A Virtual Networked Appliance using Automatic Service Composition

Ch. Namman*, A. Mingkhwan**
*Department of Information Technology, Faculty of Information Technology,
**Department of Information Technology, Faculty of Industrial and Technology Management,
King Mongkut’s Institute of Technology North Bangkok, Thailand
Email: chnamman@gmail, anirach@ieee.com

Abstract – Nowadays, developments in information technology and networking are rapidly driving the enhancement of networked appliances in the home environment. Networked appliances can be represented by the functionality that they offer to the network in terms of services. The user can then request their services via the home network framework. Such representation makes it possible to provide new concomitant capability by creating a virtual appliance. However, due to the vast number of home networked appliances and their complicated services, such composition can prove challenging for the non-technical home user. In this paper, we propose our framework for providing automatic service composition capabilities in networked appliances. It uses both centralized and decentralized approaches. The framework is based on a knowledge transfer mechanism whereby each device would have its own composition knowledge and is able to transfer to other devices in the network.

Keywords – automatic service composition, heterogeneous environment, networked appliance, service discovery

I. INTRODUCTION

Nowadays, the development of networked appliances in the home environment has become hugely supported by the great advances in information technology and broadband networks [1-3]. A networked appliance [4] is a dedicated function consumer equipment containing a networked processor (in addition to computer devices such as PC, PDA) which can be connected with other devices over a network. They describe the functionality they can offer to the network in terms of services which the user and other networked devices would become possible to access. The devices can therefore offer the possibility of providing new synergetic capability by creating a virtual appliance. For example, a virtual TV is created by the cooperation of the PC’s visual service, the Hi-Fi’s audio service and the set-top box’s receiver service. Due to the variety of networked appliances and their complex services, it has not proved easy for an ordinary home user to configure those services to provide a new virtual appliance. Our paper is addressing this issue by proposing a framework for automatic service composition that can easily and effectively be operated by a normal home user.

The Service Oriented Computing (SOC) [5] paradigm defines mechanisms to publish, find and bind service. Those features make it particularly applicable for a heterogeneous environment where interoperability is essential. Service composition is the functionality in SOC that can be defined as the process of discovering, integrating, and executing a set of related services in order to fulfill either user or device requirement. “Service” can be as either a functionality of hardware device or a software function, which accept a proper input and generate some output according to its functionality. Service composition enables the user to obtain resources and services from the environment to perform required tasks.

Many researches and projects in service composition focus on Web Service Composition [6-8]. Some researches use a centralized approach for service discovery and service integration. The centralized approach may not be efficient and suitable for some situations. For example, it may incur a large overhead or a bottleneck when network services access and use the system frequently. Also, a centralized server can suffer from problems such as high operational and maintenance costs, single point of failure, and scalability. Other researches addressed a de-centralized approach. With this approach each node can be both client and server where they can advertise their need or/and aid each other very directly. The downsides of the approach are the higher bandwidth used, increased security issues, and the difficulties of maintaining the network.

Our framework proposes a hybrid approach for automatic service composition that use both centralized and decentralized approaches. Any networked device has its own composition knowledge which can be transferred to other devices. We call this process Knowledge Transfers. The devices use a centralized approach by default to perform the composition process. However, when the central server is unavailable they have ability to use a decentralized approach.

The paper is organized as follows: Section 2 provides a survey or the related work; we illustrate a scenario of automatic service composition in our framework in Section 3; then present a framework for automatic service composition in Section 4. Section 5 presents the current implementation; and
II. RELATED WORKS

There are several previous works related to automatic service composition in heterogeneous systems. Several projects have demonstrated techniques for Web Services. To work with the rich language, Rao et al [9] developed a system that translates a user’s request from DAML-S language [10] into Linear Logic (LL). An LL theorem is used to find a sequence of available services which are input/output can be cooperative to fulfill the user’s request. The result of composition is described in DAML-S too. Evren et al [11] introduced a semi-automatic solution for Web service composition. All of web services are described in OWL-S language [12]. A user is given a list of available services to start the service composition process.

HP labs has developed a system called eFlow [13], for enacting, monitoring and specifying for composite e-service. It has two type node such as service node and decision node. Service nodes specify the desired service’s characteristics or composite service. A composite service is modeled by a graph to define a composition process schema. Each service is described by an XML structure. A Decision node is responsible for the additional functionality and service selection rules to control the execution flow. The e-Flow composition engine is used to execute the composite service when all the services in the graph have been discovered.

Canal [14] is a distributed service composition system based on JXTA peer-to-peer model and mobile agent approaches that can avoid a downside of the centralized approach. Their mobile agents can be considered as the brokers to link the services together and bring the result to the user. The user can entrust the work to the mobile agents and just wait for the results. The processes of this system are: discover services, send out a mobile agent to link them together, set up the inputs and outputs, return back to the client along a reverse path and initiate each service on the way. The composite service now starts to serve the client.

Pourreza and Graham [15] proposed a service composition model which involves third-party “service providers” (SPs) in the composition process thereby allowing the discovery of services without direction from the users as to what type of service is desired. In the initial prototype system, ontology based matching is done by the service providers and the resulting composite service is deployed as a workflow using available in-home protocols.

Chakraorty et al [16] developed a distributed broker-based protocol for service composition in pervasive/ad hoc environments. For each composite request, a Broker is elected from within a set of nodes. The elected broker is delegated the responsibility of composition (e.g. discovering, integrating and executing) from the request node. The elected Broker uses a distributed service discovery architecture to discover services. It then integrates and executes services needed for a composite request.

III. SCENARIO

John is sitting on the living room. While he waits for his favorite television program, he would like to watch a video clip from his PDA phone. When the PDA phone first joins the network it would describe and register its services such as visual service, audio service, clip play service and others with the central server. After the central server registers the PDA phone services, the PDA phone asks the central server which are the best services for playing the video clip and request them. For Example, the TV’s visual service and the Hi-Fi’s audio service could be composed with the PDA’s movie player service. When the television program arrives, John switches to watch the TV. At that time, all service composition are revoked. When the television program comes to an end, John requests to watch the movie again. The PDA phone will use the last composition information which is stored on it to examine and request the services. It will then directly request the composition with both the TV’s visual service and the Hi-Fi’s audio service.

On a later day, John uses the PDA phone to play a new video clip at the bedroom. The PDA phone asks the central sever which are the best services for playing the video clip and request them. Unfortunately, the central sever is unavailable and the PDA can’t use the local composition information to examine and request its already registered services because its location has changed. Thus PDA phone will then use a discovery service around it. It will enquire the device that they known from the local composition information to find the service that is related to them. For example, the PDA can find services from a PC and inquire both the TV and Hi-Fi, what is the PC service that is related to the TV service and Hi-Fi services. With the reply, the PDA phone will be using both audio service and visual services from the PC to play the video clip.

IV. SYSTEM FRAMEWORK

To compose services on a heterogeneous environment, networked devices must describe a capability on term of semantic services which include as an input, an output, type of service, group of appliance, location, and other. Moreover, devices must be grouped to stationary devices, mobile devices or both. It is the central server responsibility to discover, register, plan and store information in the system. The planning of a service composition is generated by both manual and automatic manners. Some service composition planning is stored on the device.
In this section, we would like to propose an automatic service composition framework for networked appliances that uses both centralized and decentralized approaches. A networked appliance in the framework can require information for composition from the central server; on the other hand, when the central server is unavailable, it can use local information to perform the composition process. If its local information is inadequate, it can be interrogated from other devices in the environment. Figure 1 illustrates our system.

Our system has two service brokers to perform the service composition process. The Central Service Broker is the central server. It has a high priority where a requester device must be advised by it first. Some information of the service composition process that is related to requester device will be synchronizing to it.

Figure 2 shows the architecture of the Central Service Broker. It consists of eight processes, these are, Service Classify, Service Repository, Service Matching, Service Planning Template, Planning Knowledge, Service Planner, Service Evaluation and Service Execution.

First, the Service Discovery is used to discover services in the environment. The Service Classify check on some parameters, for example, is that service the latest? What is the service group? What type of service it is? After the classify process, the service will be stored into the Service Repository. That information is stored in a service ontology form. A Service Ontology includes Service Profile, Service Model and Service Grounding which is based on the OWL-S language.

Service Discovery \[\rightarrow\] Service Request
\[\rightarrow\] Service Repository
\[\rightarrow\] Service Matching
\[\rightarrow\] Service Planning Template
\[\rightarrow\] Planning Knowledge
\[\rightarrow\] Service Planner
\[\rightarrow\] Service Evaluation
\[\rightarrow\] Service Execution
\[\rightarrow\] Virtual Appliance

The Service Profile and Service Model are the abstract representations of a service and Service Grounding deals with the concrete level of specification. The Service Profile describes the capabilities and parameters of the service. A description of the service’s capabilities is in terms of Inputs, Outputs, Pre-conditions and Effects (IOPE). Generally, it answers the questions “what does the service do?” and “what does it require?” With this information the user or device will know if the service will meet its needs. The Service Model describes what happens, when the service is carried out. Formally, it describes how the service works by specifying the workflow and possible execution paths. This description contains a specification of a set of sub-processes that are coordinated by a set of control constructs. The Service Grounding explains how the service can be accessed and used. Grounding will specify the necessary information for the service invocation, such as the protocol, message formats, serialization and other details specific to the service.

When an appliance requests services by invoking a Service Request process (this is external process), Service Matching is the process of automatically deciding whether an offered service is able to fulfill a given service request. It also configures the service appropriately in case a match is required by the process. It then sends the service information to the

![System architecture for automatic service composition.](image1)

![Central Service Broker Architecture.](image2)
Service Planner. The Service Planner takes the service request as the input and generates planning information of service composition that constituent services listed in the Service Repository. It can also use planning information of service composition from both Planning Knowledge and Planning Template to consider the best planning information of the service composition. In general, a service request is expanded into constituent services until no more services need to be expanded. Services are expanded based on the inputs they provide to the composition.

The Service Planning Template is the repository of composition templates that the user can manage. Planning Knowledge is the repository of the ex-composition. With the Service Planning Template, a Service Planner can comfortably plan the service composition process. It is a generalized workflow where some of the steps are defined as abstract activities. There are various different ways to define an abstract service. The idea is not to fix each step of the workflow at design time and have the ability to discover and substitute services at run time. Such a template is more suitable in a dynamic environment where the available services and task requirements are constantly changing. The Service Evaluation process is used to evaluate related services to ensure that it matches the service specification. The Service Execution process is the process to execute composition of services to a virtual appliance. This process uses the related service to transport the required data to the destination. The data then flow in the chain.

The Peer Service Broker architecture is show in Figure 3. The Peer Service Broker starts when the Central Service Broker is unavailable. It has some processes similar to the Central Service Broker such as Service Classify, Service Matching, Planning Knowledge, Service Evaluation and Service Execution. It also has the Planning Exchange process to exchange information such as Service Planning to other peers. This is not part of the Central Service Broker. In the Peer service Broker the Planning Knowledge process stores a repository of the light weight composition information that is related and makes it accessible when the Central Service Broker is unavailable.

V. INITIAL WORK EXPERIMENT

In our initial work, we implemented a Networked Speaker’s audio service that provides an audio service to other devices such as a PC or a Laptop. A Networked Speaker was created using a TINI board [17], Java and XML-RPC. TINI is a TINI Internet Interface, developed by a company called Dallas Semiconductor. A TINI board is a “a microcontroller-based development platform that executes code for embedded web servers. The platform is a combination of the broad-based I/O, a full TCP/IP stack, and an extensible Java runtime environment that simplifies development of the network connected equipment”. The advantage of the TINI board is that it is a Java programmable runtime environment. The TINI board comes with a 512 KB flash memory, 1 Mb NV SRAM and a network support. The implementation of the Network Speaker is illustrated in Figure 4.

The Initial experimental environment consists of a PDA, a Networked Speaker and a Laptop. Both the PDA and the Laptop describe and provide a visual service, an audio service and an audio player service. The Network Speaker describes only an audio service. In this system, both the PDA and the Laptop are both the service requester and service provider.
They can describe their services such as a visual service, an audio service and an audio player service via the network; on the other hand they can request available services on the system. All the Network Speakers are only service providers. The Laptop’s audio player service can be composed with the Network Speakers’ audio service to play music. The application on our framework is developed using the Java Media Framework.

VI. CONCLUSIONS AND FUTURE WORK

Our proposed framework enables automated service compositions of networked appliances in heterogeneous environments. The framework is designed to provide a hybrid solution for automatic service composition which can avoid problems such as high operational and maintenance cost, single point of failure, and scalability. Our model has two service brokers represented by the Central Service Broker and the Peer Service Broker. The experimental implementation has shown the initial results of our work which includes service requesters and service providers.

Our future plans include implementing the Central Service Broker and the Peer Service Broker to the experimental environment. A knowledge transfer process would also be implemented to the device.

REFERENCES