Service Platform Technology for Accelerating Use of Network-ready Devices

● Jun Kakuta ● Shoji Naito ● Ryuichi Matsukura

An increasing number of devices, including home appliances, audio-visual equipment, energy-related devices, and healthcare devices, are network-ready. While the focus at present is on smartphone-based services for monitoring and controlling the state of individual devices, the spread of advanced services for simultaneously controlling a variety of devices is anticipated. However, the control procedures and functions of network-ready devices differ significantly between vendors and even between models. Moreover, they are frequently upgraded. As a result, the development of services that use various types of devices is hampered. A service platform that can conceal such differences among individual devices is thus becoming increasingly important. We have developed a service platform that models any device as a virtual device having common properties for that device type and that provides application programming interfaces (APIs) for controlling these virtual devices. These APIs enable a service developer to control an actual device by controlling its virtual device, which negates the need for detailed knowledge of individual devices and promotes efficient development of services. Fujitsu has applied this service platform to its "Smart Sensing Platform (SSPF)."

1. Introduction

Devices equipped with communication functions that make them network-ready are increasing in number. They come in many types, from home appliances and audio-visual (AV) equipment such as air conditioners and televisions to energy-related devices and healthcare devices such as solar-power generators and weight scales. This trend is driving the development of services linked to network-ready devices including living-support services for remotely monitoring and controlling home appliances and performing security functions,[1),2)] services for scheduling the recording of TV programs,[3)] services for monitoring solar-power generators,[4)] and services for managing healthcare.[5)]

At the same time, control methods and functions of network-ready devices depend on device type, vendor, and model and can therefore vary greatly. As a result, developers must familiarize themselves with the control protocol of each and every device, but this makes the development of a service that handles a variety of devices all the more difficult. This state of affairs has limited the range of services to those dedicated to individual devices or to devices of a specific vendor, which does not contribute much to user convenience.

We, the authors, consider that services that make use of network-ready devices have the potential to become living-support and social infrastructures of the future. To accelerate the development of these services, we are developing service platform technology, the topic of this paper.

2. Problems in device-control services

Standardization in this field has been active in recent years at such organizations as the Energy Conservation and Homecare Network (ECHONET),[6)] targeting the control of home appliances, home fixtures, and energy-related devices, the Digital Living Network Alliance (DLNA),[7)] targeting the control of AV devices, and the Continua Health Alliance (CHA),[8)] targeting the control of healthcare devices. Support products related to this standardization have also been appearing in rapid succession.

Though emphasis is currently being placed on services for monitoring and controlling the state of individual devices via smartphones and other smart terminals, we anticipate the spread of services that are capable of simultaneously controlling various types of devices. For example, given a smart house that contains a variety of network-ready devices, a device-control service must not be limited to just the monitoring and control of individual devices—it must also be able to control those devices in a way that maximizes energy-usage efficiency for the entire house. The following problems must be addressed to achieve such a service.

1) Abundant and diverse device-control procedures

Standard control procedures differ greatly between industries such as the home appliance industry, AV device industry, and healthcare industry. This means that any attempt at developing a service that handles devices applicable to more than one industry will require separate development work for each device having a different control procedure. This requirement can extend the development period substantially. In addition, functions of even devices applicable to the same industry can have minor differences depending on the vendor and model, meaning that control procedures must often be modified for individual devices.

2) Frequent device upgrading

The cycle of placing new device models on the market and updating functions is quite short, especially for home appliances, and the frequency of changing functions and interfaces is high. Thus, when a user purchases a new device to replace a current one, the application programming interface (API) for controlling that device may have changed, which means that the device may not operate unless control software is updated. However, revising an entire service every time device functions are updated can raise the development and operations costs of that service.

3. Approach to solving problems

The following processes are currently performed on a service-by-service basis for each device to be controlled, as shown in **Figure 1 (a)**:

1) Application processing for message display, service-specific logic, etc.
2) Device interface processing for commands and parameter conversion related to device control
3) Device protocol processing for interpreting device control protocol

In the future, device interface processing and device protocol processing—which are highly device dependent—will be performed on a service platform, while application processing will be performed on a service-by-service basis, as shown in in **Figure 1 (b)**. This approach negates the need for the service developer to be aware of device differences. In other words, developing a service platform that promotes device virtualization solves the two problems described above

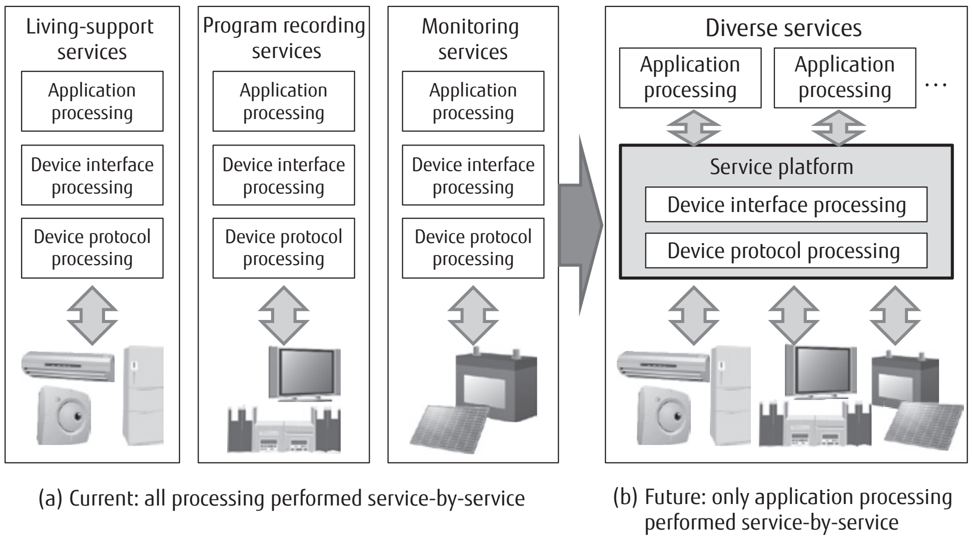
Living-support services
Application processing
Device interface processing
Device protocol processing
Program recording services
Application processing
Device interface processing
Device protocol processing
Monitoring services
Application processing
Device interface processing
Device protocol processing
Diverse services
Application processing
Application processing
…
Service platform
Device interface processing
Device protocol processing
(a) Current: all processing performed service-by-service
(b) Future: only application processing performed service-by-service

Figure 1
Current and future approaches to device control.

by enabling a service to manage any device as a virtual device having common device properties for that device type and to control any object by using virtual-device control APIs.

4. Service platform technology

4.1 Architecture

To conceal the diverse characteristics of actual (physical) devices, device virtualization architecture that we have developed provides two important layers: a protocol normalization layer and an interface normalization layer corresponding to device protocol processing and device interface processing, respectively, as shown in **Figure 2**. Furthermore, to facilitate communications, the architecture features virtual-device control APIs independent of physical devices between the service layer and interface normalization layer as well as intermediate control commands between the interface normalization layer and protocol normalization layer.

Providing loose coupling between layers in this way makes it possible to achieve a flexible configuration to fit system requirements, such as deploying only the service layer in the cloud or deploying both the service layer and interface normalization layer in the cloud.

Moreover, forming a separate protocol normalization layer enables protocol processing for specific physical devices to be accomplished on a plug-in basis, which makes for flexible extending of functions.

A service platform based on this architecture can be given the ability to handle changes in the communication protocol of an existing device by simply adding a software module and an intermediate-control-command generation dictionary for processing the new protocol to the protocol normalization layer. In short, there is no need to revise a huge software program. The service platform can also be given the ability to handle the appearance of a new device equipped with an existing communication protocol by simply revising the existing data-model generation dictionary and intermediate-control-command generation dictionary.

4.2 Protocol virtualization technology

As shown in Figure 2, the protocol normalization layer uses protocol virtualization technology to facilitate communications with the interface normalization layer. This technology terminates communications with either standard communication protocols like ECHONET or DLNA or proprietary protocols installed in physical devices, which it normalizes into intermediate control commands independent of physical devices.

This protocol virtualization technology uses the intermediate-control-command generation dictionary to create abstractions of actual communication-protocol names and commands of physical devices and generates intermediate control commands in eXtensible Markup Language (XML) format.

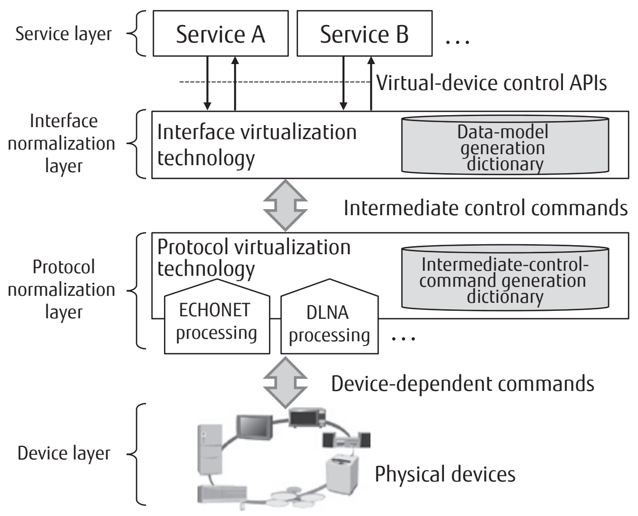
Service layer
Service A
Service B
...
Virtual-device control APIs
Interface normalization layer
Interface virtualization technology
Data-model generation dictionary
Intermediate control commands
Protocol normalization layer
Protocol virtualization technology
Intermediate-control-command generation dictionary
ECHONET processing
DLNA processing
...
Device-dependent commands
Device layer
Physical devices

Figure 2
Device virtualization architecture.

Intermediate control commands generated in this way can be conveyed to the interface normalization layer via the Simple Object Access Protocol (SOAP). Adopting such a communication configuration makes it possible to deploy only processing related to protocol virtualization technology in the gateway on the home-network side, thereby reducing the processing load in the gateway.

To achieve a plug-in platform for protocol processing, we adopted the Open Services Gateway initiative (OSGi) platform,[9] a Java-based dynamic plug-in execution platform, and implemented protocol-dependent processing as OSGi software modules (OSGi bundles). This enables us to use OSGi bundle management functions and make dynamic additions and changes to protocol processing modules.

4.3 Interface virtualization technology

Again as shown in Figure 2, the interface normalization layer features interface virtualization technology that enables a service to control any virtual device via virtual-device control APIs. This is accomplished by mutually exchanging intermediate control commands with proprietary control commands of the virtual-device control APIs based on a data-model generation dictionary.

For example, if a change occurs in the power ON/OFF status of an air conditioner, the protocol normalization layer will issue an intermediate control command setting device type as “Air Conditioner” and command type as “notification of change in power.” Then, on receiving this command, the interface normalization layer will interpret “notification of change in power” as a change in the “Operation Status” property and notify the service accordingly via a virtual-device control API as described below.

Conversely, the interface normalization layer can receive control instructions from a service for a virtual device via a virtual-device control API. On receiving such an instruction, this layer will generate an intermediate control command by using the data-model generation dictionary and send it to the protocol normalization layer.

Four types of virtual-device control APIs are provided for controlling any virtual device: *get* (obtain a virtual-device property), *set* (specify a virtual-device property), *subscribe* (request immediate notification of any change made to a virtual-device property), and *notify* (issue information on any change made to a virtual-device property). These get-set and subscribe-notify types of APIs are widely used in Web services and other applications. They simplify the development of services that use devices even for Web service developers who are not familiar with the specific control procedures of network-ready devices.

For example, a service can obtain the operation status of a physical air conditioner by calling the *get* API to obtain the “Operation Status” property of the corresponding virtual air conditioner, and it can call the *set* API and set “OFF” as a parameter to turn the physical air conditioner off.

A service can also call the *subscribe* API with respect to the “Detection Status” property of a virtual human sensor. It would then be advised of any change in the detection status of the corresponding physical human sensor via the *notify* API.

5. Technology testing

An energy management system (EMS) was developed to evaluate whether the technology presented in this paper could simplify service development when targeting an experimental “iHouse” equipped with many network-ready devices. This system had functions for collecting and visualizing information on home appliances and other devices as well as functions for controlling the power supply, operation mode, etc. of those devices. It ran on a service platform incorporating the developed technology.

The experimental iHouse contained about 210 network-ready devices from home appliances and controllable windows and curtains to temperature/humidity sensors (**Figure 3**).[10],[11]

This EMS was developed in about one week by a developer having no experience in developing services using home appliances and other network-ready devices. This is one-fourth to one-fifth the work hours that would be needed to develop a cloud service that could directly control devices via the ECHONET home-appliance control protocol.

The developed EMS could also be enhanced to deal with added functions in network-ready devices by simply modifying configuration files on either the platform or service side. This made it practically unnecessary to revise software programs whenever a change in device functions occurred.

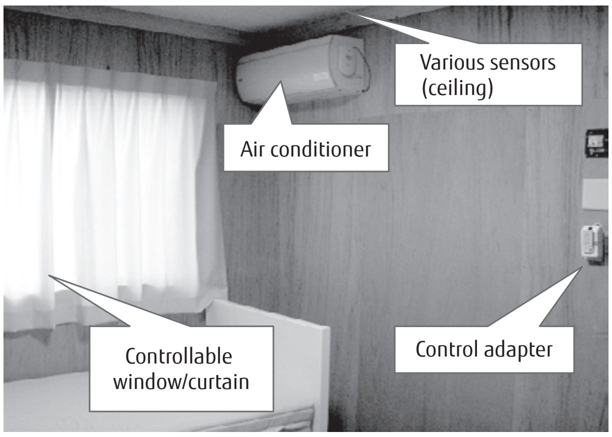

Figure 3
Interior of experimental "iHouse."

6. Future activities

As described above, we developed service platform technology that provides a common control interface in the form of APIs and data models with the aim of making the development of services using network-ready devices more efficient.

At the same time, making the development of different types of services more efficient requires platform technology corresponding to those types of services. Taking EMS as an example, we can see that devices installed in a home differ from house to house, which means that, even with a common interface making use of virtual devices, the service will have to determine what devices and device types exist in the target home. A large number of devices will only complicate application processing.

Solving this problem requires that the platform be able to process service requests such as "get room temperature" and "set room temperature to 28°C" independent of devices. To achieve this capability, device virtualization technology that includes not only syntax (data structure and format) but also semantics (data meaning) will be essential. Such technology should appropriately model real-world objects like houses, rooms, and people so that they can be handled by an EMS, should assign correspondences between those objects and models on the basis of the meaning of collected data, and should appropriately decompose the control of non-device real-world objects like a room into the control of devices installed in that room.

A highly promising service area for such technology is EMS. We thus plan to study specific approaches to modeling real-world objects for future EMSs and achieving semantic-based processing and to implement such functions as high-level EMS middleware in the service platform introduced here.

7. Conclusion

We described service platform technology that can accelerate the development of services using network-ready devices. This technology has been applied to Fujitsu's Smart Sensing Platform (SSPF),[12] which facilitates the creation of EMSs for homes and retail shops. The SSPF supports the ECHONET Lite standard drafted by the ECHONET Consortium and recommended by the Smart House Standardization Study Group in the Japan Smart Community Alliance—an organization affiliated with the Japanese Ministry of Economy, Trade and Industry—as a standard EMS protocol for homes. SSPF enables ECHONET Lite-certified devices to be controlled by using the *get*, *set*, *subscribe*, and *notify* APIs described in this paper.

Network-ready devices will be able to naturally determine the conditions surrounding the user and achieve an environment suitable to the user by setting appropriate controls based on those conditions. Such living-support services, of which the user need not be aware, will be continuously present in people's lives, and the existence of network-ready devices will come to play an important role in living-support and social infrastructures of the future.

We feel that the number and types of network-ready devices will increase by leaps and bounds going forward. Providing a service platform that can simplify the control of such a large number of devices should promote the birth of innovative services supporting people's lives and society and help create a safe, comfortable, and sustainable way of life.

This technology was a result of research and development conducted as part of the "Standardization of Intermediate and Management Platform Interfaces for the Popularization of Services that Contribute to a Reduction in Environmental Load" project funded by the Japanese Ministry of Internal Affairs and Communications. Some of the technology resulting from this project is used in SSPF.

We would like to express our deep appreciation to Professor Yasuo Tan of the Japan Advanced Institute of Science and Technology for his cooperation in developing this technology and to the Ishikawa Sunrise Industries Creation Organization, Ishikawa Information System Association, and all concerned for their assistance in setting up and managing the experimental iHouse.

References

1) Toshiba Lighting & Technology Corporation: FEMINITY. *http://feminity.toshiba.co.jp/feminity/feminity_eng/index.html*
2) Panasonic Corporation: Lifinity (in Japanese). *http://www2.panasonic.biz/es/densetsu/lifinity/*
3) DiMORA (in Japanese). *http://dimora.jp/*
4) Sharp Corporation: Web Monitoring Service (in Japanese). *http://www.sharp.co.jp/sunvista/select/monitoring/*
5) TANITA Corporation: Body Chart (in Japanese). *http://www.karadakarute.jp/tanita/index.jsp*
6) ECHONET CONSORTIUM. *http://www.echonet.gr.jp/english/index.htm*
7) Digital Living Network Alliance. *http://www.dlna.org/*
8) Continua Health Alliance. *http://www.continuaalliance.org/*
9) OSGi Alliance. http://www.osgi.org/
10) M. Kaneshima et al.: The Test Bed House for Home Network Demonstration / i-House: Part 1. House Design & Thermal Load Characteristic. Proceedings of Annual Meeting of the Architectural Institute of Japan D-2, 2010, pp. 1399–1400 (in Japanese).
11) S. Ogino et al.: The Test Bed House for Home Network Demonstration / i-House: Part 2. Sensor & Network Equipment. Proceedings of Annual Meeting of the Architectural Institute of Japan D-2, 2010, pp. 1401–1402 (in Japanese).
12) Fujitsu Ltd.: Fujitsu Launches SSPF v01 Software to Facilitate the Construction of Energy-Management Systems for Homes and Shops. *http://www.fujitsu.com/global/news/pr/archives/month/2012/20120515-02.html*

Jun Kakuta
Fujitsu Laboratories Ltd.
Mr. Kakuta is engaged in the research of machine-to-machine and human-to-human real-time communication technologies.

Ryuichi Matsukura
Fujitsu Ltd.
Mr. Matsukura is engaged in the development of network solutions for smart houses.

Shoji Naito
Fujitsu Kansai-Chubu Net-Tech Ltd.
Mr. Naito is engaged in the design and development of sensing-platform architectures.