

A Multi-domain Service Composition Platform

Seung-Ik Lee*, Jonghwa Yi*, Shin-Gak Kang*, and Young-Il Choi**

*Standards Research Center, ETRI, 218 Gajeong-ro, Yuseong-gu, Daejeon, 305-700, Korea

**Future Internet Research Division, ETRI, 218 Gajeong-ro, Yuseong-gu, Daejeon, 305-700, Korea

seungiklee@etri.re.kr, jhyice@etri.re.kr, sgkang@etri.re.kr, yichoi@etri.re.kr

Abstract— The distributed services over the Internet can be inter-connected by service compositions using Service-Oriented Architecture (SOA) and Service Delivery Platform (SDP) technologies. However, these traditional solutions have limitations on supporting the services in different domains and dynamic environments. In this paper, we propose a multi-domain service composition platform which dynamically orchestrates and composes multiple component services provided by different service providers' domains.

Keywords— Service composition, service orchestration, abstract service type, dynamic service binding

I. INTRODUCTION

With the demand of diverse use of value-added services and applications by heterogeneous users, service-oriented technologies such as Service-Oriented Architecture (SOA) [1] and Service Delivery Platform (SDP) [2] have been introduced to support a rapid and low-cost service development. Using these technologies, a service can be orchestrated and composed of one or more component services which are distributed over the Internet. Thus, one or more component services in different domains need to interoperate together. However, traditional service composition platforms lack of supporting delivery of such services because they mainly focus on the service management and delivery handled only in their own domains. In the traditional approaches, moreover, the component services are determined in a design time so that failure of a single component service may cause failure of the entire procedures of a composite service.

To address these limitations, we propose a multi-domain service composition platform (SCP) which dynamically orchestrates and composes multiple component services provided by different service providers' domains. In this platform, the component service instances for a composite service are not determined at design time. The composite service developer specifies an abstract service type which generally describes its operations and interfaces in the specification of a composite service. When executing the composite service at a service request by an end-user, the proposed platform discovers and selects component service instances which implement the abstract service types and orchestrates them. Based on this approach, the component services in different domains can be exploited in service compositions and the component services are bound in a run

time so that the composite service executions can actively adapt to dynamic changes in the service state, e.g., failure, load, and performance.

The remainder of this paper is organized as follows: in section II, we provide a brief summary about the relevant research and standardization activities. An overall description of the proposed scheme, called SCP is given in section III. System architecture for SCP is described in section IV and finally, we conclude the paper with our future work in section V.

II. RELATED WORK

The initial concept of SCP was firstly introduced in [3] as a service overlay network (SON) to facilitate the creation and deployment of value-added internet services in different network domains. A SON provider purchases bandwidth with a certain QoS guarantee from individual network domains. Thus, SON provides the users a simplified way to get QoS-controlled service delivery, which results in decoupling the application services from network services and network QoS management. The main focus of [3] is the study of bandwidth provisioning problem of the overlay links. In the similar way, the authors in [4] have focused on a design of QoS-aware routing in overlay networks (QRON) to find a QoS-satisfied overlay path with balancing the overlay traffic among the overlay nodes. They have proposed a hierarchical approach to routing of overlay brokers (OBs) which cooperates with each other to implement the proposed path selection algorithms. With the generic overlay routing protocol, QRON supports well the QoS-sensitive overlay applications by providing overlay paths which are QoS-satisfied and adaptive to the traffic loads. SpiderNet [5] was proposed to provide high quality and failure resilient service composition by exploiting probing protocol and quick failure recovery of the service paths with backup compositions.

With the increasing interest in providing a multi-domain service composition infrastructure, several standardization activities have been also conducted to define their own frameworks and unified interfaces to interact with third-party service providers in different target networks. They are OMA Service Environment (OSE) [6], Service Delivery Framework (SDF) of TM Forum [7], ATIS Service Overlay Network (SON) [8], Open service environment in NGN [9-10], and Next-Generation Service Overlay Network (NGSON) in IEEE P1903 WG [11]. Out of them, NGSON provides most similar

features with SCP. NGSON provides a framework for control and delivery of composite or base services over IP-based multiple networks with a QoS support by underlying networks. It is featured with the advanced functions of context-aware and dynamic adaptive operations in service and content delivery, including self-organization for overlay network elements and functions.

III. PROPOSED SCHEME

A. Overall architecture

A key idea of the proposed scheme is to move the core functions for service compositions and orchestrations from applications and composite service providers to an SCP provider which is loosely coupled with the composite service providers or component service providers. Based on this approach, component services are registered to and managed by SCP. These component services are used for composite services at service orchestrations by SCP.

The conceptual network model of SCP is depicted in Figure 1 with a brief summary of execution procedures of a composite service.

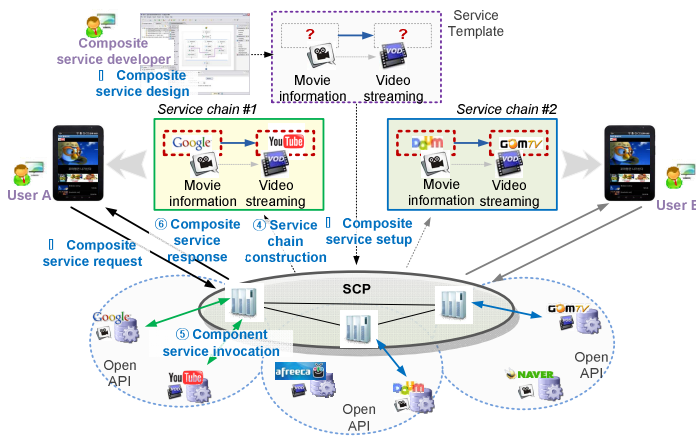


Figure 1. Conceptual network model of SCP

As depicted in Figure 1, the overall procedures of SCP can be categorized into four parts: composite service design, service chain construction, composite service orchestration, and component service invocation.

B. Composite service design

While a composite service developer designs business logic of the composite service, in the same manners of the traditional approaches, he or she splits the main functions into separate and atomic sub-functions (called service component). These sub-functions and their interactions are specified (called *service template*) using a business process description language (e.g., Business Process Execution Language (BPEL) [12]). In proposed approach, however, the sub-function is not specifically determined as an instance of the component service. The developer selects and specifies one of the abstract service types which are supported by SCP to implement the corresponding sub-function. This abstract service type

represents a common and atomic functionality which can be implemented by different component service instances. The abstract service types of component services are managed by SCP and their interfaces and required operations are described using a service type description language (e.g., Web Application Description Language (WADL) [13]).

An example of composite service template is depicted conceptually in Figure 2.

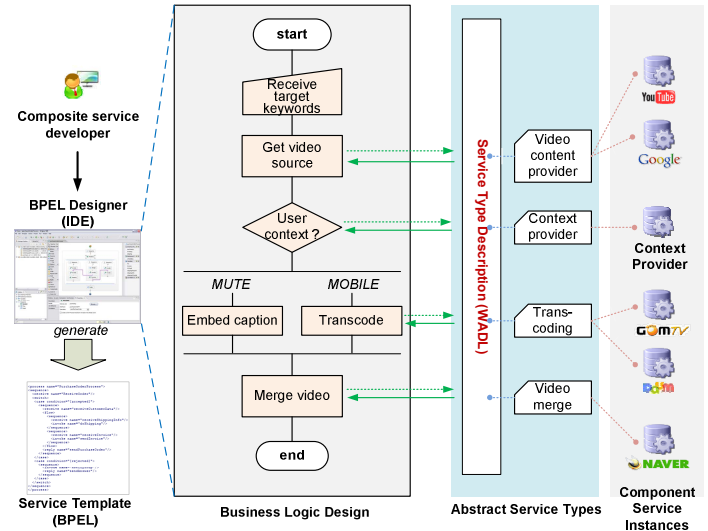


Figure 2. An example of composite service template

C. Service chain construction

According to the service template specified by a composite service developer, SCP executes the service logic with interpreting and controlling the interactions of component services. In SCP, the component services are specified in service types at a design time so that component service instances whose functions are derived from the service type should be determined at run time. A component service instance is selected for each service type out of one or more service instances in the same service type. By chaining those selected component service instances as defined in the service template, the composite service gets ready to be executed by invoking them and orchestrating their execution results. We call this procedure *service chain construction*.

The selection algorithm of a component service instance out of multiple ones may vary. In general, the status (e.g., failure), performance, load, cost, and capacity of the component service instances can be considered in the service selection. There has been related work to select most appropriate service instances for an abstract service type at service composition [14]. In order to provide better Quality of Experience (QoE) of users, they exploit context information about services, networks, and users to obtain different criteria for evaluations of component service instances. In this paper, however, we rather focus on the overall architecture for service composition platform than the details about it.

D. Composite service orchestration

Using the composite service template and service chain described earlier, SCP orchestrates the control flows of the composite service. The control flow specifies which component service should be invoked with what input parameters; and what output data should be managed and returned (see the conceptual design of business logic in Figure 2). Here we note that the component service specified to be invoked with some input parameter is an abstract service type at the orchestration. Thus, SCP maps the abstract service types to the real instances of the component service which were selected at service chain creation procedures. By invoking each component service and obtaining its execution result according to the control flows, SCP returns the final execution result of the composite service to the end-user consequently.

E. Component service invocation

For service orchestration, component service instances are selected and invoked with some input parameters. In general, service invocation is to request a component service to execute its own operation which may result in response data. In SCP, however, the invocation target at service orchestration is not a component service instance but an abstract service type. Thus, it needs additional steps for service invocations in SCP: service binding and interface translation. With the given component service instances which were determined at the service chain creation procedure, SCP extracts binding information of a component service instance needed for exchanging protocol messages with the instance. Because the interface of an abstract service type is a representative one of the instances, the different interfaces of component service instances should be translated by converting input parameters of a service type to ones of a service instance for invocations; and output parameters of a service instance to ones of a service type for further service orchestrations.

IV. SYSTEM ARCHITECTURE

In this section, we describe a system architecture and the relevant functional entities to provide the aforementioned functionalities. The overall system architecture is depicted in Figure 3.

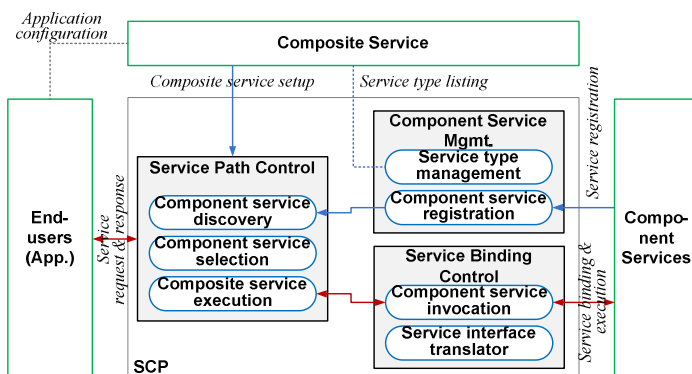


Figure 3. System architecture of SCP

There are three essential functional entities in SCP: service path control function (SPCF), component service management function (CSMF), and service binding control function (SBCF).

A. Service path control function (SPCF)

SPCF is the main component of SCP and takes a major role in service chain construction and composite service orchestration. Firstly, the composite service provider or developer requests SPCF to initialize the composite service by providing its service template. Upon an execution request to the composite service by an end-user, SPCF discovers component service instances for each service type specified in the service template from CSMF which will be described later. Among the instances, one is selected for each service type, which constitutes a service chain. Based on the given service template and the service chain created, SPCF executes the composite service by orchestrating its control flows and invoking the component service instances via SBCF which will be described later. Consequently, SPCF obtains the final execution result of the composite service, and returns it to the requestor (end-user).

B. Component service management function (CSMF)

CSMF is responsible for managing the component services types and their instances. The component service instances which are distributed in different domains over the Internet should register themselves with CSMF by identifying their target service types and mapping interfaces. This registration information is used by SPCF and SBCF for service discovery and interface translations, respectively. Service type information is also provided to composite service developers so that they can design the composite service logic with the service types given for specific operations and functionalities.

C. Service binding control function (SBCF)

SBCF is responsible for interworking with component service instances. Upon an invocation request by SPCF, SBCF extracts binding and interface information of the given component service instance from CSMF. Using this information, SBCF converts the input parameters to the format of the instance; requests the instance to execute; obtains a response to the execution request; converts the response to the output parameter format of service type; and returns it to SPCF.

The overall procedures of a composite service execution using the aforementioned functional entities are conceptually depicted in Figure 4.

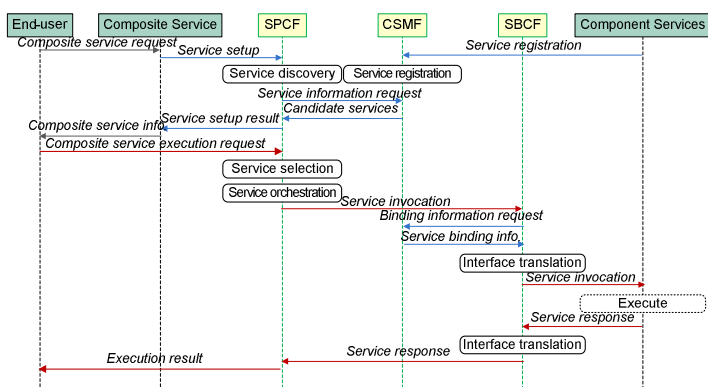


Figure 4. A conceptual procedure of a composite service execution in SCP

V. CONCLUSIONS

In this paper, we have introduced the limitations of service-oriented architectures and platforms in different service domains. To address the limitation, we proposed a service composition platform for multi-domain services, called SCP by providing its overall architecture and the corresponding system architecture. We expect that this approach would provide more flexible and easier way to create a composite service using various component services managed in different service provider domains.

A prototype of the proposed scheme with basic features of SCP is under development with the commercial web technologies [12-13,15]. Through intensive experiments of the prototype, the performance and validity of the proposed scheme will be evaluated in the future.

ACKNOWLEDGMENT

This research was supported by the KCC (Korea Communications Commission), Korea.

REFERENCES

- [1] Thomas Erl., "Soa: Principles of Service Design (First ed.)," Prentice Hall Press, Upper Saddle River, NJ, USA, 2007.
- [2] C. Pavlovski, "Service Delivery Platforms in Practice," IEEE Communications Magazine, vol. 45, no. 3, pp. 114-121, 2007.
- [3] Z. Duan, Z. Zhang, and Y.T. Hou, "Service overlay networks: SLAs, QoS, and bandwidth provisioning," IEEE/ACM Trans. Netw., vol. 11, pp. 870-883, 2003.
- [4] Z. Li and P. Mohapatra, "QRON: QoS-aware routing in overlay networks," IEEE Journal on Selected Areas in Communications, Jan. 2004.
- [5] X. Gu, K. Nahrstedt, and B. Yu, "SpiderNet: An Integrated Peer-to-Peer Service Composition Framework," High-Performance Distributed Computing, International Symposium on, Los Alamitos, CA, USA: IEEE Computer Society, 2004, pp. 110-119.
- [6] Open Mobile Alliance, OMA-Service-Environment-V1_0_2-20050803-A
- [7] TM Forum, TR139, "Service Delivery Framework (SDF) Overview, Release 2.0," <http://www.tmforum.org/>
- [8] ATIS SON Forum, <http://www.atis.org/son/>
- [9] ITU-T Recommendations Y.2240, "Requirements and capabilities for NGN service integration and delivery environment," May 2011.

- [10] ITU-T Recommendations Y.2234, "Open service environment capabilities for NGN," September 2008.
- [11] IEEE P1903, "Draft Standard for the Functional Architecture of Next Generation Service Overlay Networks," May 2011.
- [12] T. Andrews et al., "Business Process Execution Language for Web Services," v1.1, 5 May 2003, IBM developerWorks; www.ibm.com/developerworks/library/specification/ws-bpel.
- [13] S. Weerawarana et al., "Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More," Prentice Hall, 2005.
- [14] S. Lee, J. Yi, and S. Kang, "A Context-aware Service Composition Scheme for Service Overlay Network," ICTC 2011, September 2011.
- [15] C. Pautasso, "RESTful web service composition with BPEL for REST," Data & Knowledge Engineering 68 (9) (2009), pp. 851-866, Sep. 2009.