International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:2, No:7, 2008

A Web Services based Architecture for NGN Services Delivery

K. Rezabeigi, A. Vafaei, N. Movahhedinia

***Abstract*—**The notion of *Next Generation Network (NGN)* is based on the *Network Convergence* concept which refers to integration of services (such as IT and communication services) over IP layer. As the most popular implementation of *Service Oriented Architecture* (SOA), *Web Services* technology is known to be the base for service integration. In this paper, we present a platform to deliver communication services as web services. We also implement a sample service to show the simplicity of making composite web and communication services using this platform. A Service Logic Execution Environment (SLEE) is used to implement the communication services. The proposed architecture is in agreement with Service Oriented Architecture (SOA) and also can be integrated to an Enterprise Service Bus to make a base for NGN Service Delivery Platform (SDP).

***Keywords*—**Communication Services, SOA, Web Services, NGN, SLEE.

I. Introduction

THE forthcoming trend of next generation networks, NGN, is toward the integration of various types of traffic such as voice, video, and data over packet networks. According to ITU definition [2], NGN is governed by the following rules:

- A packet-based network able to provide services, including Telecommunication, and also capable to make use of multiple broadband, QoS-enabled transport technologies.
- In which Service-related functions are independent from underlying transport-related technologies.
- Offers unrestricted access by users to different service providers.
- Supports generalized mobility which will allow consistent and ubiquitous provision of services to users.

NGN services will include session-based services, such as IP telephony, videoconferencing, video chatting, and non-session-based services such as video streaming and broadcasting. NGN, for both public and enterprise environments, will consist of converged networks, using fixed and wireless, as well as circuit-switched and packet-switched infrastructures. The key promise of NGN lies not only on the ability to interconnect these diverse transport technologies, or the potential cost reductions obtained by doing so, but also on the ability to develop and deploy innovative services rapidly and efficiently. So the NGN main idea focuses on the "service" concept in two ways:

- Independency and isolation of service layer from transport layer.
- Simplicity and efficiency of communication service development, deployment, and delivery.

A critical ingredient for service introduction is the rapid development of open Application Programming Interfaces (APIs) that span diverse networks, allowing 3rd party application developers to produce new services analogous to the development of software in the Information technology (IT) industry. In the past few years several industry efforts have emerged to develop such open APIs, including JAIN SIP [7], JAIN SLEE [8], and SIP Servlet [9]. These APIs help to realize the first rule of NGN mentioned earlier, which implies that communication services can be developed regardless of lower layers technologies. However, realization of a Service Delivery Platform requires more abilities such as Service Discovery, Service Registration, and Service Management. These capabilities rely on Service Oriented Architecture (SOA) [4], so a service oriented telecom network is required to satisfy the needs of service oriented telecom businesses [11].

A Service Oriented Architecture (SOA) is an architectural style whose goal is to achieve loose coupling among interacting software components, and component behaviors are defined completely by contracts and APIs. SOA reorganizes existing software applications and component units into a set of self-contained and self-describing services, with standard interfaces and messaging protocols. These services can be accessed without the need of traditional point-to-point communications, based on different protocols. The SOA paradigm allows the complex business processes and transactions to be delivered as integrated services and let applications be reused everywhere and by anybody.

A basic SOA includes three fundamental procedures: Service Providing, Service Registry, and Service Client with three important functions: Service Publishing, Service Discovery, and Service Binding. SOA defines an interaction between service clients and service providers. Providers are responsible for publishing a description of the services.

K. Rezabeigi is with the Department of Computer Science, University of Isfahan, Isfahan, Iran (phone: +98 9125032810; e-mail: K_Rezabeigi@eng.ui.ac.ir).
A. Vafaei is with the Department of Computer Science, University of Isfahan, Isfahan, Iran (phone: +98 9125032810; e-mail: Abbas_Vafaei@eng.ui.ac.ir).
N. Movahhedinia is with the Department of Computer Science, University of Isfahan, Isfahan, Iran (phone: +98 9125032810; e-mail: Naserm@eng.ui.ac.ir).

Clients should be able to find the description of services and to bind them with proper connections.

Web Services technology is the most popular implementation platform for SOA [6]. Web Services are distributed software components which can be described, published, discovered, and invoked with standard protocols. Web Services communicate using XML and Web protocols, which are pervasive, working both internally and across the Internet. They support heterogeneous interoperability and use SOAP for service calls, WSDL for service descriptions and UDDI for service registration and discovery.

The idea behind this paper is to provide an appropriate architecture to deliver the communication services as web services in order to be used simply by web service clients in converged applications. Web services are mostly supposed to be accessed by a synchronous request-reply manner which makes them simple to be implemented and used. In contrary, communication services are asynchronous and event-driven in nature. These types of services are delivered using a sequence of send and receive messages. For example a web service like search is accessed through a simple search request but a communication service like Make-Call service requires a sequence of event passing to be performed (Sending the MakeCall event, receiving the proceeding, and alerting and answering events and also exchanging the media capabilities of call). So delivering a communication service as a synchronous web service encounters some difficulties.

Some works have been done to integrate web and communication services. Some of them suggest a service oriented model for telecom service businesses [11], [14]. Some others use asynchronous web services to deliver the telecom services such that the asynchronous nature and complex requirements of these types of services are preserved and also appears in service clients [13]. References [1], and [12] try to make a replacement for call control protocols, such as Session Initiation Protocol (SIP), using web services protocols, translating SIP messages into proper SOAP messages and defining a gateway between SIP and Web services applications. This approach has two problems; firstly the complexity of communication services is transferred to client because of their event-based nature. Secondly complete replacements for SIP protocols by SOAP messages are needed such that most of SIP functionality be covered. Moreover, the isolation of service layer and transport layer as mentioned in NGN is not maintained. Furthermore, most of the mentioned efforts are done based on SIP, and are not applicable to protocol independent services and environments.

In this paper we try to propose a proper architecture to deliver communication services as synchronous web services in such a way that these services are independent of lower layers and technologies such as call control protocols and transport managements. In order to do so we first provide a synchronous view of some of communication services, so that they can be called in a synchronous manner, while they are asynchronous in nature. We implement a simple call service using this technique which can make a call between two SIP user agents and have control on the call state to monitor, change and terminate the call. The rest of this paper is arranged as follows. Section 2 gives a brief overview of the communication service, its definition and how it works. Section 3 describes our overall architecture and describes the major components and technologies to solve the problems stated in section 1. Section 4 briefly describes the structure of the web service. We end the paper with a brief summary and conclude with a note on future works.

II. Service Call Model

As mentioned above we try to have a synchronous view of communication services. For example any call control protocol such as *Make-Call* service is asynchronous in SIP. These protocols should be redefined in such a way that they can be called in a synchronous manner. As can be seen in Fig. 1, we redefined this service as a communication service so that it can be called synchronously. In a normal situation a SIP user agent calls another one using an event passing sequence. In our case, we have a simple client which can use the service in a synchronous manner, (for example a web application, which needs to initiate a call between two SIP UAs). Therefore we need to have a service which acts an abstraction between communication and web, or synchronous and asynchronous environment. This service interacts synchronously with its client and with communication terminals or SIP UAs in an asynchronous manner.

During this scenario we need to focus on two goals. Firstly interact with SIP UAs according to standard SIP Call message sequence so that call be made correctly, and secondly exchange media capabilities (SDP contents) between two SIP UAs. These goals can be reached using the message sequence is shown in Fig. 1.

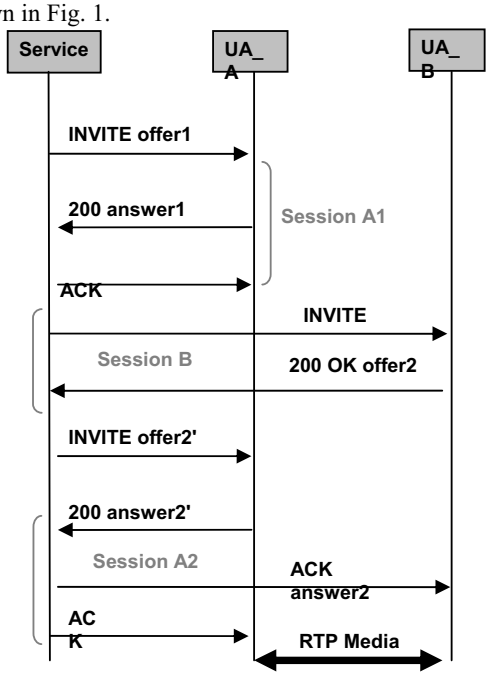

Fig. 1 SIP message sequence of service

As can be seen, the main problem is to make an agreement on session description between two SIP UAs. As shown above, service can run a reasonable scenario of offer-answer, compatible with Session Description Protocol (SDP) [5] between two UAs. During running of above scenario, the service and also call have some states and state changes that are different from normal SIP call state machine. Service state machine shows the state change of service, from initial state until its final state. This differs from the call state machine but depends on it. These two state machines are shown in Fig. 2.1 and 2.2 respectively.

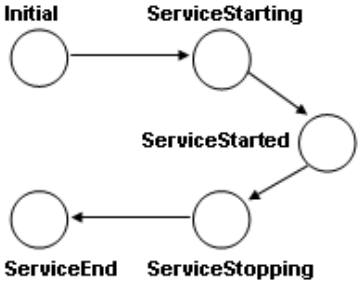

Fig. 2.1 Service State Machine

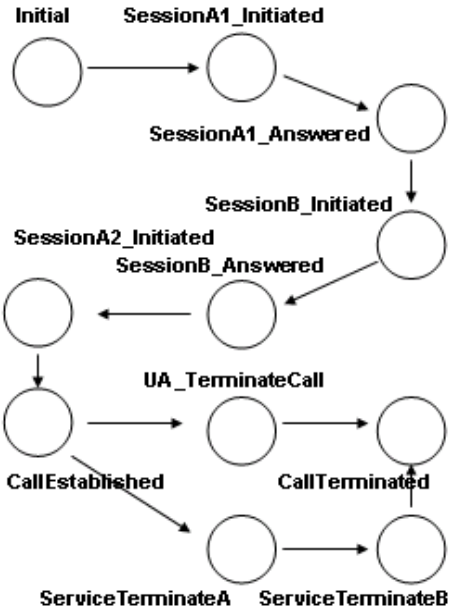

Fig. 2.2 Call State Machine

III. Architecture

The proposed architecture is to deliver the communication service as synchronous Web Services and simplify its integration into normal web services and web applications. This architecture contains a telecom server, a web service container, and a web server. Deployment view of this architecture is shown in Fig. 3.

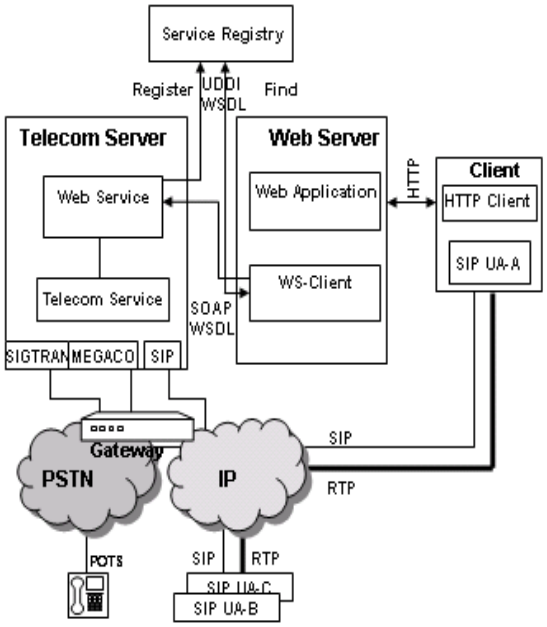

Fig. 3 Deployment view of proposed architecture

At first, a web service as an agent of a telecom service (for example a Make-Call service) registers itself into a service registry like UDDI. The main service use case starts from a http client. This client can send a request to a web application on a web server to execute a service. Web application will find the suitable web services through UDDI protocol and bind to it requesting a specified web service to be run. Web service is a synchronous service but interacts with a telecom service in an asynchronous way to initiate the communication service. For doing so, it requires to interact with the service using a message passing sequence. So service starts to run and initiates a call between two SIP UAs according the scenario described above. State of the call is exported to the web service client and also http client so that they can control, monitor and change the call (for example hold it), and also terminate it at the end. This paper mainly focuses on the web service and communication service and how they interact with each other.

We use a SLEE (Service Logic Execution Environment) as a telecom service container. A SLEE is a high throughput, low latency event processing application environment. JSLEE [8] is the Java standard API and component model for SLEE. The JSLEE specification is designed so that implementations can achieve scalability and availability through clustering architectures and the point of integration for multiple network resources and protocols. JSLEE has several components such as Service Building Block (SBB), Resource Adaptor, Service, and Event. SBB are reusable components that are sensitive to specific events. They receive those events and handle them. Each service is composed of some SBBs. Resource Adaptor (RA) is a connector to connect the SLEE platform to a specific protocol stack, such as SIP and ISUP RA. Events are

defined messages that are exchanged between JSLEE components.

In our proposed architecture we need to implement some new type of these components. Firstly the web service receives a request to initiate the service. It must start the service through firing an event. So we need to define some events that web service use to interact with JSLEE platform and communication services. These events are:

- MakeCall (sipUri1, SipUri2)
- CancelCall(callId)
- TerminateCall(callID)

The web service will fire these events to JSLEE platform and its components, so we need a specific Resource Adaptor to receive event and deliver them to SBBs. We name it Call Control Resource Adaptor (CCRA). In order to fire event from web service to JSLEE application we use JMS technology [10]. It is the asynchronous messaging technology of J2EE platform. Our CCRA can be a JMS Listener application that listens to a Message Queue for mentioned Call Control (CC) messages. After that web service fires CC messages to message queue, CCRA receives that as JMS messages and then fire equivalent JSLEE events to JSLEE event router. Event router then delivers them to those SBBs that register for them. So we need a Call Control RA based on JMS technology that relies on a message queue.

IV. Implementation

As mentioned above we have two state machines in our service, call and service state machines. So we need two Sbbs, one for each state machine. ServiceSbb registers for call control events and CallSbb for SIP messages. Firstly when MakeCall event delivered to SLEE, ServiceSbb is instantiated and receives the MakeCall event , initiate call through sending Invite message to SIP UA-A and instantiating CallSbb. After this, CallSbb will work according to call scenario mentioned above. It handles SIP messages in order to make a call between two SIP UAs.

Web Service will have control on the call through sending appropriate event such as CancelCall, TeminateCall to CCRA. Another call control events also are delivered to ServiceSbb. Handling of these events requires some SIP message passing and so is in the scope of CallSbb. So Service Sbb delegate this to CallSbb through calling some exported methods of CallSbb that can terminate or cancel call via firing appropriate SIP messages.

Another requirement in converging web and communication application is to monitor the state of communication services in the web applications. For example in our case, we need to show the state of the call (for example active, cancelled, on hold, or terminated states) in web application so that user know about call and can control it. The problem is that state change is occurred asynchronously in the service and can be initiated by a resource other than web application, and we should inform it about state change. One solution is that state of call be a shared object between communication service, web service and web application. So web application can monitor service state. In addition, web application requires call identifier (call id) to control and refer to it.

To make a call we send a MakeCall event to SLEE and have no return value. So event handler method can not return call id. In order to solve this problem we use a guid (global unique id) as id of the event. Then the ServiceSbb can save call id with this guid key in a map that is shared between web and communication service. So web application can have access to call and its status. This guid is the id of call in the web application and so must be accessible during a session. So we need to save it in the session object as a session attribute. Putting it in another way one can say that we have a one-to-one mapping between HTTP and SIP sessions. The implementation view of architecture is shown in Fig. 4.

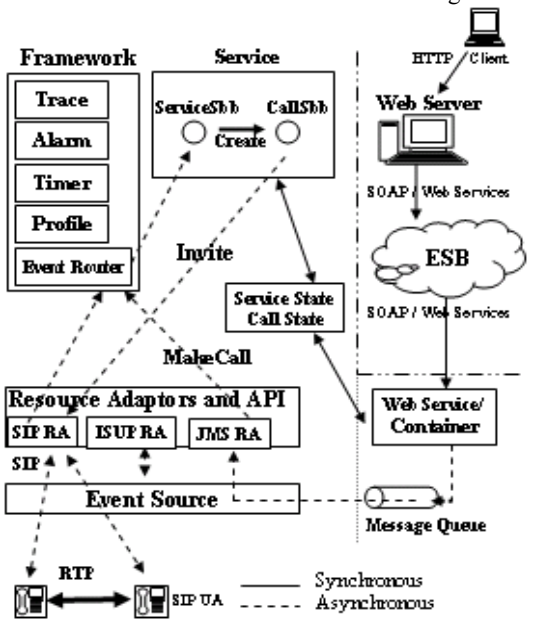

Fig. 4 Implementation view of proposed architecture

V. Web Service Structure

In order to export our communication service as a web service we should describe it in a WSDL document. Our service will provide these operations to its clients:

- callGuid MakeCall(sipUri1, sipUri2);
- String getCallInfo(callGuid);
- cancelCall(callGuid);
- terminateCall(callGuid);

Our service description contains the detailed description of the operations that are exported to clients. Here is part of WSDL file that contains the description of our web services:

```
<wsdl:operation name="makeCall">
  <soap:operation soapAction="" style="document"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
```

```
  </wsdl:output>
  <wsdl:fault name="ServiceException">
 <soap:fault name="ServiceException" use="literal"/>
  </wsdl:fault>
  <wsdl:fault name="PolicyException">
   <soap:fault name="PolicyException" use="literal"/>
  </wsdl:fault>
</wsdl:operation>

<wsdl:operation name="getCallInfo">
   <soap:operation soapAction="" style="document"/>
  <wsdl:input>
     <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
  <wsdl:fault name="ServiceException">
 <soap:fault name="ServiceException" use="literal"/>
  </wsdl:fault>
  <wsdl:fault name="PolicyException">
   <soap:fault name="PolicyException" use="literal"/>
  </wsdl:fault>
</wsdl:operation>
……
…….
```

As can be seen, MakeCall operation has two operands of type string that are the SIP URIs of two SIP UAs, and a string as the return value that is an id for referring to the call. The getCallInof operation has one operand that is the guid of the call and a string type return value that states the status of the call.

VI. Conclusion

In this paper we had look at the convergence of communication and web services as one of the NGN main ideas, and the main differences between these two types of services and also the problems encountered in accomplishing the integration into a composite service. As mentioned, the main difficulty arises from the synchronous nature of popular web services and asynchronous nature of communication services. We proposed an architecture, based on a different point of view to some of communication services that make them synchronous so that make it easy being delivered as a synchronous web service. A sample MakeCall service is also implemented and used in a web application as the proof of architecture. This service is able to make a call between two specified SIP UAs and then monitor the state of call and control it. Implementation is based on JSLEE technology as a telecom container. Architecture is protocol-independent but we use SIP as signaling protocol to implement the sample service. The proposed architecture is also in agreement with the idea of Service Oriented Architecture (SOA) [4] in that:

- Services (basic network capabilities, convergent services) are independent from each other.
- Services are open and scalable, which can be easily converged, that is, added into, or deleted from another service when needed.
- Such service convergence process can be done anywhere and anytime.

This work can be followed up by integrating our architecture to an enterprise service bus (ESB) and making some innovative and more complex composite services. ESB can be a base for providing a Service Delivery Platform (SDP) for communication services used in IP Multimedia Subsystems (IMS).

References

[1] Feng Liu, Wu Chou, Li Li and Jenny Li, *"*WSIP–Web Service SIP Endpoint for Converged Multimedia/Multimodal Communication over IP*",* Proceedings of *IEEE International Conference on Web Services (ICWS'2004)*, July 2004.

[2] ITU, "Definition of Next Generation Network", http://www.itu.int/ITU-T/studygroups/com13/ngn2004 /working_definition.html.

[3] IETF, J. Rosenberg, et al., "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[4] M. N. Huhns and M. P. Singh, "Service-oriented computing: key concepts and principles," *IEEE Internet Computing*, vol. 9, pp. 75-81, Jan-Feb 2005.

[5] SDP: Session Description Protocol, http://www.ietf.org/rfc/rfc2327.txt.

[6] Web Service Architecture, W3C Working Draft, http://www.w3.org/TR/2003/WD-ws-arch-20030808.

[7] Sun Microsystems, "JSR-000032 JAIN™ SIP Specification", http://jcp.org/aboutJava/communityprocess/final/jsr032.

[8] Sun Microsystems, "JSR 22: JAIN™ SLEE API Specification", http://jcp.org/en/jsr/detail?id=22.

[9] Sun Microsystems, "JSR 116: SIP Servlet API", http://jcp.org/en/jsr/detail?id=116.

[10] Sun Microsystems, "JSR 914: Java™ Message Service (JMS) API", http://jcp.org/en/jsr/detail?id=914.

[11] Weifeng Lv Jianchu Kang Wei Chen Ran Lei, "Integration and Application Platform of Service-Oriented Telecom Businesses", presented at *Fourth European Conference on Universal Multiservice Networks, ECUMN '07.* , February 2007.

[12] Wu Chou Li Li Feng Liu , "WIP: Web Service Initiation Protocol for Multimedia and Voice Communication over IP", Avaya Labs Res., Basking Ridge, NJ, presented at *International Conference on Web Services, ICWS '06*, September 2006.

[13] Wu Chou Li Li Feng Liu, "Web service enablement of communication services", Proceedings of IEEE International Conference on Web Services (ICWS'2004), December 2005.

[14] Xinyu Wang, Jianchu Kang, "Service-oriented business integration and management in telecom", International Conference *on Services Systems and Services Management*, Proceedings of ICSSSM, June 2005.