Adapting Legacy Home Appliances to Home Network Systems Using Web Services

Masahide Nakamura, Akihiro Tanaka, Hiroshi Igaki, Haruaki Tamada and Ken-ichi Matsumoto

Graduate School of Information Science, Nara Institute of Science and Technology, JAPAN
{masa-n, akihir-t, hiro-iga, harua-t, matumoto}@is.naist.jp

Abstract

This paper presents a framework that adapts the conventional home electric appliances with the infrared remote controls (legacy appliances) to the emerging home network system (HNS). The proposed method extensively uses the concept of service-oriented architecture to improve programmable interoperability among multi-vendor appliances. We first prepare APIs that assist a PC to send infrared signals to the appliances. We then aggregate the APIs within self-contained service components, so that each of the component achieves a logical feature independent of device(or vendor)-specific operations. The service components are finally exported to the HNS as Web services. Thus, the legacy appliances can be used as distributed components with open interfaces. To demonstrate the effectiveness, we also implement an actual HNS and integrated services with multi-vendor legacy appliances.

Keywords: home network system, infrared control, service-oriented architecture, integrated services, legacy migration

1 Introduction

The emerging technologies enable general household appliances to be connected to LAN at home. Such smart home appliances are generally called *networked appliances*. A *home network system* (HNS) consists of multiple networked appliances, intended to provide more convenient and comfortable living for home users. Research and development of the HNS are currently a hot topic in the area of ubiquitous/pervasive computing. Several HNS products are already on the market (e.g., [4][11][14]).

The HNS provides many applications and services. The applications typically take advantage of wide-range control and monitoring of appliances inside and outside the home. Moreover, integrating different appliances via network yields more value-added and powerful services [7], which we call *HNS integrated services*. For instance, orchestrating a TV, a DVD player, 5.1ch speakers, lights, curtains and an air-conditioner implements an integrated service, say *DVD theater service*, where a user can watch movies in a theater-like atmosphere.

In general, each networked appliance is equipped with smart embedded devices, including a network interface, a processor and a storage, in order to provide and execute the appliance features required for various HNS applications and services. As the embedded devices become more downsized, cheaper, and more energy-saving, it is expected in the near future that every object will be networked [3].

However, transition to the networked appliances is gradual. Most people are still using *legacy appliances*, which are the conventional non-networked home appliances, although it is usual to see a network (and PCs) at home. There are several reasons why the networked appliances are not spread yet. First, the networked appliances are yet expensive. Also, types of available appliances are limited. Due to the interoperability problem, the integration of appliances are limited especially in the multi-vendor environment. Moreover, there is a major requirement that the users want to keep using the legacy appliances that they are accustomed to use. Thus, it is not easy for the general home users to renew immediately all the existing legacy appliances with the networked ones.

To cope with both the emerging HNS and the legacy appliances, this paper presents a new framework that adapts the legacy appliances to the HNS. The proposed method extensively introduces the concept of *service-oriented architecture* (SOA) [10][13].

Focusing on the legacy appliances with the conventional *infrared remote controllers* (denoted by *IrRC*), we first develop a set of APIs, called *Ir-APIs*, by which an external PC can send infrared signals of the IrRC to the appliances. The infrared signals are specific to devices and vendors. Also, executing a feature of an appliance requires the user a vendor-specific operation of the IrRC. So, it is inconvenient for external HNS applications to use the Ir-APIs directly.

IEEE International Conference on Web Services (ICWS'06)
0-7695-2669-1/06 $20.00 © 2006 IEEE

We then aggregate multiple Ir-API calls within self-contained *services*, so that each of the service achieves a logical feature independent of the vendor(or device)-specific issues. Finally, the services are deployed in the HNS as *Web services* [15]. Thus, every legacy appliance becomes an distributed component with an open interface, which can be used by various kind of HNS applications. The users can build their own integrated services and HNS applications with the legacy appliances.

To demonstrate the effectiveness, we have implemented an actual HNS and several integrated services. As a result, it was shown that the proposed framework is well appricable to multi-vendor legacy appliances, and that practical integrated services can be created as relatively small client applications.

2 Preliminaries

2.1 Home network system (HNS)

A HNS consists of one or more networked appliances connected to LAN at home. Each networked appliance has a set of *control APIs*, so that the user or software agents can control the appliance via the network. To process the API calls, each appliance generally has embedded devices including a processor and a storage.

Communications among the appliances are performed based on the underlying network protocol. Currently, many standard protocols are being standardized for the networked appliances. Major protocols include DLNA [1] for digital audio/video appliances, and ECHONET [2] for electric white goods (e.g., refrigerator, air-conditioners and laundry machines). However, these standard protocols mainly prescribe a set of *network-layer* agreements of the appliance, such as address setup and message formats. The *programmable interoperability* at the application layer (or higher) is beyond the scope of the protocols.

To minimize the interoperability issue, most of the current HNS (e.g., [4][11][14]) are comprised of the single-vendor appliances. Applications on the HNS are also limited to the proprietary ones provided by the same vendor. Types of appliances that can be integrated are also limited. The next challenge for the industries is to establish standards at the application layer, which allows any combination of multi-vendor appliances over different protocols.

2.2 Service-oriented framework for HNS [5]

Service-oriented architecture (SOA) is a system architecture that facilitates flexible integration of distributed heterogeneous systems [10] [13]. In SOA, primary features of each system are aggregated as a set of *services*. More sophisticated systems are basically constructed by *integrating* these existing services. SOA has been extensively studied and adopted in the domain of *enterprise systems*, since

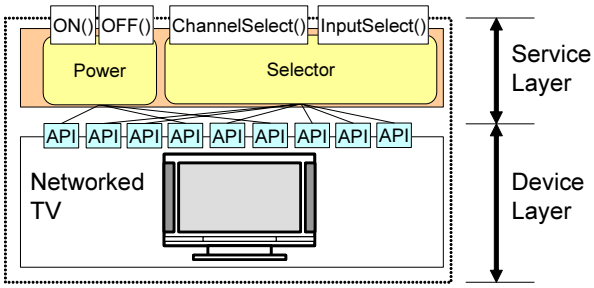

Figure 1. Architecture of networked appliance based on SOA

SOA-based systems are quite resilient for changes and integration of business processes. *Web services* [15] are known as a primary means to implement SOA-based systems.

To improve the interoperability issues discussed in Section 2.1, in our previous work [5], we have presented a *service-oriented framework for HNS*, which applies the concept of SOA to the networked appliances. Figure 1 depicts an example of a networked TV, illustrating the key idea of the framework. In this framework, each networked appliance is designed based on a two-layered architecture: a *device layer* and a *service layer*. The device layer represents the hardware and the control APIs (including the middleware) of the networked appliance. In the figure, the APIs denote the control APIs of the networked TV. They can be invoked by external entities in accordance with the HNS protocol conformed by the TV. On the other hand, the service layer aggregates the control APIs according to the logical features of the TV. Then, the layer exports the features to the network as the *self-contained* services with *open interface* (i.e., methods). The service layer is implemented as an application, and is supposed to be installed in the storage of the networked appliance.

As seen in Figure 1, methods like `ON()`, `OFF()` and `ChannelSelect()` are open interfaces for accessing basic features for any kinds of TV (i.e., TV services). Invocation of these methods does not require any knowledge specific to the underlying implementation of the device layer. Unless the interface definition is changed, any modification in the service and device layers does not give impact to the service users. Also, the service users can easily develop their own integrated services, by combining the method invocations of different appliances. As a result, we are able to achieve a HNS that copes with both system evolution and appliance interoperability.

Note however that the framework presented in [5] assumes the networked appliances only. The legacy appliances without the processor or the storage are beyond the scope. Also, the framework is not yet fully evaluated or implemented with the actual networked appliances.

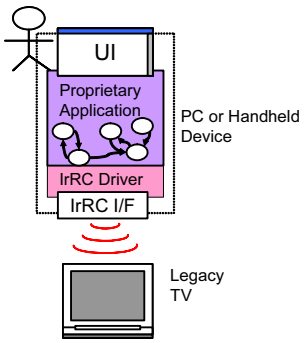

Figure 2. A soft-controller for a legacy TV

2.3 Software controller for legacy appliances

There have been several software applications with which the user can control legacy appliances from a PC or a handheld device such as handy phones [6][9]. We here call such applications *soft-controllers*.

Figure 2 shows a typical example of the soft-controllers, which is for a legacy TV with the conventional *infrared remote controller for home appliances* (denoted by *IrRC*). The user first selects and executes a control command through the user interface (UI). Then, the application sends the corresponding infrared signals to the appliance through the driver (IrRC driver) and the interface (IrRC I/F) [1].

The soft-controllers basically assume a use case that a human user controls a single appliance at a time. They are not supposed to be invoked by other applications, nor to be orchestrated by other appliances via the network. Also, since the appliance and the application are tightly coupled, the same controller cannot be used directly for other appliances. Therefore, it is difficult to apply the soft-controllers directly for the purpose of adaptation of legacy appliances to the HNS.

3 Adapting legacy appliances to HNS

3.1 Requirements

Our goal is to adapt the legacy appliances with the IrRC to the HNS. More specifically, we try to propose a framework satisfying the following requirements.

Requirement R1: The framework must achieve easy creation of HNS integrated services with arbitrary combinations of the legacy appliances.

Requirement R2: The framework must be implemented using generic PCs and IrRC devices without special hardware.

[1]The IrDA, which is often used for exchanging data among PCs and handheld devices, is completely different from the IrRC, although both protocols use the infrared for the physical layer.

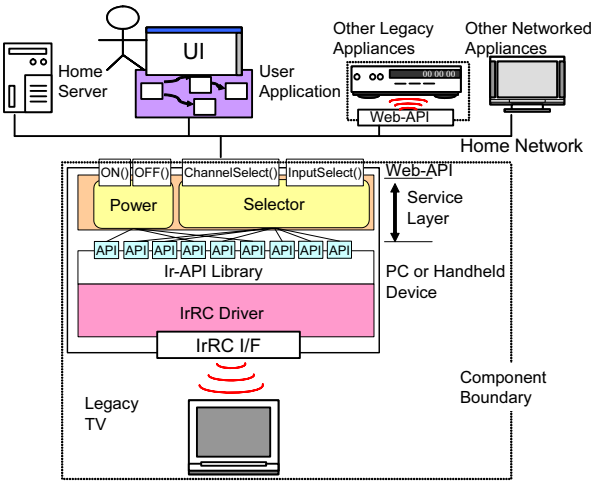

Figure 3. Proposed architecture for adapting legacy appliances

Requirement R3: The framework must be applicable to a wide range of types and vendors of the appliances.

The current HNS products in Section 2.1 do not assume integration of the legacy appliances with the HNS. Also, they have yet problems in achieving Requirements R1 and R3. Our previous framework in Section 2.2 takes Requirements R1 and R3 into account, but it cannot satisfy Requirement R2. The soft-controllers in Section 2.3 are for legacy appliances, but cannot satisfy Requirement R1.

Thus, the existing methods cannot be used directly to achieve our goal.

3.2 Proposed architecture

To satisfy Requirements R1 through R3, we here propose a new architecture depicted in Figure 3, which replaces the device layer of our former architecture (see Figure 1) with the IrRC devices and the legacy appliance.

First, we prepare a PC (or handheld device) and an IrRC device that can be connected to the PC. On the PC, we implement a set of APIs called *Ir-APIs* with which applications can send any infrared signals to the appliance. Next, for a given legacy appliance, we develop a set of services of the appliance, each of which encapsulates several Ir-API calls to achieve a self-contained and vendor-independent feature. Finally, we export these services as Web services and deploy them in the HNS.

The integrated services and any user applications are implemented as client applications that invoke the exported methods of the Web services. We call the methods *Web-API*. For this, the implementation of IrRC, which often varies among appliance types and/or vendors, is encapsulated within the service layer. The service users can execute

various features of the appliance without device-specific knowledge. The same framework is applied to other legacy appiliances. Then, the integrated service can be created by assembling Web-APIs provided multiple appliances according to a desired control flow (i.e., workflow).

3.3 APIs for infrared remote controller (Ir-APIs)

In general, a legacy appliance with an IrRC is operated as follows. When a user presses a button of the IrRC, an infrared signal corresponding to the button is issued from the controller. Upon receiving the signal, the appliance executes the corresponding feature (or operation). Thus, the communication mechanism is quite simple. However, the correspondence between the infrared signal and the feature varies among vendors and types of the appliance. Therefore, it is convenient to have all-purpose wrappers for the IrRC, with which the applications in the upper layer can reasonably switch and manage such vendor(and device)-specific signals.

For this purpose, we implement Ir-APIs on the top of the Ir-RC driver. The Ir-APIs provide a set of generic interfaces with which the applications can send any types of infrared signals to *arbitrary* legacy appliances. The Ir-APIs are relatively low-level but generic APIs, and they should not be bound with a specific appliance. Typical Ir-APIs include; initialize IrRC, set signal type, send signal, start sending burst signal, stop sending burst signal, sleep.

3.4 Service layer

Each feature provided by the Ir-APIs is so fine-grained that it does not necessarily corresponds to a single logical feature of the target appliance. Also, each IrRC operation heavily depends on the vendor and type of the appliance. It is not a good idea to expose every Ir-API directly to the HNS, since the user has to take care of the device-specific specification of IrRC of the target appliance.

The service layer aggregates, for every logical feature of the appliance, several Ir-API calls within a *service method*. What most important here is that every service method must be *self-contained*. Specifically, we recommend that every service method m should be implemented as follows:

(a) m is always executable by itself, independent of the context of other services or appliances.

(b) m achieves by itself a consistent logical feature of the appliance.

For instance, let us consider a TV manufactured by a vendor A, denoted by TV_A. Suppose that TV_A has the following restriction on its implementation: "Once turned on, it does not accept any infrared signals during 2 seconds until the hardware becomes fully operational". Then, the service method `ON()` for TV_A should be implemented as in the Java-like pseudo code in Figure 4. The first four statements are for achieving a logical feature "turn on TV_A", satisfying the above **(b)**. The last statement `sleep(2);` is for the subsequent method invocations to achieve the above **(a)**. The sleep statement consumes the 2-second wait, so that any methods executed after `ON()` should not be influenced by the device-specific restriction of TV_A.

```
public void ON(void) {
 IrControler con = new IrController();  /*Controller and */
 IrSignal sig = new IrSignal();/*signal objects of Ir-API*/
 sig.setSignalType(ON, TV_A);  /*set signal ON for TV_A*/
 con.sendSignal(sig);          /*send the signal*/

 sleep(2);                      /*Sleep during 2 seconds*/
}
```

Figure 4. Service method ON() for TV_A

```
public void setVolume(unsigned int x) {
 IrControler con = new IrController();  /*Controller and */
 IrSignal sig = new IrSignal();/*signal objects of Ir-API*/

 if (Power==OFF) On(); /*Turn on when TV_A is off*/

 sig.setSignalType(VOLDOWN, TV_A); /*Volume down signal*/
 for (repeat_enough_times) {  /*Minimize the sound level*/
   con.sendSignal(sig);
 }

 sig.setSignalType(VOLUP, TV_A); /*Volume up signal*/
 for (;x>0;x--) {     /*Issue volume up x times*/
   con.sendSignal(sig);
 }
}
```

Figure 5. Service method setVolume() for TV_A

When a feature requires a sequence of multiple Ir-APIs, these APIs should be encapsulated within a service method. The pseudo code in Figure 5 is to set the sound volume of TV_A to a given level x. In general, the volume level is adjusted by a human user with a relative scale considering the current volume level. However, here the application needs to do the task. Hence, the method `setVolume()` first minimizes the sound volume by repeatedly sending the signal of volume-down, and then sends the signal of volume-up for x times. Moreover, the sound adjustment feature depends on the power feature, since it works only when TV_A is ON. Therefore, `setVolume()` contains the invocation of `ON()` in case the power is off. Thus, the service method becomes self-contained.

3.5 Supplementary service: getting status

Some HNS applications may get the current status of an appliance, and then perform an appropriate action based on the status. A typical example is an energy-saving service, which stops a DVD player when a TV is turned off. However, the communication among a user and a legacy appliance is basically *one-way* from the IrRC to the appliance. Hence, it is impossible for the external application to obtain the current status directly from the legacy appliance.

```
DVD.ON();                 /*Turn on DVD*/
TV.On();                  /*Turn on TV*/
TV.setInput(DVD);         /*Set input mode for DVD*/
Curtain.On();             /*Turn on curtain*/
Curtain.Close();          /*Close curtain*/
Light.ON();               /*Turn on light*/
Light.setBrightness(1);   /*Minimize brightness*/
Speaker.On();             /*Turn on speaker*/
Speaker.setInput(DVD);    /*Set input mode for DVD*/
Speaker.setChannel(5.1); /*Set channel to 5.1ch*/
Speaker.setVolume(60);    /*Set volume level to 60*/
DVD.play();               /*Play DVD*/
```

Figure 6. Integrated service (DVD theater)

To cope with the problem, we implement, in the service layer, a supplementary feature that store the *current state* of the appliance according to the history of service execution. Specifically, for every appliance, we prepare a database, called *state DB*, for storing the current values of primary properties the appliance. When a service method is executed, the values of the corresponding properties are updated in the state DB. We then deploy a service method, `getStatus()`, to the HNS, which returns the current state (i.e., the tuple of the property values) upon the request from the external applications.

3.6 Exporting service methods as Web-APIs

For every service method implemented in the service layer, we export the method to the HNS as a Web-API. In this paper, we adopt the Web services [15], which is a standard SOA framework, for the service exportation. The interface definition of each service method is strictly typed by an XML-based language, called WSDL. An external application first interprets the interface definition, then invokes a Web-API via network with appropriate parameters.

3.7 HNS integrated services

Once features of every legacy appliance are exported as Web-APIs, the legacy appliances can be used as *distributed components*. By combining these components, the user can assemble various integrated services. Specifically, an integrated service can be implemented as a client application consisting of invocations of the Web-APIs and a control flow among the APIs. Note that the user can invoke Web-APIs with an arbitrary control flow, since every Web-API should be self-contained. For instance, let us consider the DVD theater service mentioned in Section 1. Figure 6 represents a sequence of Web-API invocations that can implement of the service.

The client application is installed on a remote PC or a handheld device owned by the user. It is also possible to install the integrated service on the *home server*, which manages all the appliances in the HNS and usually plays a role of a *residential gateway* to external networks. Then, the integrated service can be executed even from outside home. Also, the integrated service can be deployed as a new Web service to be reused for more sophisticated services.

4 Implementation

Based on the proposed framework, we have implemented an actual HNS with the legacy appliances.

4.1 Legacy appliances used

The following legacy appliances have been used in our HNS implementation. Note that our system is composed of *multi-vendor* appliances.

Plasma display	NEC PX-50XM2
DVD/HDD recorder	Toshiba RF-XS46
Wireless LCD TV	Sony KLV-17WS1
Ceiling light	Panasonic HHFZ5310
Curtains with actuator	NAVIO Powertrack
Air cleaner	Hitachi EP-V12
Air circulator	MORITA MCF-257NR
Power plug with IrRC	HORIBA IS-100
Climate monitor (sensor)	IT Watchdogs WxGoos-1

4.2 HNS implementation

Figure 7 depicts the overview of the developed system. In this implementation, we installed services for all appliances within a single PC. We should note that this is just for convenience of the experiment. The services can be distributed among multiple PCs if necessary (see Section 5.2 for more details). Technologies used for the system components are summarized as follows.

PC	Celeron M360J, 512MB, 80GB, WinXP Pro
IrRC I/F	Sugiyama Electron – Crossam2+USB
IrRC Driver	Serial COM library for Crossam2+USB
Ir-API	Java Native Interface (JNI) Wrapper
Service Layer	J2SE 5.0
Web-API	Apache AXIS 1.3

The Crossam2+USB (Crossam, for short) adopted for IrRC I/F is a programmable infrared remote controller, which can be connected to a PC. Crossam can memory infrared signals of various legacy appliances, and can dispatch a signal in each button. The customization of Crossam is quite simple, and thus we can easily correspond to addition or replacement of legacy appliances. Also, Crossam has a bundled serial port library written in C++ with which the external program can send Crossam low-level commands such as press a button, start and stop signal. Hence, we have used the bundle for IrRC Driver.

Next, we implemented Ir-APIs that can be invoked from Java programs by wrapping the Crossam library with Java

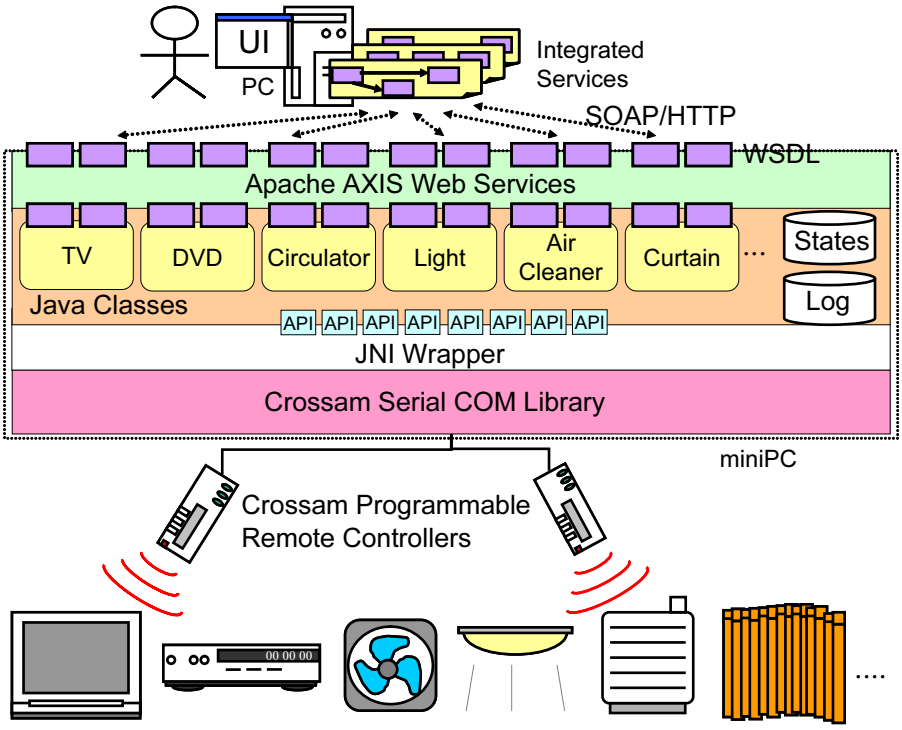

Figure 7. Overview of the HNS implementation

Native Interface (JNI). The usage of JNI is due to consideration of the portability for Web services. The JNI wrapper comprised 130 lines of Java code.

The service layer was implemented in Java. For each appliance we constructed a Java class, where features of the appliance correspond to service methods. We constructed total 12 classes (4 for the climate monitor and 8 for other appliances) with 132 methods, comprising 1196 lines of code. Finally, the Java classes were deployed as Web services using Apache AXIS.

4.3 HNS Integrated services

We have implemented the following integrated services as client programs with Perl's `SOAP::Lite` module. Although the integrated services seem to be sophisticated, they could be implemented in quite small programs, each of which comprises about 20-30 lines of code.

DVD theater: Integrating the TV, the DVD player, the speaker, the lights and the curtains, the service automatically sets up the living room in a theater configuration. Upon a user's request, the TV is turned on with the DVD input, the curtains are closed, brightness of the lights are minimized, the speakers are configured for 5.1ch mode, and finally the DVD player plays the contents.

Air cleaning: Integrating the smoke sensor of the climate monitor, the air cleaner, the circulator, the service cleans dirty air in the room, automatically and efficiently.

Wakeup support: Integrating the speaker system, the lights, and the curtains, the service assists the user to wake up smoothly. Before 10 minutes of the given wakeup time, the speaker system and the lights are turned on with minimal sound volume and brightness. Then, the volume and the brightness are gradually increased, and set to the optimal at the wakeup time. Finally, the curtains are opened.

Auto illumination: Integrating the climate monitor, the curtains, and the lights, the service always keeps the optimal lighting in the room regardless of time and weather.

4.4 Performance evaluation

We have measured *response time* of each integrated service developed in Section 4.3, in order to clarify the overhead posed by the proposed framework. There are two kinds of response time: *device response time* (DRT) and *service response time* (SRT). The DRT is the time physically taken for the legacy appliances to execute features required in the service. For instance, turning on our DVD/HDD recorder required DRT about 30 seconds for initializing the system. On the other hand, the SRT is the time purely devoted for processing the service layer with Web

Table 1. Response time of integrated services

	DVD theater	Air cleaning	Wakeup	Illumination
SRT (msec)	3,563	2,613	4,905	676
DRT (msec)	39,350	3,000	19,600	1,250
Total (msec)	42,913	5,613	24,505	1,926
# of Web-APIs executed	6	4	16	3

services, which is interesting for us here.

Table 1 shows the result. Each row represents the corresponding response time in milliseconds [2]. The last row represents the number of Web-APIs executed in each integrated service.

It can be seen in the table that the total response time varies from service to service. This is because the different services use different sets of legacy appliances. However, SRT has a strong correlation with the number of Web-APIs executed, regardless of the features executed by the Web-APIs. This is because the time taken for the middleware to process Web service messages became a dominant factor for SRT. SRT for each Web-API invocation varied from 0.22 seconds to 0.65 seconds. Hence, it can be said that the proposed framework suffers from quite a little overhead.

We consider that the overhead observed was sufficiently tolerable for the above four services. However, we should count the overhead carefully when developing services that require time-sensitive or real-time transactions.

5 Discussion

5.1 Qualitative evaluation

The proposed framework surely satisfies Requirements R1 through R3 in Section 3.1. R2 is satisfied as seen in Section 4. The service layer encapsulates well the vendor or device specific issues. The proposed framework applies to a variety of legacy appliances, as long as the IrRC device on the PC is compatible to the appliance. Thus, R3 is satisfied.

The integrated services can be implemented just by combining Web-APIs deployed with Web services. As the lines of code of our implementation indicate (see Section 4.3), the user can create their own integrated services without much effort. Thus, R1 is achieved. Although we implemented the services as application programs, we believe that using the Web service integration framework, such as BPEL4WS [16], would enable more efficient service creation.

The service layer of the proposed framework plays an important role, which achieves a *loose-coupling* between the HNS applications (as service users) and the appliances (as service providers). As long as the interface definition is not changed, one can change, replace or update the appliances without influencing the service users. Thus, the HNS based on the proposed framework is quite resilient for the *system evolution*.

Another merit of the proposed framework is that one can integrate legacy appliances with the existing Web services available on the Internet. This is a great benefit of choosing Web services. The followings are interesting service examples using news and stock services in the Internet:

News flash: As soon as an important news arrives, turn on the TV with the selected channel.

Stock alarm: When a stock price rises, chimes in the house ring. Development of more sophisticated services are left to future work.

5.2 Limitations

A technical limitation of the proposed framework is in the reliability of the physical channel between IrRC/IF and the appliance. As discussed in Section 3.5, the communication from the PC to a legacy appliance is basically one-way. Hence, it is not easy to confirm whether or not an infrared signal is successfully received by the appliance. If the signal is lost, inconsistency between the state DB and the actual status of the appliance occurs, which may lead the integrated service to malfunction. Therefore, we need to guarantee the reliability of the physical channel at all cost. Fortunately in our framework, the service layer can be distributed within multiple PCs. Hence, in an extreme case, we can assign a PC for *every* appliance so that the IrRC is close enough to the appliance. A smarter approach is to assign a PC for each group of *neighbor* appliances. The user has to choose a reasonable layout considering reliability requirement, cost for PCs, a floor plan and objects in the room.

We are optimistic for the overhead in the service layer. The overhead is basically due to performance of the current WSDL implementations in mashalling XML data to and from messages. This problem will likely be alleviated by future WSDL implementations as the technology matures.

User authentication and security management are beyond the scope of this paper, but are important issues in practical usage of the HNS. We conduct these problems in our future work.

[2] The time taken for each Perl client to prepare SOAP stubs from the corresponding WSDLs had been excluded from the data.

5.3 Related work

Loke [10] firstly modeled networked appliances as Web services, and proposed a workflow engine, called *Decoflow*, which prescribes integrated services using BPEL4WS. However, the method regards each networked appliance just as a *black box* with an open interface. Also, it does not mention the legacy appliances. The proposed framework can complement the method, which would allow efficient creation and management of the integrated services.

SOA with Web services is expected as a powerful means to achieve *legacy migration*. For instance, Lewis et al. presented a framework called SMART [8], which assists organizations in analyzing legacy capabilities for use as services in SOA. Zhang [17] proposed a method based on a code analysis, which facilitates legacy code extraction for Web service construction. However, most existing techniques including the above are addressing the legacy migration of *enterprise* systems. We believe that our original contribution was to show a concrete framework for legacy migration in the new domain, i.e., home network system, exploiting the essence of SOA with Web services.

Another interesting issue is the *feature interaction problem* [7], which is the functional conflicts among multiple HNS services on the appliances or environmental factors. In [12], we formalized the feature interaction problem and presented a conflict detection method in the context of the HNS. We are currently implementing the method on the developed system.

6 Conclusion

This paper presented a framework that adapts the legacy home appliances to the emerging home network system. The proposed framework extensively adopted the concept of SOA. Features of the legacy appliances are exposed as self-contained Web-APIs with Web services. We also implemented the actual HNS with multi-vendor legacy appliances, as well as several integrated services.

Our future work includes the security issues and the feature interaction management as discussed in Section 5.2. We also plan to investigate a method that supports *non-expert users* in creating integrated services easily.

References

[1] Digital Living Network Alliance, http://www.dlna.org

[2] ECHONET Consortium, http://www.echonet.gr.jp/

[3] D. Geer, "Nanotechnology: The Growing Impact of Shrinking Computers", *IEEE Pervasive Computing*, vol. 5, no. 1, pp. 7-11, Jan. 2006.

[4] Hitachi Home & Life Solutions inc., "HORASO Network Service", http://ns.horaso.com/index.html

[5] H. Igaki, M. Nakamura and K. Matsumoto, "A Service-Oriented Framework for Networked Appliances to Achieve Appliance Interoperability and Evolution in Home Network System" (short paper), *Proc. of International Workshop on Principles of Software Evolution (IWPSE 2005)*, pp. 61-64, Sep. 2005.

[6] "AVT Series", http://d-purasu.hp.infoseek.co.jp/

[7] M. Kolberg, E. H. Magill, and M. Wilson, "Compatibility Issues between Services Supporting Networked Appliances", *IEEE Communications Magazine*, vol. 41, no. 11, pp. 136-147, Nov 2003.

[8] G. Lewis, E. Morris, L. O'Brien, D. Smith, and L. Wrage "SMART: The Service-Oriented Migration and Reuse Technique", *Technical Note CMU/SEI-2005-TN-029*, Software Engineering Institute, Sep. 2005.

[9] NANO Media Inc., "App-rimo-con", http://www.nanomedia.jp/english/service/s02.html

[10] S. W. Loke, "Service-Oriented Device Echology Workflows", *Proc. of 1st Int'l Conf. on Service-Oriented Computing (ICSOC2003)*, LNCS2910, pp.559-574, Dec. 2003.

[11] Matsushita Electric Industrial Co., Ltd., "Kurashi Net", http://national.jp/appliance/product/kurashi-net/

[12] M. Nakamura, H. Igaki, K. Matsumoto, "Feature Interactions in Integrated Services of Networked Home Appliances -An Object-Oriented Approach-," *Proc. of Int'l. Conf. on Feature Interactions in Telecommunication Networks and Distributed Systems (ICFI'05)*, pp. 236-251, Jun. 2005.

[13] M. P. Papazoglou, D. Georgakopoulos, "Service-Oriented Computing", *Communications of the ACM*, Vol. 46, No.10, pp.25-28, 2003.

[14] TOSHIBA, "Toshiba home network – Feminity", http://www3.toshiba.co.jp/feminity/feminity_eng/index.html

[15] W3C, "Web Service Activity", http://www.w3.org/2002/ws/

[16] S. Weerawarana, and F. Curbera, "Business process with BPEL4WS: Understanding BPEL4WS, Part1", http://www-106.ibm.com/developerworks/webservices/library/ws-bpelcol1/

[17] Z. Zhang, and H. Yang, "Incubating Services in Legacy Systems for Architectural Migration", *Proc. of 11th Asia-Pacific Software Engineering Conference (APSEC'04)*, pp. 196-203, Dec. 2004.