Journal of Information Processing Vol.20 No.3 525–530 (July 2012)

[DOI: 10.2197/ipsjjip.20.525]

Regular Paper

General Middleware Bridge to Support Device Interoperability on Different Middlewares

Hark-Jin Lee[1,a)] Young-Sung Son[1] Jun-Hee Park[1]
Kyeong-Deok Moon[1] Jae-Cheol Ryou[2]

Received: August 31, 2011, Accepted: February 3, 2012

Abstract: In this paper, we investigate an integrated architecture to support interoperability among heterogeneous middlewares on home networks. We propose and implement a general middleware bridge to support device interoperability on different middlewares for efficient home networks. The IWFEngine (interworking function engine) architecture provides an interface for identifying and utilizing services among devices using simple rules in other to support interoperability among heterogeneous middlewares. Through the registered rules, local middleware messages are translated into standard messages, and vice versa. Unlike existing integrated middleware architectures, the IWFEngine architecture improves the efficiency, and a convenient adaptor development is possible through simple rules and by using local middleware messages. By this configuration, a conversion rule for exchanging messages between devices on various middlewares is described which does not require the modification of the corresponding middleware, and operations can be performed in accordance with the existing corresponding middleware mechanism. Finally, the overhead incurred by a centralized and integrated middleware architecture can be reduced by distributing adaptors into multiple devices.

Keywords: interoperability, middleware, home network

1. Introduction

Various methods of supporting different middleware devices have been developed, and the interoperability between devices on different middlewares is becoming available. Because such a method places the emphasis on enabling the interoperability between devices connected to different middlewares, whenever a new middleware appears, a new adaptor corresponding to the new middleware needs to be developed, and this is difficult to achieve. The need for a rapid and convenient method of adaptor development is growing.

Many home network middlewares are widely used, including the universal plug and play (UPnP) [1], Jini [2], KONNEX [3], ECHONET [4], home audio video interoperability (HAVI) [5], Lonworks [6], RS485 [7] and so on. There are one-to-one bridge methods for home network middlewares such that each one connects are middleware to another.

The existing middleware bridge methods create and support the required adaptors, and a method for converting a message for interoperability between devices should be differently developed for each adaptor. A middleware adaptor developer must possess knowledge of a general middleware when a new middleware appears, and a great deal of knowledge and effort is also required for message converting method to the middleware adaptor developer. For example, the middleware adaptor developer should know all essential functions of the adaptor under implementation, such as an adaptor ID system, a definition method of a protocol for a multicast, and so on. As another example, the middleware adaptor developer should know the definition method of a standard protocol and should perform the detailed analysis and substitution processes to convert the standard protocol. It is difficult for the middleware adaptor developer to determine the development time and the standard protocol. This paper presents the design and the implementation of a general middleware bridge for heterogeneous home network middlewares to support a convenient and efficient home network environment.

This paper is organized as follows. In Section 2, we introduce the research in progress for the development of integrated middlewares which support interoperability among devices in home networks, analyze problems caused by them, and present our motivation for this study. In Section 3, we describe the design of the proposed architecture for a general middleware bridge to support interoperability. In Section 4, we describe and explain the results of the implemented framework. Finally, in Section 5, we give our conclusions and discuss future work.

2. Related Work

This paper discusses the effects of heterogeneity on the interoperation of home network middlewares. In general, heterogeneity means that the interfaces and architectures of diverse components that exist in a specific domain are different [8]. Heterogeneity can exist in many parts of a system (information encoding methods, network protocols, data formats, and so on). If standardization is established among these parts, problems caused by heterogeneity

[1] Electronics and Telecommunications Research Institute, Daejeon, South Korea
[2] Chung-Nam National University, Daejeon, South Korea
[a)] gausslee@etri.re.kr

© 2012 Information Processing Society of Japan

are eliminated, and various home network usage scenarios can be designed with seamless interoperation. However, it is not easy to standardize middlewares related to home networks unlike existing middlewares such as TCP/IP because the existing devices and services are too diverse in the home network environment. Also, it is much more difficult to predict the future because the existing services and devices may continue to be developed. Even though it is predictable, there is a limit to solving the fundamental problems of heterogeneity.

2.1 Interoperation Mechanisms among Heterogeneous Home Network Middlewares

Interoperation mechanisms among the heterogeneous home network middewares currently under investigation can be classified into two types, namely, an individual bridge and an integrated framework. An individual bridge provides compatibility using a one-to-one bridge protocol between middlewares to support the interoperability among heterogeneous middlewares. The UPnP-to-Havi bridge [9] was developed by Thomson Multimedia and Philips, and the interoperation of Jini and UPnP [10] was researched at New Orleans University. It is useful for interoperation between two specific middlewares; therefore, it has a scalability problem because it fails to provide a consistent means of interoperation between various types of middleware as the number of bridges increases and connections can also be complicated by the introduction of new middlewares. An integrated framework provides an abstracted common layer above various middlewares and has an architecture that bridges each middleware based on the common layer. This type of architecture has an advantage that, even if a new middleware is developed, it can be easily integrated with other middlewares if an appropriate agent is implemented. At Waseda University, middleware integration has been attempted through a simple object access protocol (SOAP) gateway configuration [11], and studies on interoperation services among heterogeneous middlewares have been under way at OSGi Alliance [12] and ETRI [13], [14].

2.2 Analysis of Integrated Framework-based Mechanism

The model presented in this paper is based on an integrated middleware framework, and the following four issues should be considered.

(1) How can devices adopting different middlewares find each other transparently?

Each middleware utilizes different service discovery mechanisms. Due to the differences between the mechanisms, even devices providing compatible services cannot recognize each other if the middlewares they adopt are heterogeneous.

(2) How can services adopting different middlewares invoke each other?

A service invocation mechanism of RS485 uses byte code. UPnP uses SOAP to invoke a service transferring XML text stream. Problems caused by the differences in service invocation mechanisms must be solved to provide interoperability between heterogeneous middlewares. To solve this problem, the syntax elements of middlewares (e.g., method name, the order of arguments, the type size of return value and arguments) should be adjusted. It is also necessary to convert calling methods according to the service invoke mechanisms of each middleware.

3. IWFEngine

The IWFEngine is a general middleware bridge framework designed in consideration of the adaptor development method. To solve problems caused by the differences in heterogeneous middleware interfaces during the interoperation process as pointed out in Section 2, we propose a library architecture such as that shown in **Fig. 1**.

The IWFEngine consists of four main modules:

- Rule Converter
- Message Converter
- Adaptor Manager
- IWFEngine Manager

The IWFEngine can distribute each adaptor to various servers in other to solve the bottleneck problems encountered in a centralized architecture. Also, all protocols have an open architecture using XML and provide utility classes and APIs based on C++ and C to assist in developing adaptors for newly defined middlewares.

3.1 Rule Converter

The Rule schema helps the middleware adaptor developer create conversion rules of the XML type through a validity checking function. **Figure 2** shows rule schema. The Rule Converter registers the message conversion rule for each message type. For example, the Rule Converter uses and generates an XSLT document to convert the conversion rules for a registered standard message and for the local message into a conversion XSLT document. **Figure 3** shows the procedure of rule conversion.

- 'Type' attribute is rule type.
- 'For' element is used to process the XML node in the case of a duplicate.

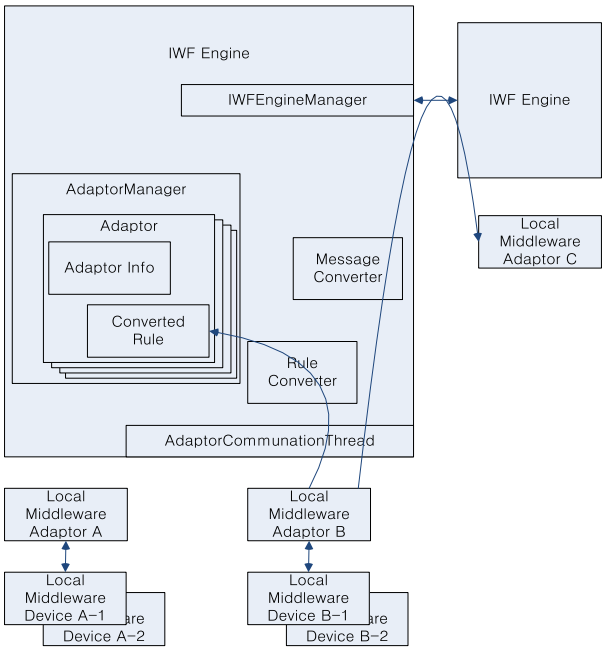

Fig. 1 IWFEngine architecture.

© 2012 Information Processing Society of Japan

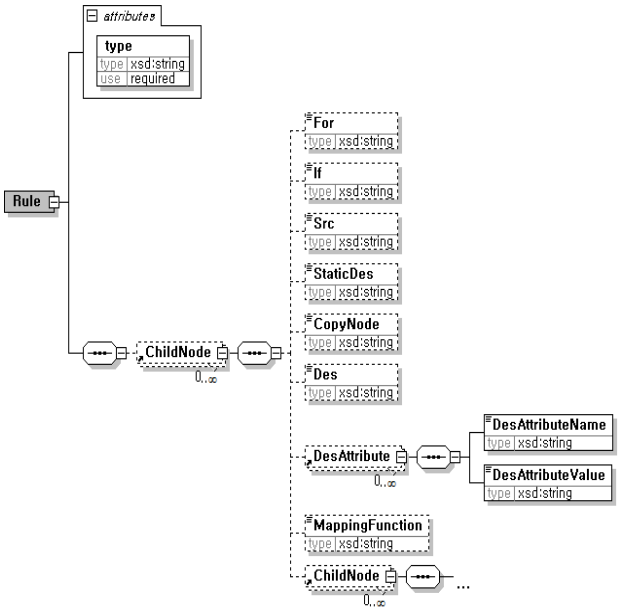

Fig. 2 Rule schema.

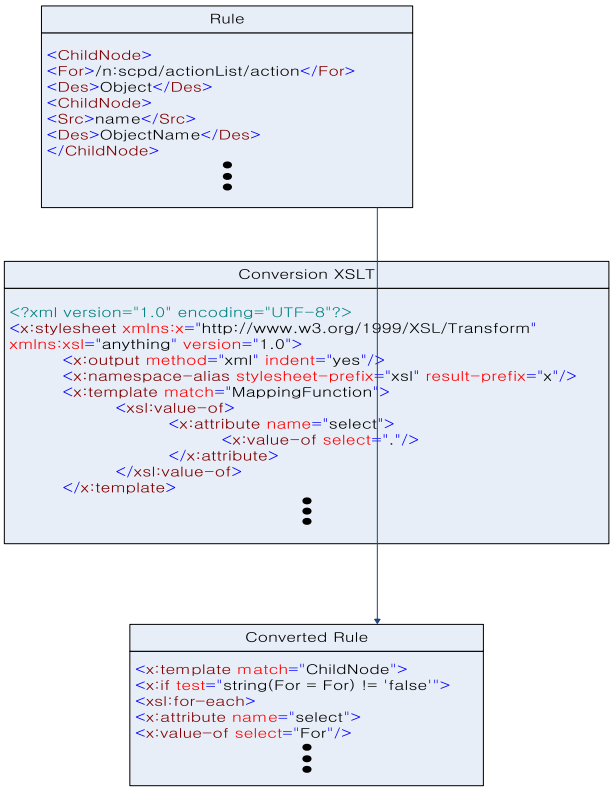

Fig. 3 Procedure of rule conversion.

- 'If' element is used to process the XML node according to a condition of the XML node.
- 'Src' element is used to process the XML node mapping from a source XML to the destination XML.
- 'StaticDes' element is used to put an explicit value of the destination XML if the node does not exist in the source XML.
- 'Des' element is the name of the node description to the destination XML.
- 'DesAttribute' element is the attribute to describe the destination XML.

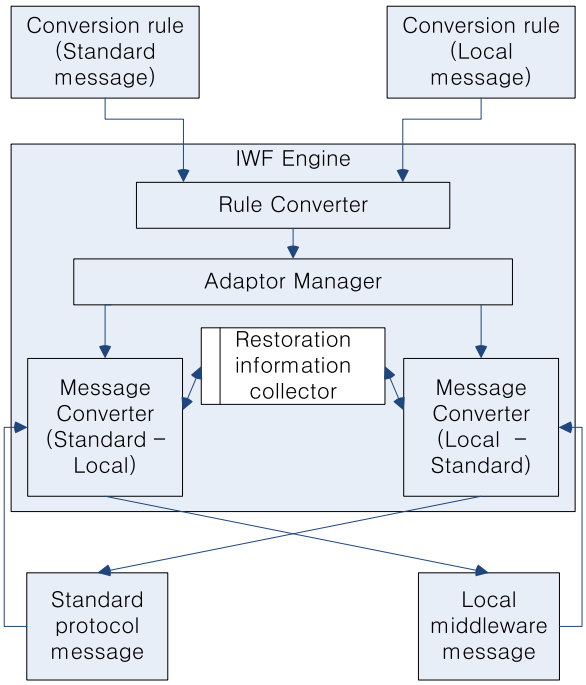

Fig. 4 Message conversion.

- 'MappingFunction' element transforms the source XML and reflects the destination XML.
- 'ChildNode' element carries out a recursive call to configure the hierarchy.

3.2 Message Converter

The Message Converter includes the standard–local Message Converter and the local–standard Message Converter. The standard–local Message Converter converts the standard message into the local message by using the registered conversion rule. The local–standard Message Converter converts the local message into the standard message by using the registered conversion rule. **Figure 4** shows message conversion.

The Restoration Information Collector collects restoration information for restoring the message in order to prevent data from being lost when the Message Converter converts the message. During the conversion of the original message into the converted message, the messages have different schemas and data expression types. Since data for the converted message in the Message Converter cannot include all data of the original message, a loss cannot be completely avoided.

The loss occurs only when a local message is converted into a standard message. When a local message is converted into a standard message, the Restoration Information Collector stores the corresponding restoration information. Restoration is performed by requesting the information when the standard message is converted into the local message. The Restoration Information Collector manages the restoration information for restoration without loss.

3.3 Adaptor Manager

As previously mentioned, home network middlewares for the operation of home appliances cannot be interoperable due to their different protocols and execution mechanisms. A layer is required

© 2012 Information Processing Society of Japan

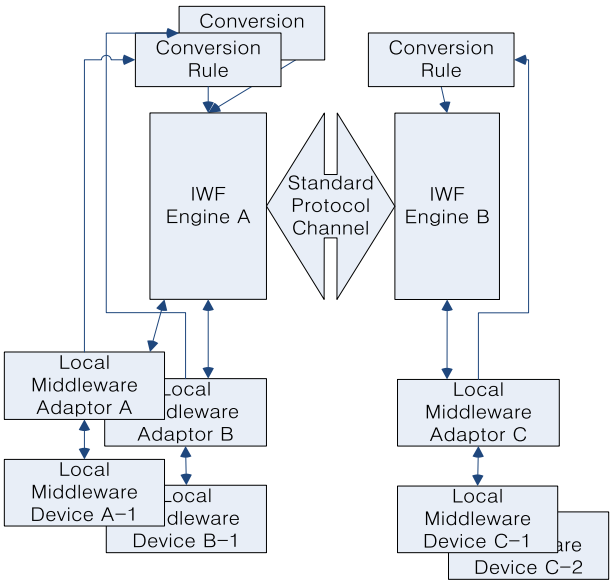

Fig. 5 Distributed adaptor architecture of IWFEgine.

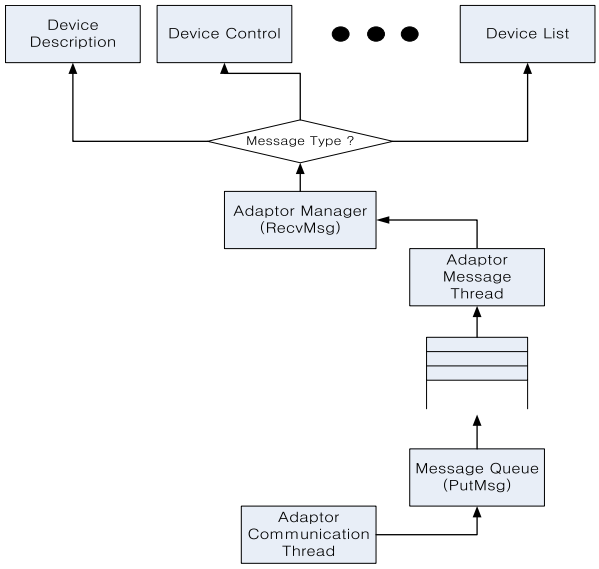

Fig. 6 Procedure of message transmission from an adaptor to the Adaptor Manger.

to abstract different middlewares into one for integration.

In this study, we define an abstract layer called IWML (Inter-Working Markup Language). This layer consists of the minimum number of components required for home appliances to contain all the common parts of diverse middlewares. In order for appliances to be capable of operating in a home network environment, they must have at least five components of the following: Device Description, Device Control, Device Sensor, Device List, Device Delete. Device Description is a device specification which can be understood by human beings and Device Control is a function performed by the device. Each adaptor must have its own Device Sensor and provide mechanisms that notify the status to the outside or assist the outside to recognize the status. Device List which is managed by the adaptor is a device list. Device Delete is use to report if a device managed by the adaptor is removed. It is possible for an appliance to interoperate with other appliances as long as this kind of mechanism is provided either from outside or inside of the appliance.

Adaptors and the Adaptor Manager are reliable and stable because they communicate through TCP/IP. Moreover, by separating the adaptor from Adaptor Manager as shown in **Fig. 5**, the IWFEngine architecture which supports the intercommunication through TCP/IP can solve the overhead problem which may occur in the centralized integrated architecture.

3.4 IWFEngine Manager

More than one IWFEngine can exist in a home network. If there is only one IWFEngine, All adaptors should be connected to an IWFEngine. This causes too much overload to an IWFEngine. Thus, we propose that each IWFEngine be assigned to the limited number of adaptors. The number of adaptors shall be decided by the hardware processing power of the IWFEngine.

When an IWFEngine starts, the IWFEngine Manager broadcasts its own ID which is selected randomly, to the other IWFEngines. If one IWFEngine acknowledges that I am using an ID, the IWFEngine Manager broadcasts the regenerated ID. Through this mechanism, the IWFEngine can assign a unique ID. When an IWFEngine ends, the IWFEngine Manager broadcasts a Bye-Bye message. The other IWFEngines are aware that the IWFEngine is disabled.

The IWFEngine Manager through the standard protocol channel delivers messages among all IWFEngines. The Standard Protocol Channel delivers standard messages without any message conversion.

3.5 Implementation of IWFEngine

The IWFEngine provides an adaptor to each middleware to discover or remove heterogeneous middleware services. In this paper, we implement each UPnP Agent and RS485 Agent using C++ and C, respectively. When new middlewares are added, the IWFEngine generates an ACT (Adaptor Communication Thread) which is in charge of the communication with the adaptor. The ACT sends messages over the network to a message queue in the Adaptor Manager and disappears along with the adaptor. Messages in a message queue are sent to the Adaptor Manager, and an appropriate routine is invoked according to the message type. **Figure 6** shows the procedure of message transmission from an adaptor to the Adaptor Manger.

4. IWFEngine Test

The IWFEngine proposed in this paper as a general middleware bridge architecture can easily make rules to meet the developer's requirements. We have constructed a Device Description, Device Sensor, Device Control, Device List and Device Delete in order to test the interoperability among heterogeneous services to support home automation.

Figure 7 shows the message conversion process example. An RS485 adaptor transfers light Device Description message to the IWFEngine. The IWFEngine converts the transferred message into a standard message through a conversion rule. The IWFEngine converts a standard message into a local middleware message for the UPnP adaptor. We confirmed that the UPnP controller discovered the RS485 light when the UPnP adaptor re-

© 2012 Information Processing Society of Japan

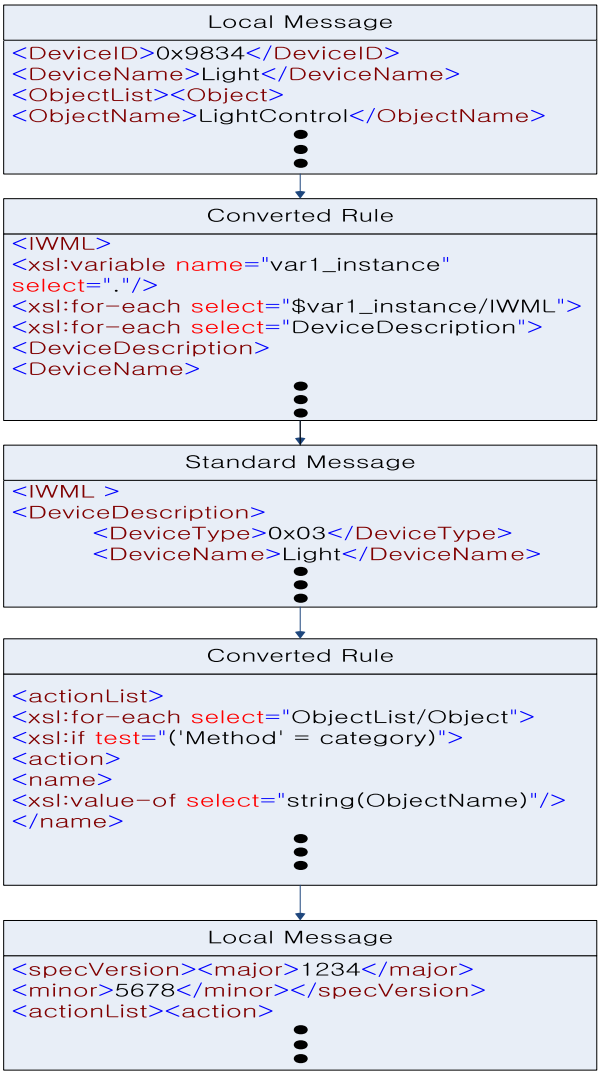

Fig. 7 Message conversion process.

ceived the Device Description message.

The experiment proved that heterogeneous devices could successfully interact with each other following the making of simple rules. Also, we verified that a developer can manage each device efficiently and conveniently using an adaptor.

We confirmed that the message gets through within 3 seconds from the UPnP Device to the RS485 Device.

In order to evaluate the efficiency of the proposed techniques, we have implemented them on Linux 3.1.2 kernel. Our hardware system is based on Intel Core i7 processor (running at 3.4 GHz) with a 2 GB RAM.

Figure 8 shows how much time it takes to convert a local message to a standard message with various message sizes. It only takes about 200 ms to convert a message of 1 Mbytes.

5. Conclusions and Future Works

For the efficient utilization of home networks, problems raised by the heterogeneity of middlewares must be solved. Also, it is very important to support developers to develop diverse home automation services through the interoperation of heterogeneous middlewares.

In this paper, we proposed a general middleware bridge to support device interoperability on different middlewares. It enables the interoperation among heterogeneous devices and can meet the demand of home network developers for various home automation services. Unlike the existing integrated middleware architectures, IWFEngine enables the interoperability of heterogeneous devices by defining simple rules for home network services and does not require local middleware messages to be changed. Finally, we solved the overhead problem incurred by centralized integrated middlewares architectures by using distributed adaptors.

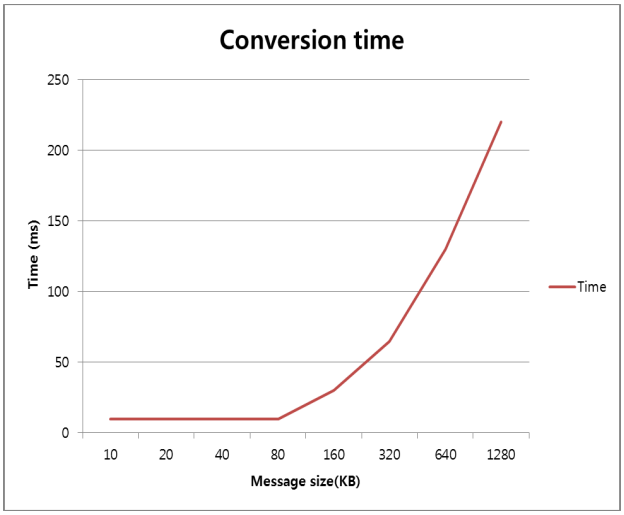

Fig. 8 Message conversion time in the IWFEngine.

We need to extend messages for the execution of home automation services under diverse environments and IWML to support extended messages as well. Also, to integrate heterogeneous home network middlewares, fault tolerance of the IWFEngine using a centralized mechanism is a major issue. A fault tolerance system which can operate when the IWFEngine does not operate, and which recognizes high-performance appliances in home networks and distributes important services to high-performance appliances should be investigated in the future. Finally, a rule builder to support developers in conveniently constructing rules and new APIs to help developers to quickly develop new middlewares adaptor are also needed.

Acknowledgments This work was supported by the IT R&D program of MKE/KEIT. [2009-F027-01, Development of Interoperable Home Network Middleware for settling Home Network Heterogeneity]

References

[1] UPnP Forum, available from ⟨http://www.upnp.org⟩.
[2] Sun Microsystems: Jini Architecture Specification, available from ⟨http://www.sun.com/jini/⟩.
[3] Konnex Association, available from ⟨http://knx.org⟩.
[4] ECHONET Consortium, available from ⟨http://www.echonet.gr.jp⟩.
[5] The Havi Organization: Havi Version 1.1 Specification, available from ⟨http://www.havi.org⟩.
[6] Echelon Co.: LonTalk Protocol Specification, Ver 3.0 (1994).
[7] Test Specification of RS-485 Protocol for Homenetwork Wallpad/ Home Gateway, available from ⟨http://www.kashi.or.kr⟩.
[8] Singh, M.P. and Huhns, M.N.: *Service-Oriented Computing*, Wiley (2005).
[9] Guillaume, B., Kumar, R., Helmut, B. and Thomas, S.: *Methods for Bridging a HAVi Sub-network and a UPnP Subnetwork and Device for Implementing said Methods*, Thomson Multimedia (2002).

© 2012 Information Processing Society of Japan

[10] Allard, J., Chinta, V., Gundala, S. and Richard III, G.: Jini Meets UPnP: An Architecture for Jini/UPnP Interoperability, *Symposium on Applications and the Internet*, pp.268–275 (Jan. 2003).
[11] Box, D.: Simple Object Access Protocol 1.1, available from ⟨http://www.w3.org/TR/SOAP/⟩.
[12] OSGI Alliance, available from ⟨http://www.osgi.org/⟩.
[13] Moon, K., Lee, Y., Son, Y. and Kim, C.: Universal Home Network Middleware Guaranteeing Seamless Interoperability among the Heterogeneous Home Network Middleware, *IEEE Trans. Consumer Electronics*, Vol.49, No.3, pp.546–553 (Aug. 2003).
[14] Kim, D., Lee, C.-E., Park, J.H., Moon, K.D. and Lim, K.: Device Conversion and Message Translation for the Home Network Middleware Interoperability, *IEEE Trans. Consumer Electronics*, Vol.53, No.1, pp.108–113 (Feb. 2007).

Hark-Jin Lee received his B.S. and M.S. degrees in computer science from Chung-Ang University, Korea in 2005 and 2007 respectively. He has been a researcher of Green Computing Research Department at Electronics and Telecommunications Research Institute, where he develops the home network middleware. His research interests include home network middleware, Linux system, and embedded computing.

Young-Sung Son received his B.S., M.S., and Ph.D. degrees in computer science from Pusan National University, Korea in 1995, 1997, and 2006 respectively. From 1997 to 1999, he worked for developing file system and VOD server of Linux clustering software at Electronics and Telecommunications Research Institute. Since 1999, he joined embedded software center for developing the home network middleware and Java embedded architecture.

Jun-Hee Park received his B.S., M.S., and Ph.D. degrees in computer science from Chung-Nam University, Korea in 1995, 1997, and 2005 respectively. He was a researcher at System Engineering Research Institute from 1997 to 1998 where he had worked on network computing and clustering system. From 1998 to 2009, he was a senior researcher at Electronics and Telecommunications Research Institute, where he had worked on home network middleware especially interoperability framework. Since 2010, he has been the team leader of Emotion-IT convergence Middleware Research Team. He has researched on Ship and ICT convergence area, and developed ship area network technology. His recent research interests are smart home and smart ship.

Kyeong-Deok Moon received his B.S. and M.S. degrees in computer science from Hanyang University, Korea in 1990 and 1992 respectively. He received his Ph.D. degree in information engineering from KAIST ICC, Korea in 2005. From 1992 to 1996, he was researcher at System Engineering Research Institute where he worked on high performance computing and clustering computing. Since 1997, he has been a principal researcher of Green Computing Research Department at Electronics and Telecommunications Research Institute, where he develops the home network middleware and Java embedded architecture. His research interests include home network middleware, Java, active network, and pervasive computing.

Jae-Cheol Ryou is a professor in the Division of Electrical and Computer Engineering at Chungnam National University in Korea. He is also the director of the Internet Intrusion Response Technology Research Center (IIRTRC), Chungnam National University, Korea. He received his B.S. degree in Industrial Engineering from Hanyang University in 1985, M.S. degree in Computer Science from Iowa State University in 1988, and Ph.D. degree in Electrical Engineering and Computer Science from Northwestern University in 1990. His research interests are Internet Security and Electronic Payment Systems including Wireless Internet Security.

© 2012 Information Processing Society of Japan