Method of Unified Communications and Collaboration Service in Open Service Platform based on RESTful Web Services

Sunhwan Lim and Hyunjoo Bae

Future Communications Research Laboratory, ETRI, Daejeon, Korea
{shlim, hjbae}@etri.re.kr

Abstract. In this paper, the functional architecture for third party call, short messaging, directory, and discussion RESTful web services was designed that enables IT developers to create applications using telecommunications network elements. In the modeling of third party call, short messaging, directory, and discussion, we proposed resource definitions and the HTTP verbs applicable for each of these resources. And we measured the TPS of the open service platform including RESTful web services above. Also, using the above model, an example service (i.e. unified communications and collaboration service) being composed of basic communication service (e.g. a third party call service, a short messaging service, etc.) and social networking service (e.g. a directory service, a discussion service, etc.) was created. Through third party call, short messaging, directory, and discussion process, the feasibility of the creation of a new service using the proposed architecture and resources was confirmed.

Keywords: RESTful Open API, Third Party Call, Short Messaging, Directory, Discussion

1 Introduction

From the viewpoint of service, integration between the wire and the wireless services is a current issue. The integration between wire and wireless services provides subscribers with the opportunity for a new of level services using the broadband capability of wired service coupled with the mobility of wireless. Open API (Application Programming Interface) can be easily used to implement or provide integration between wire and wireless services. Open API is a set of open, standardized interfaces between an application and a telecommunications network [1], [2]. This technology can provide a range of services for the integration of wire and wireless systems independently from network infrastructures, operating systems, or developing languages.

In this paper, the functional architecture for third party call, short messaging, directory, and discussion RESTful web services was designed. The architecture was implemented with Eclipse Galileo version and tested on Apache Geronimo version 2.2.1. In the modeling of the functional architecture, resource definitions and the HTTP verbs applicable for each of these resources were proposed. And the TPS (Transaction

Per Second) of the open service platform including RESTful web services above was measured. Also, using the above model, the functional architecture for an example service was designed, implemented, and tested.

2 Open API

OMA (Open Mobile Alliance) and Parlay group is to develop open, technology independent APIs that enable the development of applications capable of operating across converged networks [1], [2], [4]. OMA group defines open APIs (i.e. OMA network APIs) based on REST (REpresentational State Transfer) and Parlay group defines open APIs (i.e. Parlay X APIs) based on SOAP that enables third party applications to make use of network functionalities [7], [8], [9]. In here, use of SOAP (Simple Object Access Protocol) based APIs because of message encoding and decoding, many related stack (e.g. WS security), and so on is considered to complex [3]. Alternatively RESTful APIs using HTTP protocol, and so on are a light weight [5], [6].

3 Designed Architecture and Resources for RESTful Web Services

3.1 Functional Architecture

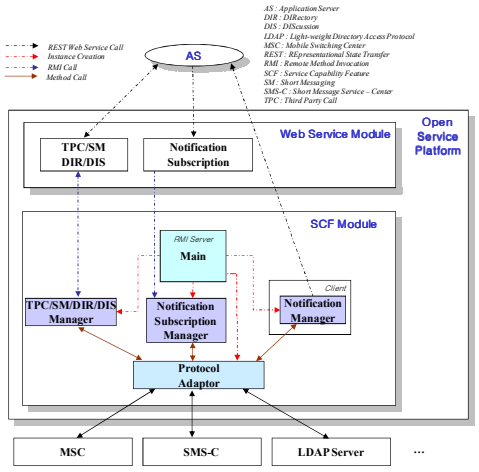

Fig. 1. Functional architecture of the third party call, short messaging, directory, and discussion RESTful web services

The functional modules of third party call, short messaging, directory, and discussion RESTful web services are illustrated in Fig. 1. This architecture is composed of a web service module and a SCF (Service Capability Feature) module. In here, the main

reason of the separation between web service module and SCF module is the effective support of services including state information like third party call, and so on. Web service module only publishes API and SCF module implements service logic of both including (i.e. stateful) and not including state information (i.e. stateless). Alternatively we can only use web service module. However, for the process of services including state information, we may implement service logic using DB including all state information or using request message including all state information parameter. This results in low performance of system.

3.2 Resources for Third Party Call RESTful Web Services

Currently, in order to perform a third party call in telecommunication networks we have to write applications using specific protocols to access Call Control functions provided by network elements (specifically operations to initiate a call from applications). This approach requires a high degree of network expertise. Alternatively it is possible to use open API approach based on web service, invoking standard interfaces to gain access to call control capabilities.

Table 1. Resources summary for the third party call RESTful web services

Resource	**URL** **Base URL: http://{serverRoot}/thirdpartycall/{apiVersion}**	**HTTP verbs**			
		GET	**PUT**	**POST**	**DELETE**
All call sessions	callSessions	get a list of all call sessions	no	**create new call session** *(callSessionId assigned)*	no
Individual call session	callSessions/{callSessionId}	**get information of an individual call session**	no	no	**terminate call session**

3.3 Resources for Short Messaging RESTful Web Services

Currently, in order to programmatically send and receive SMS it is necessary to write applications using specific protocols to access SMS functions provided by network elements (e.g. SMS-C). This approach requires a high degree of network expertise. Alternatively it is possible to use open API approach based on web service, invoking standard interfaces to gain access to SMS capabilities.

Table 2. Resources summary for the short messaging RESTful web services

Resource	**URL** **Base URL: http://{serverRoot}/shortmessaging/{apiVersion}**	**HTTP verbs**			
		GET	**PUT**	**POST**	**DELETE**
SMS message requests	/requests	return all message requests	no	**create new messages request** *(requestId assigned)*	no
Individual SMS message request	/requests/{requestId}	return one message request	no	no	no

3.4 Resources for Directory RESTful Web Services

For directory service within an enterprise, it should support the following functionalities : a polling mechanism for getting part list within an enterprise through part search keyword, a polling mechanism for getting user list within an enterprise through user search keyword, a polling mechanism for getting part profile within an enterprise through part ID, a polling mechanism for getting user profile within an enterprise through user ID, and managing contact list (e.g. get contact list, add contact, modify contact, delete contact, etc.).

Table 3. Resources summary for the directory RESTful web services

Resource	**URL** **Base URL: http://{serverRoot}/directory/{apiVersion}**	**HTTP verbs**			
		GET	**PUT**	**POST**	**DELETE**
Parts	/parts	**search part profiles**	no	no	no
Users	/users	**search user profiles**	no	no	no
Individual part	/parts/{partId}	**get part profile**	no	no	no
Individual user	/users/{targetId}	**get user profile**	no	no	no
Contact list	/contactList	**get contact list**	no	**add contact** *(contactId assigned)*	no
Individual contact list	/contactList/{contactId}	**get contact info**	**modify contact**	no	**delete contact**

3.5 Resources for Discussion RESTful Web Services

For discussion service within an enterprise, it should support the following functionalities : a polling mechanism for getting all discussion group list, a polling mechanism for getting discussion group information, and managing discussion group post (e.g. get all discussion group post list, add discussion group post, delete discussion group post, etc.).

Table 4. Resources summary for the discussion RESTful web services

Resource	**URL** **Base URL: http://{serverRoot}/discussion/{apiVersion}**	**HTTP verbs**			
		GET	**PUT**	**POST**	**DELETE**
Discussion groups	/groups	**get all discussion group list**	no	no	no
Individual discussion group	/groups/{groupId}	**get discussion group info**	no	no	no
Discussion group posts	/groups/{groupId}/posts	**get all discussion group post list**	no	**add discussion group post** *(postId assigned)*	no
Individual discussion group post	/groups/{groupId}/posts/{postId}	**get discussion group post info**	no	no	**delete discussion group post**

4 Designed Architecture for Example Service

4.1 Functional Architecture for Unified Communications and Collaboration

Functional architecture for UC&C (Unified Communications and Collaboration) is illustrated in Fig. 2.

UC&C applications are composed of the web based UC&C and the mobile UC&C. Enterprise members can connect into the web based UC&C via web browser in office and the mobile UC&C via smart phone out of office. Web based UC&C interacts with back end servers directly. However, mobile UC&C interacts with back end servers via RESTful service components on open service platform. Because mobile UC&C has a limitation on service usage, considering capacity of smart phone, security, and so on.

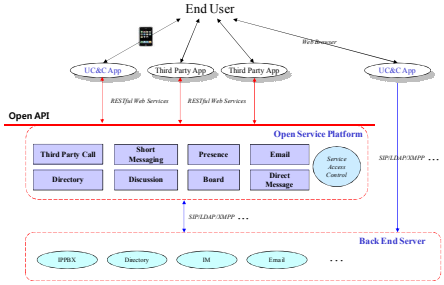

Fig. 2. Functional architecture for UC&C

Open service platform publishes service components providing functionalities of back end servers, and then third party applications use these service components. Open service platform also includes service access control function which performs authentication and authorization of service access. Service components are published to RESTful web services.

5 Implementation of the prototype function

5.1 Environments and Testing

Third party call, short messaging, directory, and discussion RESTful web services were implemented using Eclipse Galileo version and tested on Apache Geronimo version 2.2.1. These RESTful web services was composed of a web service module and a SCF module. The web service module interacts with the SCF module using RMI. And also these RESTful web services interact with a Mysql DB to record transaction history information using JDBC (Java Database Connectivity), with MSC to setup a call session between two terminals, with SMS-C to send a SMS message to a terminal, and with LDAP server to manage enterprise organization chart and contact list.

For the TPS measurement of the open service platform including RESTful web services, the above four and additional four RESTful web services were tested using SOAP UI pro version 4.0.1.

- User : 20, Count Per Second : 200, Duration (Second) : 86,400

Table 5. TPS for open service platform

Service Component	API	RESTful	TPS (Average : 295.31)
Third Party Call	makeCall	O	168
	endCall	O	
Short Messaging	sendSMS	O	327
Directory	getPartProfile	O	311
	getContactList	O	307
	addContact	O	286
	deleteContact	O	279
Discussion	addPostReply	O	295
	deletePostReply	O	282
Mail	sendMail	O	321
Presence	setUserPresence	O	363
	getUserPresence	O	322
Board	getBoardPostList	O	287
Direct Message	getNewDMCount	O	291

6 Conclusion

Regarding new market growth, a range of new intelligent services is on the horizon. Potential subscribers must be introduced to these services, but it is currently not feasible to bring third party service providers and developers into the vertical architecture of current telecommunications networks. Thus, open, technology independent APIs that enable the development of applications that operate across converged networks are necessary.

In this paper, the functional architecture for third party call, short messaging, directory, and discussion RESTful web services was designed that enables IT developers to create applications using telecommunications network elements. The architecture was implemented with Eclipse Galileo version and tested on Apache Geronimo version 2.2.1. In the modeling of the functional architecture, resource definitions and the HTTP verbs applicable for each of these resources were proposed. And the TPS of the open service platform including RESTful web services above was measured. Also, using the above model, the functional architecture for an example service was designed, implemented, and tested. Through the process, the feasibility of the creation of a new service using the proposed architecture and resources was confirmed.

Acknowledgement. "This research was supported by the KCC (Korea Communications Commission), Korea, under the R&D program supervised by the KCA (Korea Communications Agency)" (KCA-2012- (09911-05001))

References

1. 3GPP, Third Generation Partnership Project, http://www.3gpp.org/
2. OMA (Open Mobile Alliance), http://www. openmobilealliance.org/
3. W3C, World Wide Web Consortium, http://www.w3c.org/

4. Roy Fielding, “Architectural Styles and the Design of Network-based Software Architecture”, Dissertation of Doctor of Philosophy in Information and Computer Science, University of California, IRVINE (2000)
5. Leonard Richardson and Sam Ruby, RESTful Web Services, O’Reilly Media, (2007-5)
6. Cesare Pautasso, “REST vs. SOAP: Making the right architectural decision”, 1st International SOA Symposium (2008-7)
7. Parlay X Working Group, Parlay X Web Services White Paper v1.0 (2002)
8. Web Services Working Group, Parlay Web Services WSDL Style Guide (2002)
9. Parlay X Working Group, Parlay X Web Services Specification v1.0 (2003)