

Middlewares for Home Monitoring and Control

Long Nguyen Hoang
Helsinki University of Technology
silver@cc.hut.fi

Abstract

Our home today is a place where more and more digital devices meet. Those devices not only satisfy the user's needs but also cooperate with other equipments independently to provide intelligent services for people living inside the house. However, in such environment with many appliances from different manufactures, it may not be straightforward to interconnect all the devices. What is required is common middleware. The problem is that we do not have just one but many kinds of middleware (e.g., Jini, UPnP, oBix) whose domains are distinct or overlap each other. This paper investigates the role and common characteristics of home automation middleware, focusing on current key middlewares which are suitable for home network monitoring and control. Options for building middleware for home networks are also discussed in form of a comparison.

KEYWORDS: Middleware, home networking, automation.

1 Introduction

Consumer electronic appliances such as PVRs, HVACs, and home security systems are sophisticated and expensive digital processing systems. The current trend in home networks is to make use of these devices by connecting them in order to build intelligent connecting services. For instance, an alarm clock should turn on the light at 7:00AM then activate the coffee maker and so on. It also means that the processing and storage resources can be controlled and shared between members of family. This is achieved by using a software system called "middleware", which allows connected information appliances to exchange both control information (home automation) and streaming multimedia content. In the other words, middleware is software solution that makes interaction between underlying hardware and application software smooth by providing an abstract layer.

The current key middleware products are OSGi[3], UPnP[1], Jini[2], VESA, HAVi, OpenCable, X10, and LonWorks[6]. However, not all kinds of middleware can be applied in home networking monitoring and control. Particularly, middlewares for home network can be grouped into two categories: one specialized in control and automation system (light control, heating, auto cooking and so on) and one for media control and streaming. The focus of this report is limited to the former group in the context of home automation. The rest of the paper is structured as follows. Section 2 of this text raises roles and common characteristics of middleware in home network. In section 3, we investigate main-

streams to enable home automation using middleware. After that a comparison of current key middlewares (mentioned in the previous section) is made in section 4. Finally, the conclusions are formulated in section 5.

2 Roles and characteristics of middleware in home network

Middleware eases interchange of information in a distributed, multi-vendor, and heterogeneous environment while keeping the same levels of security, reliability, and manageability. Broken down to its simplest form, home networking middleware is the software layer that lies on top of an information appliance operating system and exposes APIs to which home networking applications can be attached. The principle of major roles of home middleware [9] [10] [12] is described in the following. These roles can be considered key points for asserting a home network middleware.

- Addressing: Each of devices is addressed by the middleware so that other nodes know where to reach it. The purpose of addressing is mainly to identify the device and to indicate its virtual location within the domain. The address is usually allocated and given automatically in the beginning when the device is plugged into the network.
- Discovery: The middleware should provide mechanism(s) to enable nodes to search for other nodes and their services. In order to do that, the middleware maintains a directory list of all nodes within its domain. IP-based middlewares make use of the discovery service of home IP network while non-IP-based ones (e.g, X10 and LonWorks) define their own discovery schemas.
- Control: The main role of home network middleware is certainly to control devices and services. The action of the system can be done manually or automatically via a user control interface. User-interface management can be provided as an additional function but it is not mandatory.
- Eventing: Whenever a change of network device status happens, the middleware should be informed and take corresponding action(s). Middleware model should specify a way to send the events to expected subscribers.
- Description: Information about available devices and their services (known as meta data) should be described

in a formal way. The middleware specifies the description format and its semantics. For instance, OSGi uses a meta-data file packed in .jar file to describe services while oBIX uses WSDL to do that.

- **Presentation:** On top of that, the middleware exposes a set of API for catching status announcement and input for devices.

Home network, namely, is not such a large one but contains various types of devices. A middleware for home automation itself inherits characteristics from general middleware [10]; there are five important attributes of home network middleware described below.

- **Configurability:** The middleware lets user adapt its infrastructure to an application's real needs. Components of middleware are usually separately configurable so that the targeted middleware can adapt to particular environment without any difficulty. In the other words, features and capabilities of applications can be modified without change to the technical architecture.
- **Genericity:** The middleware should have ability to adapt to a large variety of distribution models. This is a technical challenge because existing models are built on top of different types of hardware and various communication protocols.
- **Extensibility:** The middleware enables communication between components built on top of different distribution models. Typically, the communication is translated either statically or dynamically point-to-point from one distribution model to another. The implementations of extensibility feature are usually hardware devices such as adapters, bridges, and gateways.
- **Invisibility:** Good middleware is invisible. The better it works, the less people need to notice it. In fact, what users need is they plug devices into the network and those devices are supposed to work together. Zero configuration can be considered a part of invisibility.
- **Human factor:** The aim of home network middleware is to comfort human life. This means that human factors should be taken into account. For example, if an environment detects that a user and his girl friend are together in a room, it is desirable to automatically make the room's lighting more romantic.

3 Approaches in building home network middleware

There are a lot of middleware standards built for home networks. The middlewares investigated in this section are current key middlewares in industry and academic research. Each approach has its advantages and challenges.

3.1 The electrical-standard approach

Regarding low level protocols for resource connectivity, electrical standards have been widely used for a long

time. Representatives of this approach are LonWorks[6] and X10[5]. Basically, these standards define electrical specifications and do not provide any high level service.

LonWorks is chosen for investigation in this paper due to its notable features. LonWorks is an important solution for control networks which is developed for lowering the cost and design complexity of distributed control systems. The protocol itself is media-independent, allowing LonWorks devices to communicate over any physical transport media. Essential elements in LonWorks middleware system include *LonWorks protocol; the implementation of the protocol* (usually packed in one single chip called Neuron Chip); and *network components*.

LonWorks protocol (also know as *LonTalk protocol* and *ANSI/EIA 709.1 Control Networking Standard*) is used in communication between LonWorks devices. The design of the protocol follows Open Systems Interconnection (OSI) model. The protocol provides two sets of communication services: the first one enables application program(s) in one device to communicate with other devices over the network, the second one supports network management services including addressing reconfiguration and controlling device application programs (start/stop/reset), etc. As depicted in Figure 1, LonWorks is unique in that it is the only control protocol implementing all seven layers of the OSI model.

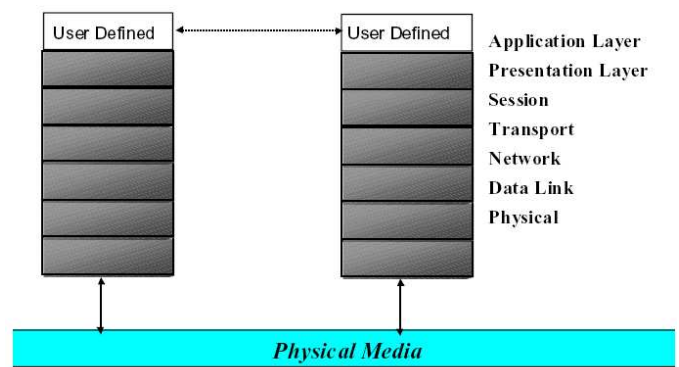


Figure 1: Application using LonWorks protocol stack [6]

In LonWorks network, each device is given a unique MAC address and has one description file to describe itself. The address can be one of four types: globally-unique Physical Address (or Neuron ID); Device Address assigned when the device is installed into a particular network; Group Address for a logical collection of devices; and finally Broadcast Address for all devices in a subnet or within a domain. The description file (with .XIF extension) is for self-documenting so that any LonWorks Network Service (LNS) server can obtain all the information needed to connect the device into the system and to configure and manage that device.

For information exchanging, LonWorks introduces the concept of network variable. A network variable is any data item (e.g., temperature, light switch state) that can be either "input" network variable or "output" network variable. In a typical transaction, the application program simply passes the new value of output network variable to the device firmware then the value will be automatically carried to the correct device(s) expecting that network variable. The application program thus doesn't need to know anything about

the departure and destination of network variables. The process of fragmenting the value into network packages, carrying the packages via LonWorks network then reforming them in the corresponding nodes is called binding, which takes place during design time and installation.

Neuron Chip is the commercial implementation of LonTalk protocol. Neuron Chip operates as both a network communication processor and an application processor, it simply leaves the execution of the LonWorks protocol to the device's application. As a result, Neuron Chip helps to standardize implementation and make development plus configuration relatively easy. Although the protocol is open to anyone, the most cost-effective manner to implement the LonWorks protocol continues to be by purchasing a Neuron Chip.

Network components: In LonWorks middleware, beside the devices themselves, there are network control components: transceiver, LNS(LonWorks Network Service), router and gateway. A *transceiver* is attached to each network device to provide a physical communication interface between that device and the LonWorks network. Multiple LonWorks devices with the same type of transceiver can form a physical communication media called channel. *LNS* operates as the network management component in LonWorks. *Router* is a network-transparent device that is used for interconnecting different channels of the network, controlling network traffic, and/or partitioning different sections in the network. Finally, *gateway* is a protocol translator which allows proprietary legacy control systems (non-LonWorks systems) to be interfaced to LonWorks systems.

LonWorks continues to prove its leading position in home automation solutions. Its challenge is the interconnection with other kinds of network. The gateway can not communicate with all types of non-LonWorks devices (e.g, media boxes, streaming contents). Besides, LonWorks is not a high level solution, which somehow reduces flexibility. However, LonWorks is still the first choice for monitoring and controlling small-to-medium size families.

3.2 The open-standard approach

The representatives of this solution are the open-standard technologies UPnP, OSGi and oBIX. This approach is better than using bridge/gateway since it reduces complex connections or any one-to-one converter. Moreover, this choice is more accepted and widely deployed in real world.

3.2.1 UPnP

"*Universal Plug and Play (UPnP) is an initiative to bring easy-to-use, flexible, standards based connectivity to consumer networks, whether in the home, in a small business, or attached to the global Internet*" [1, UPnP]. In the context of home network middleware, UPnP is a suite of protocols and system services designed to enable the simultaneous control of multiple networked home appliances. UPnP does not specify the APIs applications which will be used, it lets operating system vendors create their own APIs instead. UPnP is designed to be independent of any particular operating system, programming language, and physical medium. It

makes use of TCP/IP standard and Internet protocol [8], enabling it to seamlessly fit into existing networks. The strong point of UPnP is that any device can dynamically join an UPnP network, obtain an IP address, convey its presence and capabilities, then learn about the presence and capabilities of other devices. One typical UPnP middleware environment includes the following components:

- **UPnP device:** a container of services. Services and properties associated with the device are described in a device-description document in XML format following "UPnP Device and Service Descriptions"[1]. That file is then hosted by the device and can be accessed via URL.
- **Service:** The smallest application unit in UPnP network is a service. A service in a UPnP device exposes actions (or operations) and manages its state variables. It consists of a state table, a control server and an event server. The state table maintains the status (variables) of the device. The control server receives action requests, executes them, updates the state table then returns responses. Finally, the event server publishes events to interested subscribers whenever the state of the service changes. It is noted that the device itself can host several services within.
- **Control point:** A control point in a UPnP network is responsible for discovering and controlling other devices. A control point is also used for invoking services or subscribing to the service's event source. One physical device in UPnP network can operate as either a service unit or a control point or both.
- **Bridge:** A bridge allows legacy information appliances to communicate with native UPnP information appliances. This enables UPnP to interconnect to other UPnP networks and non-UPnP networks as well.

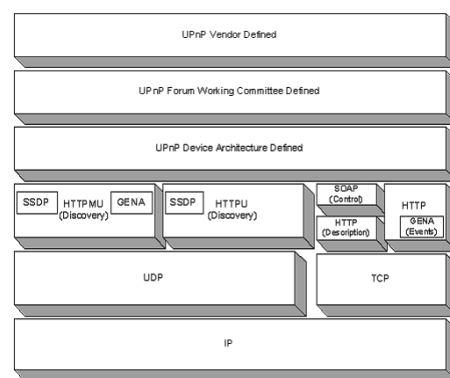


Figure 2: UPnP protocol stack [1]

UPnP leverages many existing standard protocols to improve interoperability between different vendor implementations. The UPnP protocol stack is shown in Figure 2. The TCP/IP protocol stack and HTTP are core parts of UPnP. Simple Service Discovery Protocol (SSDP) defines how control points can locate resources in the network and how devices can advertise their availability as well as their services.

Generic Event Notification Architecture (GENA) was defined to provide the ability to send and receive notifications using HTTP over TCP/IP and multicast UDP. Simple Object Access Protocol (SOAP) is used to execute remote procedure calls. UPnP device, by default, uses AutoIP for addressing mechanism (i.e, to obtain an IP address when joining the UPnP network); DHCP is preferred if there is one DHCP server in the network.

Currently, UPnP is one of the most used middleware. The idea of peer-to-peer network and common standards significantly increases the flexibility attribute. Although nodes in UPnP are not required to be PCs and specific platform, practical issues show that UPnP nodes have to have enough resource and capacity to run the UPnP protocol stack, also to parse the complex XML-based request/response (i.e. high footprint). However, with the increasing capacity of home appliances nowadays, these obstacles can be easily bypassed.

3.2.2 Jini

Jini [2] is a distributed systems platform based on Java. Jini adds to Java and JavaRMI other services which are useful for complex distributed applications (naming, lookup and discovery services, an eventing paradigm, distributed garbage collection and a transaction scheme). One typical Jini middleware environment is comprised of the following components: Jini services, Jini clients and at least one Jini Lookup Service (LUS).

- **Jini services:** Jini service is element playing as a front-end of network resource, producing service descriptions and representing services. Services in Jini are required to register themselves in at least one Jini Lookup Service before being acquired and used.
- **Jini clients:** Jini client is client consuming services. Jini clients search for a given service in Jini Lookup Service(s). Once a positive matching occurs, the client downloads a proxy (Java bytecode representing the service's front-end) of the service from the Jini Lookup Service, and executes it in client's JVM with the help of JavaRMI mechanism.
- **Jini Lookup Service:** A Jini Lookup Service is equivalent to a directory or index of available services, where proxies to those services are stored. Consequently, it would be possible to obtain the service description of the resource even when the resource is temporarily out of service or disconnected. This brings strength to flexible discovery mechanism of Jini.

Jini is most well known for discovering plus registering resources and services. Currently there are no Jini-compliant physical resources to be used in environments such as smart homes. Therefore, other available resources which belong to different standards should be integrated in present and near-future applications. It is because of its resource consuming and complex negotiation of JavaRMI. In spite of that, Jini is considered a potential and suitable platform for the integration of low-level protocols and discovery mechanisms.

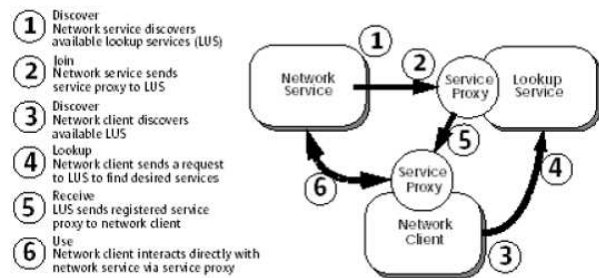


Figure 3: Jini component interaction [17]

3.2.3 OSGi

OSGi[3] is a specification which provides an open standard for remote programmable devices. OSGi specification works with various device access standards and it is compatible with other kinds of middleware such as LonWorks[6], Jini[2] and UPnP[1]. OSGi facilitates the installation and operation of multiple services on a single service gateway. The gateway can be any of capable devices from set-top-box or DSL modem or PC to dedicated residential gateway. The gateway platform execution comprises the downloading of software, application life cycle management, security of the programming environment, management of device drivers for external devices, configuration management, user management and a remote administration model. Key entities in OSGi framework are services and bundles:

- **Services:** Services in OSGi are actually Java classes that perform certain operations. One OSGi service has a Java interface and its separated implementation. In OSGi model, an application is built around a set of co-operating services. For example, a text editing application designed this way may rely on a spell-checking service. The editor then would instruct the framework to download a spell-checker for it to use. Services are queried and called by using LDAP services and the dependencies among them are managed by the OSGi framework.
- **Bundles:** To be available to the framework, implementations of a service must be packaged into one bundle. A bundle in .JAR file contains resources implementing zero or more services and a Manifest file for description so that the framework can correctly install and activate the bundle. These bundles are then published in a provider site. After that, they can be downloaded to a typical gateway to provide the desired services.

As Figure 4 shows, specific middleware layer is placed on top of heterogeneous discovery layer in OSGi architecture. This is somehow up-side-down compared to the bridge/gateway architecture. In fact, this architecture model ensures interoperability among heterogeneous middlewares. Using the provided APIs, end-users can load services on demand regardless of their middleware-specific provider. The gateway itself manages the installation, versioning and configuration of these services. Once present in an Open Services Gateway, these services may be accessed by all connected devices and networks in the home.

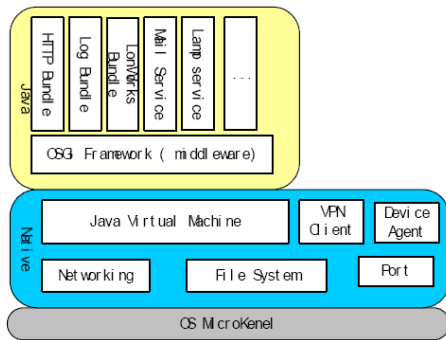


Figure 4: OSGi protocol stack [3]

The problem of OSGi now might be the interaction between distributed OSGi. This is because OSGi was designed for long-running appliances, not for enterprise servers. OSGi assumes that each JVM has a unique, persistent set of bundles. Moreover, system developers usually face the problem of making code modular and the security model in OSGi specification is not flexible enough to handle complex inter-module security requirements. These problems are hardly realised in design time, usually they appear when deploying bundles. However, the effective design in a way that everything is simply packed inside the gateway makes OSGi one of the most popular and effective middlewares. There are very good bundles provided in the OSGi community and we can download then use them freely (e.g., HTTP service, Log service). The next version of OSGi specification which will be published soon is expected to be stronger and more flexible.

3.2.4 oBIX

oBIX (Open Building Information Xchange) [4] is a specification which aims to facilitate the exchange of information between intelligent buildings and to enable enterprise application integration. The oBIX specification takes advantage of XML standard and Web Services specifications. Particularly, oBIX defines a publicly available web service interfaces that can be used to obtain data from building automation systems, and to provide data exchange between the systems and enterprise applications. Overall architecture of an oBIX system is described in Figure 5.

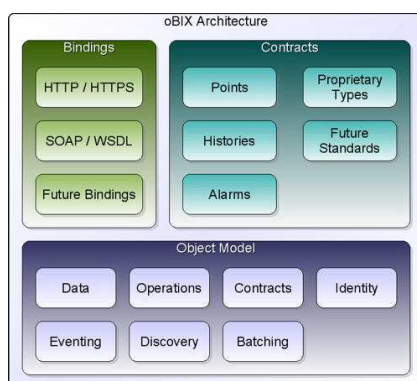


Figure 5: oBIX Architecture [4]

Object Model layer defines primitive blocks to model the semantics of behaviour of the systems. On top of that, the

Contract block is defined for the purpose of improving extensibility. A contract is a list of all patterns to which a complex piece of data conforms. Contracts are used to describe standardized structures such as points, historical trends and alarms; they are also used to describe proprietary vendor data. The main point of the contract is that new ones can be introduced without changing the oBIX schema.

In oBIX, objects (web services with their states) are identified by URLs for being transferred using Hyper Text Transfer Protocol. oBIX uses SOAP binding to leverage WS-* [11] and HTTP REST standard. oBIX servers can be accessed via a web browser thus their services can be indexed by search engines, linked to by other web pages. The result is that provided services are accessible over the networks and, moreover, basically compatible with any other web technologies.

The role of web service in oBIX is obviously to guarantee interoperability of home appliances. Physical components such as sensors and actuators aren't required to understand Web Services, but their controllers are, as well as new devices entering the home network. oBIX gateways also need to understand web services in order to communicate with world outside and coordinate external services. Due to the increasing capacity of home appliances, the use of innovative WS-* specifications and its ability to integrate existing middlewares, oBIX is considered the next-generation middleware for home automation.

3.3 The bridge/gateway approach

Not all kinds of middleware take into account the interoperability problem. If home appliances in the same house use different middlewares, one cannot know others' existence or use functions of each other. One solution to enable interoperability among heterogeneous middleware technologies is to use Middleware-to-Middleware (M2M) bridges. Most of bridges are point-to-point functional converter between two specific middlewares. Little practical work has been achieved on providing a unified and interoperable translator the different architectures. Fortunately, some middlewares provide their native bridges whose covering domains are large enough. The UPnP/non-IP bridge [16] is one example of this.

We can also find some hybrid solutions [13] [14] [15] proposed in publications or laboratory works. The advantage of this kind of combination is that the proposed middleware is usually highly optimized for one or a couple of areas. The bridge/gateway is typically embedded inside the middleware architecture. For instance, [14] on the adaptive framework while IEEE1394/IP focus on making use of IEEE1394 strong points and the advantages of IP network at the same time. However, implementations of them are only within laboratory works and they are not open standards.

4 Comparison of current options to enable home automation using middleware

Table 1 shows the feature comparison of all middlewares mentioned in the previous sections. All of the investigated

Middleware	LonWorks	Jini	UPnP	OSGi	oBIX
Network base	powerline	independent	independent	independent	independent
Platform	independent	Java	independent	Java	independent
Addressing	MAC	IP	IP	IP	IP
Service discovery	N/A	LUS/JavaRMI	SSDP	LDAP	under development
Service description	XIF	proxy object	XML	XML	WSDL
Control point	LNS	directly	control point	gateway	Web server
Service unit	application	service	service	bundle	Web service

Table 1: Feature comparison of examined middlewares

middlewares were designed to be platform independent (including Java) and mostly satisfy what are stated in section 2. We can see there is a significant degree of overlap in the application domain among these standards. However, if we focus on one aspect of device interworking such as device discovery, we find that each of them has its unique technology to address the issue, bringing strength to its target domain. LonWorks, Jini and UPnP are preferred for connecting as well as controlling indoor appliances for their simplicity in user perspective. In the other hand, if the host wants to cooperate existing middlewares (e.g., LonWorks, UPnP) not only inside his house but among interconnected home environments then OSGi is what he needs in current time. oBIX is a potential one but still underdevelopment and not fully supported yet. The choice also depends on the user's cognition and knowledge.

Home networks in future will consist of multiple networking and platform technologies, integrated through a series of gateways and shared devices. Therefore, high-level standards will be preferred; or low-level ones should provide flexible solutions for the integration issue.

5 Conclusion

Home networks are just about to enter millions of private homes. They will connect a large variety of devices including home appliances, consumer electronics, white goods, and telecommunication devices. In this paper we have attempted to claim general roles and characteristics of middleware in home network, then investigated current key middlewares for home automation. In addition, we present a feature comparison of them, also a brief guide for selecting middlewares. Still, there are several issues in home network middleware field that need to be solved. Convergence is one of them, though many bridge/gateway solutions have been introduced in the effort to extend the target domain. Smooth transition from one middleware to another is generally impossible so far. Distributed home middleware is also a problem as the view of "home" is changing. However, in our opinion, the most important issue is not about technical but human factor - the ultimate target of home middleware. The technology setting off human life the most will win the competition. In this context, there should be disappearing middleware in user's perspective, as Weiser's vision of disappearing technology "*The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.*" [7].

We believe intelligent and automated home which enables a better environment for members of the family is not so far.

References

- [1] Universal Plug and Play Forum UPnP Device Architecture. *UPnP Home page* http://www.upnp.org/download/UPnPDA10_20000613.htm
- [2] Sun Microsystems, Inc. Jini specifications v1.2 *Jini Network Technology - Specifications* <http://www.sun.com/software/jini/specs/>
- [3] OSGi Alliance Open Service Gateway initiative specification v1.0 *OSGi Service Gateway Specification - Release 1.0* http://osgi.org/osgi_technology/download_specs2.asp?section=2
- [4] OASIS Open Building Information Exchange (oBIX) Committee Specification 01, 5 December 2006 *Committee Specification 01, 5 December 2006* http://www.oasis-open.org/committees/documents.php?wg_abbrev=obix
- [5] X10. X10 Powerline Carrier (PLC) Technology *X10 Home Automation homepage* <http://www.x10.com/support/technology1.htm>
- [6] Introduction to the LonWorks system Echelon homepage - General Manuals <http://www.echelon.com/support/documentation/manuals/general/078-0183-01A.pdf>
- [7] M. Weiser The computer for the 21st century in *Human-computer interaction: toward the year 2000*. Pages: 933 - 940. Year of Publication: 1995. ISBN:1-55860-246-1.
- [8] Postel, J., Internet protocol. *RFC 791 Internet protocol* September, 1981.
- [9] Amit Dhir. *Home network middleware* March 21, 2001. <http://direct.xilinx.com/bvdocs/whitepapers/wp136.pdf>
- [10] Tatsuo Nakajima, Kaori Fujinami, Eiji Tokunaga, Hiroo Ishikawa *Middleware Design Issues for Ubiquitous Computing*

- [11] IBM developerWorks *Standards and Web Service WS-* standards* <http://www-128.ibm.com/developerworks/webservices/standards/>
- [12] State of Connecticut *Middleware Domain Technical Architecture* September 15, 2002 State of Connecticut www.ct.gov.
- [13] Eiji Tokunaga, Hiro Ishikawa, Makoto Kurahashi, Yasunobu Morimoto, Tatsuo Nakajima *A Framework for Connecting Home Computing Middleware* 2002 IEEE Proceedings of the 22 nd International Conference on Distributed Computing Systems Workshops.
- [14] Eiji Tokunaga, Hiro Ishikawa, Makoto Kurahashi, Yasunobu Morimoto, Tatsuo Nakajima *A Java-based home network middleware architecture supporting IEEE1394 and TCP/IP* IEEE Transactions on Consumer Electronics, Vol. 48, No. 3, AUGUST 2002.
- [15] Kyeong-Deok Moon, Young-Hee Lee, Young-Sung Son *Universal Home Network Middleware (UHNM) Guaranteeing Seamless Interoperability among the Heterogeneous Home Network Middleware for Future Home Networks* IEEE.
- [16] Vijay Dhingra *How to connect non IP devices into the UPnP.v1 fabric* UPnP forum <http://www.upnp.org/download/summitslides2003/01-13Echelon.ppt>
- [17] Antonio F. Gomez Skarmeta *Identification of IPv6-Enabled Devices to be Used in Home Automation* UMU www.6power.org http://www.6power.org/open/6power_pu_d4_8_v2_5.pdf