Self-Configurable Service Middleware for Pervasive Digital Home Environment

Kyeong-Deok Moon*, Young-Hee Lee**, and Chae-Kyu Kim*, Member, IEEE

*Electronics and Telecommunications Research Institutue, 161 Kajong-dong, Yuseong-gu, Taejeon, 305-360, Korea
TEL:+82-42-860-3870, Fax:+82-42-860-6671, e-mail:{kdmoon, kyu@etri.re.kr}

**Information and Communication University, 119, Munjiro, Yuseong-gu, Taejeon, 305-714, Korea
Tel:+82-42-866-6112, Fax:+82-42-866-6154, e-mail:yhlee@icu.ac.kr

Abstract — Users will be able to access ubiquitously present appliances anywhere and anytime through pervasive digital home environment. For this, we need middleware that provides a high-level abstraction and self-configuration, and guarantees the interoperability among middleware. Further, it makes each appliance's behavior context-aware, and supports a variety of adaptive behaviors. However, contemporary middlewares — HAVi, Jini, and UPnP — typically assume a static and well-administrated computing environment, and does not ensure interoperability with different middleware. This paper identifies self-configurable service middleware for the future pervasive digital home environment. It provides a desirable environment that supports adaptability and dynamic composability through appropriate high-level abstraction.

Index Terms — Self-Configurable Middleware, Pervasive Digital Home, Dynamic Composing Service, High-level Abstraction, Interoperability among middleware

I. INTORDUCTION

Over ten years ago Mark Weiser identified the goal of future computing to be ubiquitous or pervasive computing. Thus, in the near future, computing-power and communication-capability will be embedded in virtually every appliance [1]. Users will be able to access the ubiquitously present appliances anywhere and anytime. Accordingly, pervasive computing has the potential to radically transform the way people interact with computers. The key idea behind pervasive computing is to deploy a wide variety of appliances throughout our living and working spaces [2][3]. Digital home is recognized as starting point of pervasive computing by many researchers.

As people move throughout the physical world, and an appliance's execution context changes all the time in pervasive digital home environment, middleware needs to embrace contextual change. Users also expect that their appliances and applications just plug together. Thus, middleware needs to encourage dynamic composition and not assume a static computing environment with a limited number of interactions [2][3].

Since embedded appliances have limited resources and functions, they cannot provide effective services without co-operating with other appliances. The increasing diversity of appliances implies that different middleware will be in use. It is highly improbable that there will be, in the near future, single dominant middleware that would be good enough for different appliances. Pervasive service requires interoperability among middleware, high-level abstraction, zero-configuration, and context-awareness [1][2].

However, contemporary middleware — HAVi [4], Jini [5], and UPnP [6] — typically assume a static and a well-administrated computing-environment, require that applications are well-behaved, and closely couple devices to each other. Moreover, people need to adapt the system instead of the applications adapting for changes. It is not easy to develop a pervasive service to support such adaptability and dynamic composability based on currently available digital home middleware, which do not provide a sufficiently high-level abstraction to dynamically compose appliances [2][7].

Therefore, we need new middleware with which we can integrate any middleware in a simpler way, and deploy pervasive digital home services without the conscious of diversity of appliances. Also it supports adaptability and dynamic composability through appropriate high-level abstraction. To accomplish this goal, we propose new middleware guaranteeing seamless interoperability of appliances and deployment of services under heterogeneous middleware with considering contextual change. In this paper, we identify self-configurable service middleware, called Universal Home Network Middleware (UHNM), enabling seamless service provisioning in heterogeneous, dynamically varying future pervasive digital home environment.

Our proposed UHNM architecture is consisted of following components; Adaptor, Messaging Layer (ML), Event Manager (EM), Configuration Manager (CM), Resource Manager (RM), Device Manager (DM), Service Manager (SM), and Virtual Proxy (VP). An Adaptor converts the sub-middleware protocol to the UHNM protocol to guarantee seamless interoperability among heterogeneous contemporary middleware. All UHNM components communicate using a message passing mechanism through ML, and EM delivers events to the other components, when the status of networks is changed. CM serves as a directory service to provide zero-configuration as a result of the addition or removal of an appliance. The RM allows the services to reserve and release

any appliance. The DM is responsible for installing and removing VP, a virtually pre-defined appliance control module, which abstracts and represents a single appliance, and is the source of interoperability and deployability. SM invokes and deploys the context-aware and adaptive services by dynamically combining various VPs under different middleware with considering the user's location and circumstances change.

The rest of the paper is organized as follows. We discuss the related works on digital home middleware in Section 2. In Section 3, we present the design issues, and describe the details of the UHNM architecture and experiments for verifying the feasibility of the proposed UHNM in section 4. Finally, in Section 5, we conclude the paper with future work.

II. RELATED WORKS

There are several approaches to ensure the interoperability among the different home network middlewares. They are one-to-one and one-to-many protocol conversion approaches. HAVi-to-UPnP bridge at Thomson provides the interoperability between UPnP and HAVi. It solves some problems with diversification of middleware. But, these are not enough to develop a single bridge that connects two specific middleware one to one, when new middleware will be developed one after another [8].

Framework for Connecting Home Computing Middleware at Waseda University enables any appliances under any middleware's control to communicate any other appliances. It uses and deploys service of home without special conscious of diversification of middleware. The drawbacks of this are the it cannot generate services dynamically by combining any functions of any appliances [9].

Context-/Location-based Middleware for Binding Adaptation (Colomba) Framework automatically updates mobile user references to needed resources whenever a user moves, and dynamically selects and enforces the most suitable binding strategy. Colomba operates according to dynamic environmental conditions, administrator management requirements, and user profiles [7].

III. DESIGN REQUIREMENTS

In the near future, a variety of appliances will be connected to each other via wire/wireless home networks, and future pervasive self-configurable service will need to support a variety of adaptive behaviors. In this section, we present several requirements for self-configurable service middleware for pervasive digital home environment [2][3].

- Dynamic Configuration

Future pervasive digital home service will have to provide flexible interaction to control appliances because the best way to control the appliances changes according to a user's situation. Dynamic composing is very important for future pervasive services, which should provide a way to select appliances dynamically to support user mobility, and to consider circumstances change. This also makes it possible to use the same service anytime, anywhere. However, it is not easy to implement the dynamic composing service because the behavior of pervasive digital home services should be specialized according to the user's situations.

- Context Monitoring

Moreover, the mobile user of tomorrow will not appreciate a static binding between her and an access appliance. To provide adaptability, an appliance used by application need be changed according to a user's location, and circumstances change. For this, the computing environment needs to know each user's location and circumstances change to behave in a consistent way. Thus, context monitoring is one of the fundamental enablers of adaptability.

- Abstraction

Abstraction is very powerful to deal with the complexities, and heterogeneity of pervasive digital home environment. Context-aware services require accessing and binding appliances dynamically without considering the difference among appliances in pervasive digital home environment. By inserting a lot of "if" statements to check the current situation, we can build context-aware applications. However, the program is very difficult to modify when the program needs to consider another situation. Thus, we need good abstraction to build well-structured context-aware applications. Furthermore, to exploit characteristics of middleware, we need meta-level interface. However, it is not easy to export a generic interface to control underlying middleware because the generic interface usually hides some low level characteristics of the underlying middleware.

IV. DESIGN GOAL AND UHNM ARCHITECTURE

In this section, we describe the goal of UHNM architecture and details of its implementation. Also we explain the experiments conducted to verify the feasibility of our proposed UHNM.

A. Degign Goal of UHNM

As we mentioned earlier, future pervasive digital home environments need to be context-aware and adaptive. Accordingly, we need new middleware that provides a high-level abstraction and zero-configuration, and makes each appliance's behavior context-aware and supports a variety of adaptive behaviors – changes in the execution and communication capabilities, efficient use of available communication resources, and location of mobile users.

Our UHNM provides a desirable environment in which an appliance can interact with and detect other appliances under different middleware through considering the circumstances change. It also appropriately deploys certain services, which composes functions of multiple appliances, and new middleware can be easily integrated. Also it supports adaptability and dynamic composability through appropriate high-level abstraction. In this paper, we identify self-configurable service middleware enabling seamless service provisioning in heterogeneous, dynamically varying the future pervasive digital home environment.

B. Implementations of UHNM

Fig 1 shows the proposed UHNM architecture. Every appliance in the pervasive digital home is physically connected through the home network – IEEE1394, 802.11a/b, Power-Line, and Ethernet. UHNM supports several sub-middleware defined by several consortiums – HAVi, Jini, UPnP, and LonWorks. They provide the logical connection between appliances based on same middleware. Our UHNM architecture is consisted of following components; Adaptor, ML, EM, CM, RM, DM, SM, and VP. UHNM provides the seamless connection among heterogeneous sub-middleware.

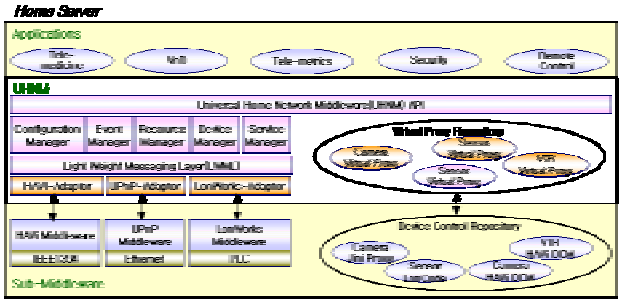

Fig 1. UHNM Architecture

- Messaging layer

All UHNM components communicate using a message passing. Although the implementation of the message passing can differ from sub-middleware to sub-middleware, the format of UHNM messages and the protocol, used for their delivery, are fully specified by UHNM to ensure interoperability. We only support reliable message transfer for guaranteeing the reception of request, since the lower sub-middleware provides a good level of transmission reliability. Source is blocked and refrained from sending a message until it receives the acknowledgement from the destinations. To avoid a dead lock and prevent replication, the ML maintains a timer, and retains message number respectively.

- Event manager

All UHNM components are activated by events from other components. EM dispatches and delivers events to the other components, when it receives the events. Thus, EM is an agent to help assure the event posted by a component will reach all components that care about the event. If a component wishes to be notified when a particular event is posted, it must register such intention with EM.

- Configuration Manager

CM serves as a directory service providing zero-configuration as a result of the addition or removal of an appliance, searching for an appliance, and detecting its capabilities and properties. CM maintains the internal table, *Conf_T*. In *Conf_T*, *M_ID* is the ID for each sub-middleware, and *SubM-ID* is the ID, allocated by sub-middleware. *Type* of VP specifies the functionalities of components or appliances; *Name* is the name of component, or appliances; *Company* is the name of the manufacturer; *Model* is the number of the model. *Rep_Loc, Rep_V, Rep_ID, and Rep_N* are VP location, version, unique ID, and the name respectively, and *Phy_Loc* is the physical location of an appliance in the home. *Net* is the media of network for this appliance; *Dev_S* is status of appliance. CM only supports queries on one attribute and its value in the *Conf_T*. Thus, CM supports searches for all devices, a particular device with a unique ID, or any devices of a specific type, manufacturer, or model.

- Resource Manager

The RM allows the services to reserve and release any appliance according to the resource availability of the home network by maintaining the internal table, *Resource_T*. In *Resource_T*, *UID* is a unique ID, *Status* is the status of an appliance, *Service* is the name of service that uses the appliance, *BW* is the bandwidth for the sending data. RM controls admission, and performs authorization for the valuable resources by protecting services from unauthorized access. In order to ensure that the service has reserved all resources, RM reserves resources in an all-or-nothing fashion, and with non-intrusive reservation. For this, resources are locked until the reservation or release process is completed, since effects of a reservation or release, which is in the middle of executing, appear invisible to other reservation and release. Device resources must be reserved by a service for exclusive use.

- Adaptor

Adaptor allows the new middleware to be integrated easily by adding a new Adaptor, which converts the sub-middleware protocol to the UHNM protocol. In order to convert the protocol, Adaptor maintains mapping tables, *Msg_Map_T* for event message and *Info_Map_T* for attributes. Adaptor abstracts the details of sub-middleware, such as information on appliances. Adaptor notifies the change of home that is managed by each sub-middleware, and UHNM. Adaptor registers the event with the manager of sub-middleware in order to be notified. When UHNM service reserves an appliance, Adaptor notifies that event to the manager of related sub-middleware in order to prevent the legacy service from trying to use it.

- Virtual Proxy

The VP, a virtually pre-defined appliance control module, abstracts and represents functions of a single appliance over the sub-middleware's control module. It is the source of interoperability and deployment in pervasive digital home environments. Through these VP objects, each service can access other appliances transparently as appliances are implemented on the same middleware. Furthermore, VP enables the dynamic composition service according to user mobility and circumstances change. VP automatically maps the UHNM command to the sub-middleware command based on the VP schema. The VP schema contains the command relationship between UHNM and each sub-middleware for providing seamless interoperability by automatically converting the command. VP schema is defined by XML to provide interoperability and deployment. It provides the en-

vironment of configuring the service dynamically with combining the functions of appliances without intervention of user. Table 1 shows VP schema.

TABLE 1. VP SCHEMA

XML Documentation	Description
<?xml version="1.0"?>	•xml : required for all XML documents
<serviceScript>	
<specVersion>	
<major>1</major>	•minor : major version
<minor>0</minor>	•minor : minor version
</specVersion>	
<script>	
<applianceList>	•applianceList : list of appliances needed by service
<appliance>	•appliance : repeat once for each appliance
<type>applianceVPtype</type>	•type : VP type of appliance
<toList>	•toList : list of destination appliances
<toAppliance>	•tpAppliance : description for destination appliance
<toType>applianceVPtype </toType>	•toType : VP type of destination appliance
<rule>	•rule : definition of composition rule between this appliance and destination appliance
<cond>evalution</cond>	•cond : describing the status for dynamic combining
<composition>reconfigure or not</composition>	•composition : describing whether performs composition or not
</rule>	
Declarations for other rules	•describe the other composition rules
</toAppliance>	
Declarations for other destination appliances	•describe the other destination appliances
</toList>	
<fromList>	•fromList : list of source appliances
<fromAppliance>	•fromAppliance : description for source appliance
<fromIType>applianceVPtype </fromIType>	•fromIType : VP type of source appliance
<rule>	•rule : definition of composition rule between this appliance and source appliance
<cond>evalution</cond>	•cond : describing the status for dynamic combining
<composition>reconfigure or not</composition>	•composition : describing whether performs composition or not
</rule>	
Declarations for other rules	•describe the other composition rules
</fromAppliance>	
Declarations for other source appliances	•describe the other source appliances
</fromList>	
</appliance>	
Declarations for other appliances	•describe the other appliances
</applianceList>	
</script>	
</serviceScript>	

- Device Manager

The DM is responsible for installing and removing VPs using the appliance type of *Conf_T* of CM, *Rep_T* – consisted of the type of UHNM and location of VP schema based on URL – of DM, and ID of the appliance. DM automatically installs the VP when an appliance is added. For this, DM automatically determines the appropriate type of VP for new appliances according to the type of UHNM. Then, DM downloads that VP, and finally allocates the unique ID to the VP.

- Service Manager

The SM invokes and deploys the services, and provides the environment for dynamically composing functions of appliances according to user's location and circumstances change. To enhance adaptability in digital home, each appliance is designed independently and the service requirements may be specified separately. For this, our middleware lets service providers express composition strategies at a high-level that are cleanly separated from service code; changes in composition strategies thus require no intervention in the application logic. SM maintains the internal table, *Service_T*, which is composed of *UID* for unique ID, *Status* for state of service, *Name* for the name of service, and *SVR_SCPT* for service script that describes the list of appliances used by service, and composition strategy for context–aware and adaptive service. Service script is defined by XML to provide interoperability and deployment. Table 2 shows the service script schema.

SM request RM to reserve needed appliances with a service script in order to acquire all appliances. The SM allows the adaptation service to dynamically combine appliances based on composition strategy as a result of user mobility, and changes of digital home. For this, SM utilizes information on appliances and circumstances change from CM, the list of appliances used by the service and composition strategy from service script, and user's location from user.

TABLE 2. SERVICE SCRIPT SCHEMA

XML Documentation	Description
<?xml version="1.0"?>	•xml : required for all XML documents
<serviceScript>	
<specVersion>	
<major>1</major>	•minor : major version
<minor>0</minor>	•minor : minor version
</specVersion>	
<script>	
<applianceList>	•applianceList : list of appliances needed by service
<appliance>	•appliance : repeat once for each appliance
<type>applianceVPtype</type>	•type : VP type of appliance
<toList>	•toList : list of destination appliances
<toAppliance>	•tpAppliance : description for destination appliance
<toType>applianceVPtype </toType>	•toType : VP type of destination appliance
<rule>	•rule : definition of composition rule between this appliance and destination appliance
<cond>evalution</cond>	•cond : describing the status for dynamic combining
<composition>reconfigure or not</composition>	•composition : describing whether performs composition or not
</rule>	
Declarations for other rules	•describe the other composition rules
</toAppliance>	
Declarations for other destination appliances	•describe the other destination appliances
</toList>	
<fromList>	•fromList : list of source appliances
<fromAppliance>	•fromAppliance : description for source appliance
<fromIType>applianceVPtype </fromIType>	•fromIType : VP type of source appliance
<rule>	•rule : definition of composition rule between this appliance and source appliance
<cond>evalution</cond>	•cond : describing the status for dynamic combining
<composition>reconfigure or not</composition>	•composition : describing whether performs composition or not
</rule>	
Declarations for other rules	•describe the other composition rules
</fromAppliance>	
Declarations for other source appliances	•describe the other source appliances
</fromList>	
</appliance>	
Declarations for other appliances	•describe the other appliances
</applianceList>	
</script>	
</serviceScript>	

C. *Its Experiments*

In order to verify the feasibility of the proposed middleware, we've implemented an event-based surveillance service using a HAVi-camera and HAVi-display on IEEE1394, a LonWorks-motion sensor on PLC, and a Jini-display on TCP/IP. Event-based surveillance service monitors the intrusion with LonWorks-motion sensor. If someone broke into the house, LonWorks-motion sensor notifies the event. Then, HAVi-camera would send the captured-image to display appliance nearest to user immediately upon receiving an event from the LonWorks-motion sensor.

UHNM and several sub-middlewares are executed over home server, which is the central of home network, and manages and controls the appliances on the home network without user intervention. We have implemented our home server with JDK1.3.1 and Linux. A user point PDA to select the room where he is. Then SM dynamically combines the HAVi-camera and the display-appliance nearest user that receives and displays the captured-image from HAVi-camera. If a user move to different room, he switches the location by pointing PDA. As soon as SM receives the change of user's location, existing interactions between the appliances will be hand over to the new display-appliances resulting in uninterrupted use of service according to change of user's location. Fig 2 shows the test-bed.

This experiment shows that UHNM provides the context-aware and adaptive service with considering the user's location and circumstances change through dynamically combining. Furthermore, UHNM provides users with single image view of digital home, and an appliance can communicate with other appliances under different middleware.

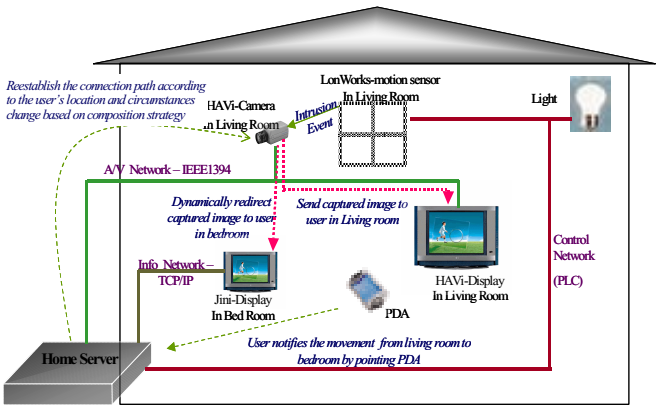

Fig 2. Testbed for Self-Configurable Service over Pervasive Digital Home

V. Conclusions and Future Works

In this paper, we present the details of UHNM and its implementations. UHNM architecture provides zero-configuration, and high-level abstraction, and makes each application's behavior context-aware and adaptive. Therefore, UHNM enables the deployment of home network services, and dynamically creates new services by combing the functions of appliances without being limited by the middleware. Moreover, UHNM architecture ensures seamless interoperability among heterogeneous home network middleware, and provides scalability by simply adding an adaptor to new middleware.

In the near future, the explosion of mobile appliances will have conditioned users to expect access to services anytime and anywhere. We extend our prototype UHNM to the mobile Internet environment with automatically determining the location of users and appliances. In present, UHNM only supports the simple composition by considering the user's location and circumstances change – appliance failure, or input from sensors, but not consider the user preference or user behavior. Therefore, we will include mobile agent technology in UHNM to provide enhanced adaptation service by considering user preferences and user behavior.

References

[1] Y. Rasheed, J. Edwards, C. Tai, " Home Interoperbility Frameworks for the Digital Home," Intel Technology Journal, November 2002

[2] K. Raatikainen, H. B. Christensen, T. Nakajima, " Application Requirements for Middleware for Mobile and Pervasive System," Mobile Computing and Communications Review, Volume 6, Number 4, 2002

[3] R. Grimm and B. Bershad, " System support for Pervasive Applications," LNCS 2584, pp.212-217, 2003

[4] The Havi Organization. "Havi Version1.1 Specification". http://www.havi.org.

[5] Sun Microsystems. "JINI Architecture Specification". http://www.sun.com/jini/.

[6] UPnP Forum. "Universal Plug and Play". http://www.upnp.org.

[7] P. Bellavista, A. Corradi, and R. Montanari, " Dynamic Binding in Mobile Applications," IEEE Internet Computing, March 2003

[8] B. Guillaume, R. Kumar, B. Helmut, and S. Thomas, " Methods for bridging a HAVi sub-network and a UPnP sub-network and device for implementing said methods," Thomson Multimedia, 2002

[9] E. Tokunaga, H. Ishikawa, M. Kurahashi, Y. Morimoto, and T. Nakajima, " A Framework for Connecting Home Computing Middleware," ICDCSW'02, 200

KyeongDeok Moon received the B.S. and M.S. degrees in computer science from Hanyang University, Korea in 1990 and 1992 respectively. 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 senior researcher and a member of Control Software Research Team at ETRI, where he develops the home network middleware and Java embedded architecture. He has been PhD student at ICU since 1998. His research interests in home network middleware, Java, active network, and pervasive computing

YoungHee Lee received the B.S. and M.S. degrees in EE from Seoul National University, Korea in 1976 and 1980 respectively. He received Ph. D degrees in computer science from Universite de Technologie de Compiegne, France in 1984. From 1984 to 1997, he had been working at ETRI as a various research team leader including as a director of protocol engineering center. He was invited scientist at IBM T. J. Watson Research Center from 1986 to 1987. From 1998 to 2002, he was the dean of school of engineering in Information and Communications University, Korea and now he is a professor at ICU. From 1994, he was serving as a vice chairman of ITU-T SG7, and chairman of WP3. His research interest is in Internet protocols, active networking and network supports for pervasive computing.

Dr. Chaekyu Kim received BS degree on mathematics from Korea University, Seoul, Korea, in 1978, the MS degree on computer science from Univ. of Technology Sydney, Australia, in 1993, and the PhD degree on computer science from the Wollongong University, Australia, in 1997. He has worked for ETRI since 1980, and currently as a principal researcher and the vice president of Computer & Software Research Laboratory in ETRI, where he develops the real-time OS, and home server architecture. His recent interests include Internet Appliance, RTOS, Multimedia, Database, e-business, and so on.