems," in Proceedsing of the SIGCOMM'88, 1988.

[20] T. Nakajima, "A Framework for Building Adaptive Continuous Media Applications using Service Proxies," Handbook of Multimedia Computing, , CRC Press, 1998.

[21] T. Nakajima and F.Stajano, "Technical Challenges for Building Internet-Scale Ubiquitous Computing," in Preparation.

[22] C.E. Perkins, ed. "IP Mobility Support," RFC2002 IETF, 1996.

[23] S. Pope, D. Roberts, D. Riddoch, K. Mansley, D. Clarke, T. Mills, A. Hopper, "CLAN Scalable High Performance User Level Networking," IEEE Gigabit Networking Workshop GBN 2001.

[24] K. Savets, N.Randall, and Y. Lepage, "MBONE: Multicasting Tomorrow's Internet," IDG, 1996.

[25] J. Scott, F. Hoffman, M. Addlesee, G. Mapp, A. Hopper, "Networked Surfaces: A New Concept in Mobile Networking," in the Proceedings of International Workshop on Mobile Computing, Systems and Applications, 2000.

[26] A.C. Snoeren, H. Balakrishnan, F. Kaashoek, "Reconsidering Internet Mobility," in Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII), 2001.

[27] D. Tennenhouse, et. al, "A Survey of Active Network Research," IEEE Communication Magazine, Vol.35, No.1, 1997.

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

[29] G. Voelker, and B. Bershad, "Mobisaic: An Information System for a Mobile Wireless Computing Environment," Proceedings of Workshop on Mobile Computing Systems and Applications, IEEE Computer Society, 1994.

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

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

[32] D. Wessels, K. Claffy, "Internet Cache Protocol(ICP), Version 2," RFC2186 IETF, 1997.

[33] The X10 Protocol, http://www.x10.org/

IEEE COMPUTER SOCIETY