WSIP – Web Service SIP Endpoint for Converged Multimedia/Multimodal Communication over IP

Feng Liu, Wu Chou, Li Li, and Jenny Li
Avaya Labs Research, 233 Mt. Airy Road, Basking Ridge, NJ 07920, USA
{fliu1,wuchou, lli5, jjli} @avaya.com

ABSTRACT

We present an approach for converged communication services over IP, based on the concept of WSIP (Web Service SIP). In our approach, each WSIP node is both a SIP endpoint that communicates in the SIP world through SIP signaling, and a web service (SOAP) node that provides a native and generic service integration environment for binding SIP based communication in web services. The combination of SIP with web service provides a new converged communication paradigm. It allows dynamic service discovery and binding that integrate SIP as a component in the business transaction. The clear separation of service integration environment and SIP signaling in our approach maintains the simplicity and efficiency of the existing SIP protocol, while leverages the power of both methods in IP based communication services. The ubiquitous service integration nature of web service provides a disciplined solution to maintain and update large number of SIP endpoints from different installation bases. It enables dynamic monitoring, repairing, and updating SIP UA endpoints, which is essential to the success of VoIP using SIP. The proposed approach was implemented in a prototype research system, and several advantages of WSIP over the existing approaches are observed.

Keywords

Web service, Session Initiation Protocol, service integration

1. INTRODUCTION

SIP (Session Initiation Protocol) [1] is one of the most important protocols for VoIP (voice over IP) communication. The explosive growth of IP network and the need of voice service over IP as an alternative to PSTN (Public-Switched Telephone Network) greatly accelerate the acceptance of SIP as the primary method for the next generation VoIP in the converged communication environment.

From internet protocol stack point view, SIP is an application layer protocol for VoIP services. It incorporates elements from two most important application layer protocols of the internet, HTTP (Hyper Text Transport Protocol), and SMTP (Simple Mail Transport Protocol). It borrows from HTTP the concept of the request/response client-server design and the use of uniform resource locators (URLs). The signaling part of SIP reuses the SMTP headers, such as To, From, Subject, Date, etc., and it inherits the basic SMTP simple text encoding scheme and header style. The VoIP call in SIP can be initiated and setup by some basic request/response methods, such as the original six methods, INVITE, REGISTER, BYE, ACK, CANCEL, and OPTION [1].

Despite the success of SIP as an efficient VoIP protocol, SIP is under constant expansion in order to support new communication services which are not covered in its original scope, such as messaging [9], authentication, presence, etc. In order to integrate SIP for converged communication services, there is a critical need for an extensible service integration environment that allows applications to integrate SIP as part of the business transaction process. The simple text encoding scheme and SMTP type headers in SIP can be both its strength and limitation. The advents of XML based protocols for switching, call control, and service monitoring, such as CSTA[4,5,6] and CCXML[10], have made this issue even more acute.

Web service [2,3] is a major development for providing services over IP. It is based on the concept of using a structured XML document, in the form of a standard based SOAP message [7, 8], to access, control and integrate various services remotely for complex business transactions. In web service world, the services for business transactions are linked together through a collection of web service SOAP nodes. SOAP provides an extensible and generic XML framework for XML based service function calls and data exchange format. The binding of SOAP with HTTP makes web service ubiquitous, accessible from anywhere through the internet.

However, SIP and web services are rarely cross each other in the past, and each method tries to address its needs from its own domain through its own extensions. This leads to some serious issues, and the strengths of both methods are not fully leveraged. In this paper, we describe a new approach for converged communication services over IP based on the concept of WSIP (Web Service SIP). In WSIP paradigm, each WSIP node is both a SIP endpoint that communicates in the SIP world through SIP signaling, and a web service (SOAP) node

that provides a native and generic service integration environment for binding SIP based communication in web services.

WSIP is a fundamental departure from the prior practice of mixing service integration environment in SIP headers and signaling. It takes the separation of signaling and media transmission used in SIP to the next level of separating service integration from signaling. WSIP provides an extensible and modularized three-tier infrastructure among service integration, signaling, and media transmission. WSIP is a combination of SIP and Web Services and a new paradigm for converged communication over IP.

WSIP provides tremendous flexibilities for integrating SIP in business transactions. The light weight protocol and efficient media stream transmission in SIP complement the message based web services, where carrying media stream in SOAP messages can be costly. On the other hand, SIP protocol standards as well as the customized SIP applications are under constant revision and extension, leading to various versions of SIP user agent (UA) endpoints. The interoperability between SIP UA endpoints has become a serious issue for VoIP using SIP. The ubiquitous service integration nature of web service provides a disciplined solution to maintain and update large number of SIP endpoints from different installation bases. The web service mechanism in WSIP for dynamically discover and update SIP UA endpoints is essential to the success of VoIP using SIP.

This paper is organized as follows. We discuss the related work in Section 2. The approach of WSIP and the structure of WSIP terminal are presented in Section 3. We describe a prototype WSIP terminal implementation in detail and address the issues of using web service in a situation that requires the maintenance of the service state, such as in a SIP UA. In Section 4, we study the event notification and communication service model under the paradigm of WSIP. The integration of WSIP with XML based switching, call control and service applications, such as CSTA and CCSML, is discussed in Section 5. The proposed WSIP approach has been implemented in a research prototype system. The performance advantages of the proposed WSIP approach are highlighted in Section 6.

2. RELATED WORK

SIP protocol defines a basic set of request message methods, such as INVITE, INFO, ACK, etc. These request messages in SIP are used to initialize and set up calls through the signal channel. Numerical values are used to indicate the status of request/response (e.g. 200 indicate O.K., etc.), when two parties in a peer-to-peer communication negotiate a session. The media transmission in SIP is carried over by a separate channel using real time media transport protocol such as RTP.

The separation of signaling and media transmission in SIP has the advantage of being a light weight application level protocol for fast call setup and termination, because no real media is transmitted through the call setup SIP messages. This design is very different from HTTP, which is based mainly on two fundamental methods GET and POST. The rich content and control in HTTP are conveyed through the structured XML/HTML documents, which have an extensible markup language structure. In order to support messaging, MESSAGE requests are added to the SIP protocol. MESSAGE requests are very different from other SIP requests in that they carry the media, in the form of IM (Instant Messaging), as payload [9]. This broke the convention that SIP payloads carry signal information not the media. It has caused concern that when a SIP infrastructure is shared between a call signaling and IM, the IM traffic may interfere with call signaling traffic. Congestion control can be an issue. However, since MESSAGE traffic patterns are likely to be different from other SIP methods, MESSAGE request is treated as a special exception.

In order to integrate SIP in business transactions through XML, efforts are made to embed XML based service control/response in the original SIP request messages, such as INVITE, INFO, SUBSCRIBE, NOTIFY, as part of its message body. This pragmatic approach encounters various problems. For example, the use of INFO method to carry XML service control\response is discouraged, because INFO in SIP is used by a SIP UA to send call signaling information to another SIP UA, and it lacks semantics to support such use. One other disadvantage is that it is not analogous to the existing use of INFO for carrying a peer-to-peer communication protocol.

Another attempt is to embed XML based service control/response in the SDP (session description protocol) part of SIP INVITE. However, SDP is more of a description syntax than a protocol. It does not provide a full-range of control/response negotiation capability. SDP is not designed to be easily extensible, and parsing rules are strict. This again departs from the existing use of SDP in INVITE. Moreover, the receiving SIP UA can ignore the INVITE SDP and respond with its own SDP for call setup.

Direct embedding service control/response in the existing SIP signaling has some serious drawbacks. First, it deviates from the existing use of the SIP protocol, and departs from the intent of being a light weight application layer protocol that separates signaling from media. The embedded media or service control/response can interfere with the original SIP signaling. Because of lacking an XML standard based extensible structure, there is a serious danger of introducing inconsistencies among SIP UAs, and service interoperability can be in real risk. Secondly, there is no standard service integration

environment for SIP endpoints. One consequence of direct embedding service control in SIP signaling is that for an application to drive a SIP endpoint, it has to have in itself a SIP endpoint, because all service integration controls are communicated through SIP call control signaling. This may not always be feasible and incur unnecessary overhead, especially for XML based applications.

Our approach of WSIP, on the other hand, provides a unified interface for service integration with SIP. The use of SOAP in WSIP provides an extensible and well-formed structure for remote service control and data exchange. The unification of SIP with Web Service provides a new converged communication paradigm. It allows dynamic service discovery, repair/update, and binding that integrate SIP as a component in the business transaction. The clear separation of service integration environment and SIP signaling in our approach maintains the simplicity and efficiency of the existing SIP protocol and leverages the power of both methods for converged communication services over IP.

3. WSIP (WEB-SERVICE SIP)

The conventional SIP endpoints has two major components, a SIP user agent (UA) and an application that is built on top of the SIP UA, such as a SIP phone based application, an IM application, etc. The SIP UA handles the low level call control and media communication. It exposes a set of APIs to its application. In the SIP world, the application interacts with other users or applications by calling the SIP UA APIs that communicate with other SIP UAs through SIP signaling. Applications are tied to the particular SIP UA APIs, and constrained to operate within the support of SIP protocol.

On the other hand, a web service endpoint is typically a SOAP node. A SOAP node is both a client and a server. It listens to the SOAP channel for the incoming SOAP messages and sends the proper SOAP messages as its response. Web service is based on a loosely coupled service model. It separates the service protocol and application data from the application layer protocol. It encapsulates the service protocol and application data in a well-formed SOAP message. SOAP is an XML standard from W3C. It provides a standard format for XML based remote procedure call and data exchange. This allows web services to be supported on multiple application layer protocols, such as HTTP and SMTIP.

Moreover, web service provides a standard service integration environment through WSDL (web service description language). It allows dynamic application service binding between the SOAP message and the platform service APIs. This leads to a distributive service model, where multiple services and platforms can be linked remotely through the exchange of SOAP messages to perform business transactions.

The key idea behind our WSIP approach is not to further complicate the SIP signaling protocol, but to expose the SIP UA as a SOAP node in the world of web service. By doing so, it separates the signaling and service integration environment through the loosely coupled web service model. It provides an extensible and standard based XML service integration environment through the use of SOAP/WSDL, which allows easy integration of SIP in business transactions. In our approach, a WSIP terminal can be integrated with any authorized application remotely and in a distributed fashion through the standard web service method. There is no need to change the SIP signaling protocol, and it does not require that an application has to own a SIP UA in order to transmit the service content and control. The ubiquitous of the service integration environment is achieved through the binding of SOAP with HTTP and the common service description interface of WSDL.

Figure 1 illustrates the communication diagram of WSIP. As shown in the diagram, two WSIP terminals can communicate directly using SIP signaling (represented in solid line). Moreover, two WSIP terminals can also communicate directly through web service using SOAP. In addition, the WSIP terminals can communicate with remote applications that bundles SIP in service directly through web service channel (represented in doted line). For example, a SIP phone call to alert the customer of the arrival of his product order can be initiated by the application of a remote agent who travels on the road with only a web connection.

3.1 The Structure of WSIP Terminal

The structure of WSIP terminal is illustrated in Figure 1. It has four major components: SIP UA, Wrapper Layer Controller (WLC), SOAP Server, and SOAP Client. The SIP UA in WSIP is responsible for SIP signaling and media transmission. It handles functions such as making a SIP call, answering an inbound SIP call, disconnecting the call, holding the call, etc., according to the SIP protocol. It also sends the call related events to WLC, such as call arrival, call hang-up, media change, etc. The SIP UA exposes a set of APIs to WLC for basic communication service using SIP, such as MakeCall, SendIM, HandupCall, etc. The WLC wraps the API functions of SIP UA into XML and exposes them as web services. One embodiment of our WSIP implementations contains the following web service functions.

1. Receive and parse the SOAP request at the WSIP SOAP Server.
2. Bind SOAP service request to WLC according to the WSIP WSDL specification, and execute the request by calling the corresponding SIP UA API functions.
3. Receive the SIP event from SIP UA, propagate the even to WLC.

4. Package the event into SOAP message, and send the event notification to the application through the WSIP web service SOAP Client.
5. WSIP system services (described below).

The WSIP system services expose a set of web service functions for SIP UA configuration, testing, maintenance, and update. It allows a SIP service application to dynamically discover and configure the SIP UA functionalities through SOAP/WSDL web service interface. The SOAP message for WSIP system services can contain instructions to WLC with configuration parameters. For example, the configuration message can include the maximum number of calls that the SIP UA of the WSIP terminal is allowed to handle, the media type and formats that the SIP UA can access, the restriction to use certain type of media coders by the SIP UA of the WSIP terminal, a very useful feature to control the network congestion.

The WSIP system services allow WSIP terminal to be maintained and updated on demand and dynamically. System administrators can use web service to install new version of the SIP UA components through WLC. With proper authorization, the WLC of WSIP can install new SIP UA components automatically without user's intervention. This capability of WSIP is extremely important for communication using SIP, because there is a large installation base of SIP phones, and SIP standard is under frequent revision. Maintaining the interoperability between SIP UAs from different installation bases has become difficult. Most importantly, the discovery capability of the web service combined with WSIP system service enables the system administrator to monitor, discover, update and even repair a WSIP terminal. The functionalities of WSIP can be published through the standard web service method, and the WSDL based service integration environment makes it easy to create client applications.

3.2 Web Service Implementation in WSIP Terminal

The web service on WSIP terminal is a stateful web service. It contains the status of the SIP UA, such as the number of active calls, the port number that is using, etc. It also contains the status of each active calls, for example, session type (Voice, IM), the state of the session (ringing, call is preceeding), caller ID, starting time, etc. So it is important to guarantee that all the states can be kept correctly. The situation is very different from using the SIP protocol where certain state information can be carried in the SIP signaling. The GET and POST methods in HTTP, that carry the web service SOAP messages, are stateless. It is up to the application to maintain and manage a stateful transaction. This situation is further complicated by the multi-thread nature of the WSIP terminal, where these threads can be initiated by either the SIP UA or the application through the WSIP terminal web service.

This requires WSIP web service infrastructure to maintain a dialogue history registry, and the rule is that if a contact is initiated or answered by one application, all the subsequent contact events have to be forwarded to the same application. For example, when an application initiated a SIP call through the web service of WSIP, then all subsequent events of that application have to interact to the same thread on the SIP UA, until the final call hang up.

To address these issues, we developed a two-way proxy web service middleware to manage the many-to-many two-way connections between the WSIP web service agent and the application. It consists of the following technical procedures.

- Use XML condition/action rules to specify which application to launch based on the contact events from the SIP UA of WSIP terminal.
- Maintain the web service registry for the contact from the WSIP SOAP server.
- Maintain the sessions between the external applications and the applications running on SIP UA, if necessary.

Each external application web service has a unique service key, which is a URI, and when a new contact is routed to WSIP terminal, the event XML includes the service key, event type and contact description, among other information. If necessary, the proxy will launch the specified application, and create a session that maintains the relationship between the external web service that submits the application and the state of the application that is running on the WSIP SIP UA. This process will be discussed further in the WSIP event notification at Section 4.

In our implementation of this architecture based on Microsoft SOAP Toolkit 3.0 [13] and IIS web server on Win2K, we encountered the problem that the proxy is treated as a stateless object under ISAPI mode. To solve this issue, we implemented our web service proxy as a COM+ service that use the Singleton pattern to represent the session table, so that the session connections between web service proxy and SIP UA can be maintained through out the interaction.

Another issue is how to make the web service thread safe. Since the WSIP web service allow multiple applications to assess the SIP UA services simultaneously, it is important for the SIP UA to guarantee the synchronization. For example, if two sessions are requesting RTP ports from a Port Manager, it is important for the Port Manager to use a synchronization method, for example, the method "*mutex*", to make sure that it assigns different ports to the two sessions.

4. EVENT NOTIFICATION IN WSIP

On WSIP terminal, the application can access the SIP UA services through a SOAP client, which sends service request through the web service channel. For example, the application can make a SIP call by sending an appropriate SOAP request through the SIP client according to the WSDL specification of the WSIP terminal.

In order for application to receive the SIP UA events, WSIP terminal adopts the event notification mechanism proposed in Web Service-Notifications [12]. WSIP terminal, the event producer, provides a SUBSCRIBE service. This allows the event consumer, SIP application, to register the event and the event handler that it is interested in. Such information can be stored in the event registration table. In order for the SIP UA to send the event to it, the SIP application first exposes the event handling function NOTIFY as a web service, then subscribes the events to SIP UA by sending a SOAP message through the SOAP client. Inside the SOAP message it contains the event it is interested in, and the corresponding event handler (WSDL).

When a subscribed event happens, the WSIP terminal looks up the registration table, finds the subscribers, and then sends the event to the subscribers' web services using its SOAP client. Inside the SIP application, a message queue is used to store the event temporarily. When the SOAP server receives an event, it simply posts the event to the message queue. Another thread keeps retrieving the message from the queue, and processing all the events asynchronously.

To illustrate the operation of WSIP terminal between web services and SIP signaling, an example call flow is described in Figure 2. It consists of the following steps.

0. A SIP application subscribes the events CallArrival, CallHangup, CallAnswered, etc., to WSIP terminal.
1. When a remote SIP terminal calls the SIP UA, a SIP INVITE message is received.
2. SIP UA notifies the WLC about the event, which wraps the event and sends it to the SIP application.
3. After receiving the CallArrival event, the SIP application may pick up the call by sending AnswerCall message to WSIP terminal.
4. SIP UA sends 200 OK to the remote end, and the call is set up.
5. SIP UA sends CallAnswered event to SIP application.
6. When SIP application wants to end the call, it sends HangupCall message to WSIP terminal.
7. SIP UA sends BYE to the remote to disconnect the call.
8. WSIP terminal then sends CallDisconnected event to the SIP application.

5. INTEGRATION WITH *CSTA* XML AND XML BASED APPLICATIONS

The WSIP in our approach is designed for easy integration with XML based applications. The incorporation of SOAP/WSDL web service in WSIP provides the standard SOAP message method to package and transport XML based remote procedure calls and application data from the application. One of important advances in XML technology is the emergence of XML based switching and call control environment.

CSTA (Computer Supported Telecommunications Applications) from ECMA provides a standardized abstraction layer for applications to perform call and device control in a business telecommunication environment. ECMA-348 (CSTA-WSDL) [6] provides a web services description language specification for functions in CSTA. An example of CSTA WSDL application is listed in Table 1.

```
<?xml version="1.0" encoding="UTF-8"
   standalone="no" ?>
<SOAP-ENV:Envelope
      xmlns:SOAPSDK1="http://www.w3.org/2001/XM
      LSchema"
      xmlns:SOAPSDK2="http://www.w3.org/2001/XM
      LSchema-instance"
      xmlns:SOAPSDK3="http://schemas.xmlsoap.org/
      soap/encoding/" xmlns:SOAP-
      ENV="http://schemas.xmlsoap.org/soap/envelop
      e/">
   <SOAP-ENV:Body SOAP-
         ENV:encodingStyle="http://schemas.xmls
         oap.org/soap/encoding/">
      <SOAPSDK4: AnswerCall
            xmlns:SOAPSDK4="
            http://www.ecma.ch/standards/ecm
            a-323/csta/ed2">
         <callToBeAnswered>
               <callID>1878</callID>
               <deviceID>135.10.52.22</deviceID>
         </callToBeAnswered>
      </SOAPSDK4: AnswerCall >
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Table 1 An example of CSTA SOAP message

Our approach of WSIP is native to the call and device control environment specified by CSTA WSDL, because WSIP combines the web service with SIP. It allows a WSIP terminal be easily integrated with a CSTA application, either as a server or as a client in web service

framework. The infrastructure of WSIP is agnostic to any particular XML application dialects, and should be interoperable with other XML based application with ease.

6. SUMMARY

In this paper, we presented the approach and implementation of WSIP (Web Service SIP) for converged communication services. In our approach, each WSIP node is both a SIP endpoint that communicates in the SIP world through SIP signaling, and a web service (SOAP) node that provides a native and generic service integration environment for binding SIP based communication in web service based application. The clear separation of service integration environment and SIP signaling in our approach maintains the simplicity and efficiency of the existing SIP protocol, while leverages the power of both methods for converged communication services. The ubiquitous service integration nature of web service provides a disciplined solution to maintain and update large number of SIP endpoints from different installation bases. It enables dynamic monitoring, repairing, and updating SIP UA endpoints, which is essential to the success of VoIP using SIP. The proposed approach was implemented in a prototype research system. Technical issues in the implementation were studied.

7. REFERENCES

[1] SIP: Session Initiation Protocol, http://www.ietf.org/rfc/rfc3261.txt?number=3261

[2] SOAP/WSDL, http://www.w3.org/2002/ws/

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

[4] Standard ECMA-269, Services for Computer Supported Telecommunications Applications (CSTA) Phase III. http://www.ecma-international.org/publications/ standards/Ecma-269.htm

[5] Standard ECMA-323, XML Protocol for Computer Supported Telecommunications Applications (CSTA) Phase III. http://www.ecma-international.org/ publications/standards/Ecma-323.htm

[6] Standard ECMA-348, Web Service Description Language for CSTA Phase III. http://www.ecma-international.org/ publications/standards/Ecma-348.htm

[7] SOAP Version 1.2 Part 0: Primer. http://www.w3.org/TR/2003/REC-soap12-part0-20030624/

[8] SOAP Version 1.2 Part 1: Messaging Framework. http://www.w3.org/TR/2003/REC-soap12-part1-20030624/

[9] Session Initiation Protocol (SIP) Extension for Instant Messaging. IETF RFC 3248. http://www.ietf.org/rfc/rfc3428.txt?number=3428

[10] Voice Browser Call Control: CCXML Version 1.0. W3C Working Draft. http://www.w3.org/TR/ccxml/

[11] EMMA: Extensible Multimodal Annotation Markup Language. W3C Working Draft. http://www.w3.org/TR/emma/

[12] Web Service Notification (WS-Notification). Version 1.0. http://www-106.ibm.com/developerworks/library/ws-resource/ws-notification.pdf

[13] Microsoft SOAP Toolkit 3.0 User Guide.

[14] F. Liu, A. Saad, L. Li, and W. Chou, “A distributed multimodal dialogue system based on dialogue system and web convergence”, ICSLP 2002.

[15] Voice Extensible Markup Language (VoiceXML) Version 2.0. W3C Proposed Recommendation. http://www.w3.org/TR/2004/PR-voicexml20-20040203/

[16] L. Dang, C. Jennings and D. Kelly, “Practical VoIP Using VOCAL”. O’REILLY 2002.

[17] D. Collins, “Carrier Grade Voice Over IP”. MCGRAW-HILL PROFESSIONAL TELECOM 2001.

[18] M. Schwartz, “Telecommunication Networks, Protocols, Modeling and Analysis”. Addison-Wesley Publishing Company 1987.

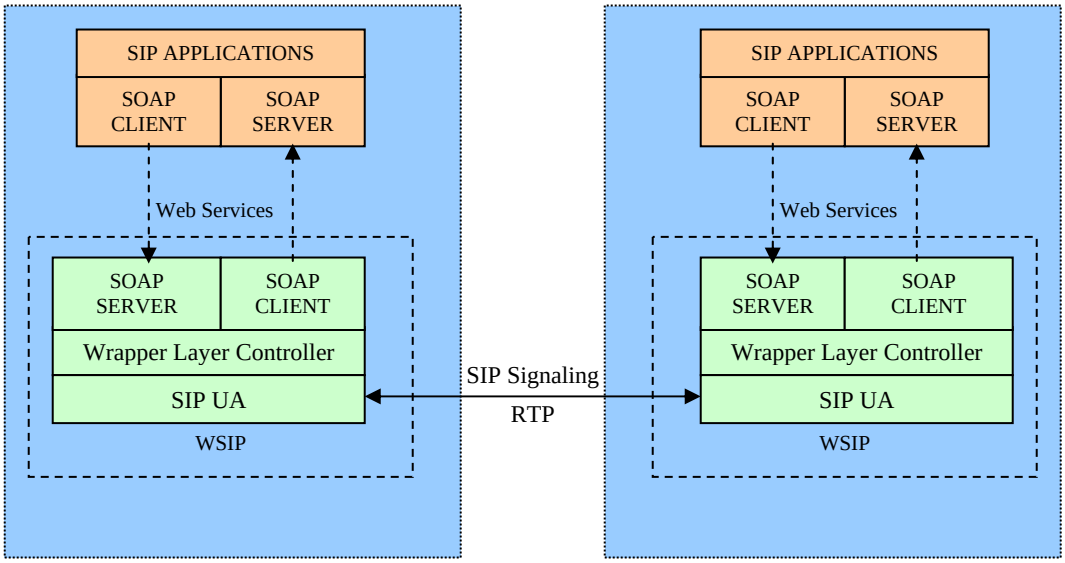

Figure 1, The architecture of WSIP

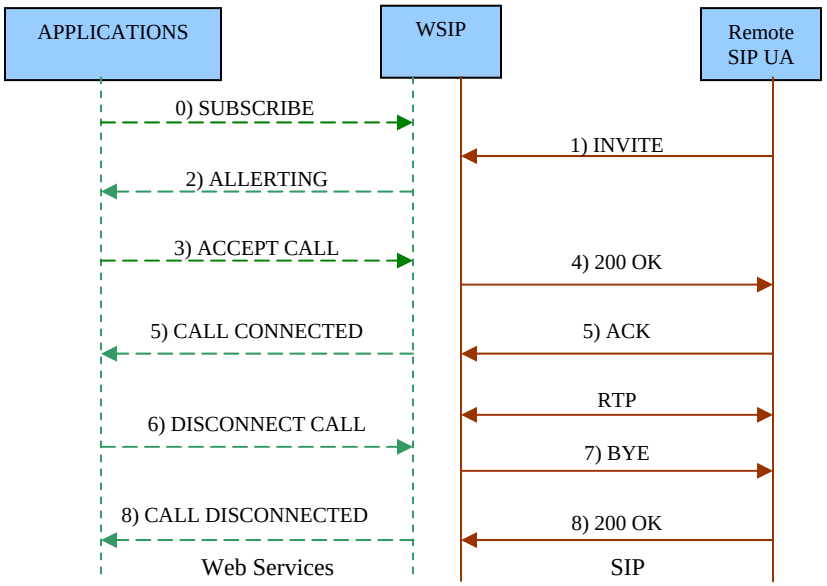

Figure 2 An example of call flow