DYNAMIC SERVICE COMPOSITION FOR VIRTUAL UPNP DEVICE CREATION IN HOME NETWORK ENVIRONMENT

Sheng-Tzong Cheng, Chun-Yen Wang, Mingzoo Wu,Wan-Ting Ho
Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan

Chia-Mei Chen
Department of Information Management, National Sun Yat-Sen University, Tainan, Taiwan

Keywords: UPnP, data type ontology, service composition, virtual application probing, virtual device creation.

Abstract: Exploiting UPnP techniques, home users can easily control intelligent devices through control points. However, UPnP devices lack a composition mechanism to complete a novel application or value-added service. This paper proposes a dynamic service composition system which coordinates the primitive UPnP services at home to create a virtual device. We define data type ontology for UPnP devices to describe their service interfaces. The interface matching mechanism is employed to construct a service graph that describes which services can be composed together. Finally, the proposed system travels on the service graph, and probes a suitable execution path to generate a new device.

1 INTRODUCTION

With the proliferation of home networked devices, all sorts of devices could be discovered and controlled by UPnP architecture. The goal of UPnP architecture is to define the communication protocols between control point (CP) and devices. UPnP uses common protocols which are independent of the underlying physical media and transports, and ensure every device vendors could follow. Many industry companies and research initiatives such as UPnP, OSGi, DLNA, and HAVi have tried to understand the communication protocol between CP and devices. However, up to now, they are very little to get in touch with composing the primitive services to create value-added services.

Each UPnP service description lists the service type, name, URLs for a service description, control, and eventing. The description is recorded in XML-based syntax. UPnP is an open networking architecture that uses Web technologies to enable seamless proximity networking in addition to control and data transfer among networked devices. UPnP is a Web-based communication protocols between CP and devices.

As Web services become more and more prevalent, many Web service composition (SC) technologies are proposed and developed. BPEL4WS (Curbera, 2001) supports process-oriented SC, which represents a specific process composition flow. CoSMoS (Fujii, 2004) can integrate the services to construct a semantic graph. Given a user request, CoSMoS checks the semantics and generate an execution path. In eFlow (Casati, 2000), a composite service is designed by a template which defines an order of execution of the services. User can choose an application template from the repository or by creating a template by himself. The request application is composed through selecting the services specified in the template and combining them according to the structure described in the template.

In this paper, we aim at designing and implementing a dynamic SC (DSC) system and create a virtual device in home network environment. We propose data type ontology to define interfaces for services. And we show how to use semantic descriptions to aid in the DSC system. Also, we present a method called Virtual Application Probing (VAP) which allows home virtual applications can be dynamically and semantically composed from the individual services of home networked devices. We

Cheng S., Wang C., Wu M., Ho W. and Chen C. (2007).
DYNAMIC SERVICE COMPOSITION FOR VIRTUAL UPNP DEVICE CREATION IN HOME NETWORK ENVIRONMENT.
In *Proceedings of the Third International Conference on Web Information Systems and Technologies - Internet Technology*, pages 349-352
DOI: 10.5220/0001269603490352

design and implement the DSC system using existing technologies such as UPnP, ontology, and XML.

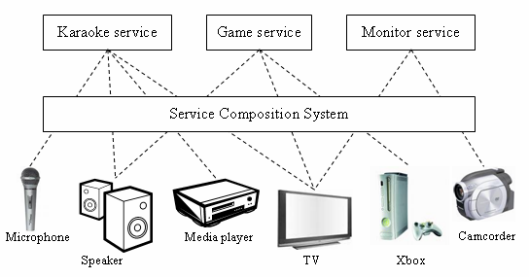

Figure 1: The concept of dynamic service composition.

This paper is organized as follows. Sec. 2 describes the DSC system. Sec. 3 describes the implementation techniques. Sec. 4 presents the demonstration of the proposed system. Conclusion remarks are drawn in Sec. 5.

2 SYSTEM ARCHITECTURE

Fig. 1 illustrates the concept of DSC. There are many independent devices, such as Microphone, Speaker, Media player, TV, Xbox, Camcorder, etc. Through DSC system, these useful services could cooperate with each other to create a virtual device or a novel application.

2.1 Data Type Ontology Classification

We extend the service description and add semantics information to define a service interface. The I/O variables are classified according to their data type and further classified according to semantics.

UPnP CP retrieves device and service information with semantics to provide accessible services. If the data type ontology share and publish the same underlying ontology of the used terms, then CP can extract and aggregate the data type and semantic information to do service matching. The benefit of using data type ontology is that it is easy for developers to design service interfaces and is easy for users to understand. After defining the data type ontology, we use the data type and semantic information to describe the service interface. With interface matching method, we could know whether the service's output can be fed into the next service's input. Fig. 2 shows the ontology diagram of the variables for communication interface of home networked devices.

2.2 Service Interface Matching

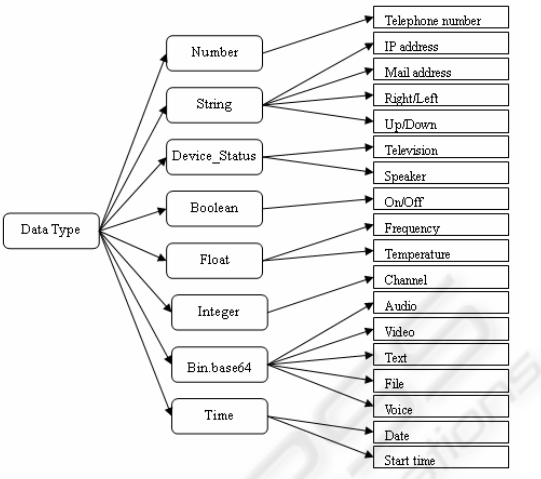

Figure 2: Data type ontology for home-domain devices.

A service interface specifies methods that can be performed on the service. Service's interfaces are public for an external use. A service can hold two sorts of interfaces: input and output interfaces. Traditionally, UPnP service description only has data type and I/O information to describe a service interface, but it is not enough to perform DSC.

With the support of data type ontology, semantics of a service can be freely defined, thus providing high extensibility. We use data type and semantic information to describe service interfaces. With interface matching method, we could know whether the service's output can be fed into the next service's input.

2.3 Service Graph Construction

Service graph (SG) is an intuitive way to represent composition concepts. Two types of nodes are defined in SG: DataSemanticNode (DSN) and ServiceInfoNode (SIN). The links between these nodes represents their associations. The SG clearly represents the composition information. The benefit of using SG for DSC is its understandability for users. Also, the SG is easy to maintain and handle for generator and designer.

- **DSN:** Each DSN represents a pair of data type and semantic defined by the data type ontology to represents the service's interface. We denote (Data Type, Semantic) as a DSN.
- **SIN:** We take down the device and service information in the SIN, which has two kinds of information, the device name and the service name. We denote SIN as (Device name, Service name).

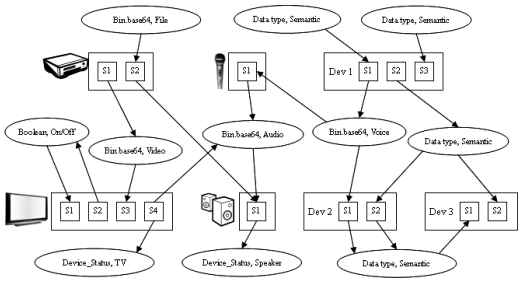

Figure 3: Directed service graph.

To construct a directed SG, we create links between the DSNs and SINs. When a device is discovered by CP, our system checks the service interface's data type and semantics of the added device. If the DSN's data type and semantics are the same with the service input interface's data type and semantics, then a link is directed from DSN to SIN. Otherwise, if the DSN's data type and semantic are the same with the service output interface's data type and semantics, then a link is directed from SIN to DSN. Each time when a device is discovered by CP, our system would make links between DSN and SIN automatically. Fig. 3 represents the directed SG.

2.4 Execution Path Exploration

We use Virtual Application Probing (VAP) to find a composite service path in the directed SG. To implement this scheme, we place each visited node into a linked list and record its preceding node with index information. Then we use VAP to find a shortest execution path (EP) in SG. **Step 0:** Check virtual device description. CP reads the virtual device's description to know the virtual service's input and output interfaces' data types and semantics. **Step 1:** Put the DSN which represents the virtual service's input interface into linked list. **Step 2:** Put SINs whose input interface is the DSN from step 1 into linked list. Record their indexes as the number of the preceding nodes in the linked list. **Step 3:** Put the DSN which represents the output interface of the SIN from step 2 into list. And record the index as the number of the predecessor nodes in the linked list. **Step 4:** If we find DSN of the virtual service's output interface, stop searching and export the EP according to their indexes. Otherwise, repeat Step 2 and 3 until there are no nodes left in linked list.

2.5 Virtual Service Provision

Once all EPs of virtual device are explored, the virtual device with the EPs will be created and CP would invoke the services which was required in the EP sequentially. Here we describe two types of flows: control flow and data flow. Control flow is used to trigger the required service in EP. If we invoke the virtual service, the CP would invoke the primitive services automatically with the control flow. And data is exchanged between services without going through the CP.

3 SYSTEM IMPLEMENTATION

Our device model is refined from CyberLink for Java (CyberLink 2004), which is a development package for UPnP developers. We append some classes and methods to implement our DSC system.

- **ControlPoint Class:** Method DeviceAdded() checks whether a service can be composed with other service by interface matching. If the added service's interface exactly matches another service's interface then we connect the two service nodes in SG. If a device is leaving the UPnP network, method DeviceRemoved() removes the links between the service nodes.
- **SemanticType Class:** This class is used to describe the data structures of SG. DSN represent the interfaces of a service. And SIN records the service information including device name and service name. With the aid of semantics, we use "interface matching" approach to find out what service's output can be fed to the next service's input. Then we use VAP to find a virtual service in the directed SG.
- **LinkedList Class:** The LinkedList class is used to find EP of virtual device. We implement a linked list to record the SG nodes in EP. Extra index information is used to record the preceding node's number in the linked list.
- **VirtualDevice Class:** In the beginning of VAP process, CP will check the virtual device's descriptions. If the virtual service can be composed from the primitive services, we create an instance of virtual device with the EP and user can invoke the virtual device from CP. Unlike the real device, the virtual service/device is not implemented beforehand. The virtual device can receive a composite EP found by VAP. The virtual service is a composite service composed from the primitive services. Once the virtual service is invoked, CP would sequentially invoke the primitive services involved in EP.

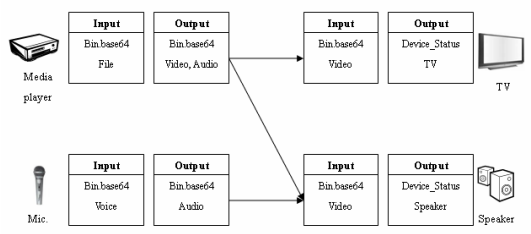

Figure 4: Creation of virtual Karaoke.

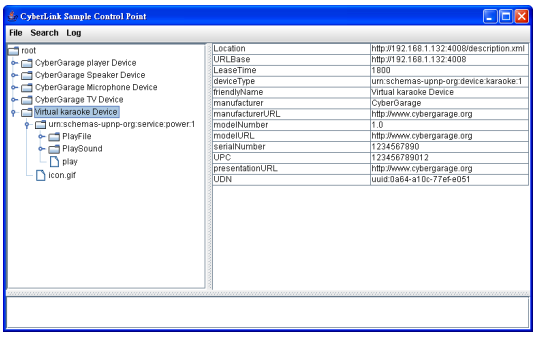

Figure 5: Virtual Karaoke.

4 DEMO SCENARIO

We implement a virtual Karaoke which is composed of TV, Player, Speaker and Microphone. In our system, VAP creates a virtual Karaoke from these four devices. Fig. 4 shows the creation of virtual Karaoke from the four devices using interface matching.

4.1 Semantic Service Description

We make description files of devices and services to create device. A service must be able to represent data type, I/O, and semantic information. We add semantic tags in XML-based service description. When a device is plugged into the network, the CP retrieves the device and service descriptions from the discovery message.

4.2 UPnP Devices Simulation

Using CyberLink for UPnP package (CyberLink 2004), we can implement devices and CP easily. The Media player has two services: SetPower and PlayFile. The input data type of PlayFile is Bin.base64 and its semantic is File. The output data type of PlayFile is Bin.base64 and its semantic is Audio and Video.

4.3 Virtual Karaoke Creation

Once CP finds the TV, Player, Speaker and Microphone devices, DSC system would create a virtual Karaoke which is composed from the four devices automatically. A composite path that found by the VAP is: (Bin.base64, File) → (Player, PlayFile) → (Bin.base64, Video) → (TV, Visual) → (Device_Status, TV) and (Bin.base64, Voice) → (Microphone, Mike) → (Bin.base64, Audio) → (Speaker, PlaySound) → (Device_Status, Speaker). Fig. 5 shows that users can view and invoke the virtual Karaoke through CP, which can invoke the required services on the EP.

5 CONCLUSIONS

In this paper, we present using semantic descriptions to aid DSC. We design the service interface with data type and semantic information. Each interface is specified by data type and semantic tags. With interface matching method, we could know which the service's output can be fed into the next service's input. We also present SG and VAP to find composite EPs of virtual device. SG is an intuitive way to represent composition concepts in an understandable way.

ACKNOWLEDGEMENTS

This work was supported in part by TWISC@NCKU, ITRI, and National Science Council under the Grants NSC 94-3114-P-006- 001-Y, B95-063B, and NSC 95-2218-E-006-011.

REFERENCES

Curbera, F., etc. 2001. Business Process Execution Language for Web Services.

Fujii, K., Suda, T., 2004. Dynamic Service Composition Using Semantic Information, In *International Conference on Service Oriented Computing*.

Casati, F., etc. 2000. Adaptive and dynamic service composition in eFlow, In *CAiSE*.

CyberLink for Java 2004, http://www.cybergarage.org/.