Web Services Methods for Communication over IP

Wu Chou, Li Li , Feng Liu
Avaya Labs Research, Avaya Inc.
233 Mount Airy Road, Basking Ridge, New Jersey, USA
{wuchou, lli5, fliu1}@avaya.com

Abstract

In this paper, we study web services methods and approaches to enable real-time communication services over IP. This approach extends web services methodologies from service integration to a service-oriented approach for communication. In particular, we describe the generic web service based application session management, WS-Session, the two-way full duplex web service interaction framework, and most importantly, the development of Web Service Initiation Protocol (WIP) which is a full featured web service and SOA based communication framework for multimedia and voice communication over IP. We show that WIP provides a service oriented architecture which can be extended seamlessly from a P2P endpoint to an endpoint with advanced call control and switching capabilities typically requiring the assistance of a dedicated PBX. A prototype of WIP system is fully developed. Architectural design and system implementation of web service based communication endpoints are studied and applied to realize service-oriented communication over IP. Advances of WIP indicate the beginning of a full web service and SOA based communication paradigm that can reshape the converged communication over IP.

1. Introduction

In the past, communication was a tightly coupled nutshell wrapped by many layers of low level protocols. It was difficult to apply and integrate communication in business transactions and to support new applications beyond the basic functions of a standalone device, e.g. telephone. In the era of Internet and collaborative computing, integrated and converged communication is a fast moving research area, and many progresses have been made to advance communication from a predefined or a confined application space to a more open, interoperable, and extensible solution space.

Service-oriented communication (SOC) is to enable communication through a service-oriented architecture and to make communication as service. It receives increasing attention in the industry [1][2][6][14][19][20]. In SOC, an extensible service abstraction layer is introduced to separate communication services from physical implementations and to make communication service-oriented. The service-oriented approach of SOC can be applied to various levels of communication and well beyond the scope of the service abstraction layer, e.g. at the communication protocol level and message exchange patterns.

Web service is a major disruptive technology for SOA enablement, and SOC uses web services to enable communication. Currently, web service receives a profound attention and adoption by the industry and standard bodies [3][10][11][12][14][16][17][18][20]. As a fundamental methodology in SOA, web service provides a way to realize loosely-coupled service-oriented architecture and interoperable solutions across heterogeneous platforms and systems. By enabling existing enterprise resources through web services, these enterprises can be expanded to provide services to a wider variety of clients.

Recent advances in the field of web services have made it practically possible to provide communication services through the web service methods and to incorporate and integrate web services for communication purposes [11][14][16][17][18][19]. One important development in the area of web services for communication is the effort at ECMA for Computer Supported Telecommunication Applications (CSTA) [13]. ECMA-348, Web Services Description Language for CSTA Phase III [14], provides the web service WSDL specification of CSTA services. It specifies a service abstraction layer for telecommunication applications, which is independent of underlying signaling protocols (e.g. H.323, SIP, T1, etc.) and independent of devices (e.g. intelligent endpoints, low-level function/stimulus devices, etc.).

In addition, the approach of WSIP (Web Service SIP) [6] describes a communication paradigm over IP that combines web services with SIP for VoIP communication. WSIP is a dual-stack (Web Service+SIP) solution that uses web service to separate service integration from signaling and media transmission. It utilizes an application layer protocol overlay, e.g. HTTP and SIP, for reliable communication and service extensions. However, the dual-stack approach of WSIP is not fully web service and SOA based and SIP is part of its

dual-stack signaling protocol. Consequently, it needs to implement SIP and deal with many SIP related legacy issues in WSIP.

This paper is focused on our recent advance of WIP (Web Service Initiation Protocol) [1]. The approach of WIP extends the web service enabled service-oriented communication from the service integration level, e.g. WSIP, to the communication protocol level, e.g. WIP, to realize a full web service and SOA based communication framework.

However, the use of web services in service-oriented communication leads to various technical challenges requiring new technical advances and solutions. For example, conventional web service is based on one-way request/response interaction pattern between a client and a server, and the service operations are typically stateless. However, communication is intrinsically two-way full duplex interaction. Services in communication are typically stateful services. It requires the establishment of "session" to identify the client on the server, in order to maintain the service context, and to transmit and exchange events.

The organization of this paper is as follows. In Section 2, we introduce the related work. Section 3 describes a generic web service method, WS-Session, for communication session establishment. This method is fully web service based and agnostic to transport layer protocols. In Section 4, we introduce the framework of two-way web services and we discuss the interface design for two-way full duplex web service interaction. In Section 5, we introduce WIP (Web Service Initiation Protocol) and its operation principles, which is a full web service and SOA based communication paradigm for multimedia and voice communication over IP. In Section 6, we study the extensibility and protocol interaction of WIP. Section 7 describes the WIP endpoint implementation, the experimental results and WIP extension for advanced call control and switching capabilities of CSTA. Finally, the paper is summarized in Section 8.

2. Related Work

Converged communication over IP is an active research area due to the high demand and its disruptive nature. SIP (Session Initiation Protocol) [7] of IETF is a strong candidate in today's multimedia and voice communication over IP. However, SIP is neither XML nor web service based. Many efforts were made to enhance its capabilities and one approach is to embed XML based service control/response in the SDP (session description protocol) part of SIP INVITE message. However, SDP does not provide a full-range of control/response negotiation capabilities. It is based on a descriptive syntax with very rigid format. The receiving SIP User Agent (UA) can ignore the INVITE SDP and respond with its own SDP in SIP call setup.

XMPP (Extensible Messaging and Presence Protocol) [8] is another IETF standard for communication over IP. It specifies a protocol for streaming XML elements in order to exchange structured information in near real time response between two network endpoints. While XMPP provides a generalized, extensible framework for exchanging XML data, it is mainly used for the purpose of instant messaging and presence applications. XMPP is quite different from SIP, and it is a XML based protocol, allowing the use of XML schema and name space for Internet based processing.

Jingle Signaling [9] is an on-going JEP standard development effort based on XMPP, because there is no widely-adopted standard for initiating and managing peer-to-peer (P2P) multimedia interactions, such as voice and video, within Jabber/XMPP clients. Jingle Signaling departs from the early dual-stack solutions of XMPP+SIP, because it may not be feasible on all computing platforms that XMPP clients have been written, nor desirable even it is feasible. Jingle Signaling is aimed at a full XML protocol by standardizing extensions of XMPP and it is used in Google Talk. The emergence of XML based Jingle Signaling is a clear indication for a more Internet and web centric protocol in communication over IP. However, Jingle Signaling is not based on the web service and SOA paradigm, and it relies on the streaming XML element model of XMPP. The call control and signaling model of Jingle Signaling still remain to be further developed.

WSIP (Web Service SIP) described in [6] is a dual-stack Web Service+SIP approach for multimedia and voice communication over IP. A WSIP node is both a web service SOAP node and a VoIP SIP node. This dual-stack approach provides an application layer protocol overlay for communication over IP that can leverage multiple protocols of SIP and HTTP. However, it may not always be feasible to have a SIP stack on a web service platform, and even feasible, maintaining two stacks of communication protocols may not be desirable.

To address the abovementioned issues, the Web Service Initiation Protocol (WIP) proposed in [1] provides a full featured web service based communication framework using a single-stack of web services to enable real-time and service-oriented communication over IP.

3. Session Management

Session association is perhaps one of the most important relations in communication. This is because the

interactions in communication are typically stateful and the stateful transactions are usually based on a stateful context "Session". In the situation of communication over TCP, a client needs to establish a TCP connection with the server before any message exchange starts. This connection (TCP/IP port) serves the purpose of identifying clients on the server as well as transmitting the events. The lifecycle of a session in this case is managed by the TCP/IP protocol.

```
<portType name="ApplicationSessionServicesPortType">
 <operation name="tns:StartApplicationSession">
 <input message="tns:startApplicationSession"/>
<output message="tns:startApplicationSessionPosResponse"/>
<fault name="StartFault"
message="tns:startApplicationSessionNegResponse"/>
</operation>
<operation name="tns:StopApplicationSession">
<input message="tns:stopApplicationSession"/>
<output message="tns:stopApplicationSessionPosResponse"/>
<fault name="StopFault"
message="tns:stopApplicationSessionNegResponse"/>
</operation>
<operation name="tns:ResetApplicationSessionTimer">
<input message="tns:resetApplicationSessionTimer"/>
<output
message="tns:resetApplicationSessionTimerPosResponse"/>
<fault name="ResetFault"
message="tns:resetApplicationSessionTimerNegResponse"/>
</operation>
<operation name="tns:ApplicationSessionTerminated">
<output message="tns:applicationSessionTerminated"/>
</operation>
</portType>
```

Figure 1 Web service operations of WS-Session

However, web service is intended to be transport protocol neutral and it cannot rely on the lower-level transport protocol to manage the session. In order to use web services to enable communication, it is needed to incorporate certain application session based web services in service-oriented communication to manage the session. WS-Session is a generic web service method for application session establishment. This approach is adopted as an ECMA standard, ECMA-366 [16], as well as an ISO standard ISO/IEC 25437.

In WS-Session, a separate portType is created that contains generic session management operations. These operations are defined to take advantage of the ECMA-354 XML messages [15]. WS-Session provides an explicit and declarative specification of a set of web service operations that are needed to establish, extend and terminate an application session. It separates session management semantics from other operations, for example, event subscription. As a result, it enforces the rule that a client and a server can exchange messages if and only if a valid application context (session) is established. In terms of architecture design, WS-Session offers flexibility for the server to perform context-related tasks in service invocation. Figure 1 lists the web service operations in WS-Session WSDL [16].

WS-Session allows explicit session establishment through web services. For example, when combined with ECMA-348, WS-Session allows a client to follow ECMA-269 [13] explicit association establishment sequence to create a session on the server through web services as an alternative to other non-web service methods [19].

Following WS-Session, a successful web service based session establishment will result in a unique sessionID for the client. This sessionID will be included in the SOAP header of the subsequent SOAP messages within the session. More examples on the use of WS-Session will be provided later in the discussion of WIP. Application session establishment of WS-Session is entirely web service based and specified through the standard based web service specifications (SOAP/WSDL). The web service operations of WS-Session are independent of the underlying transport protocol (e.g. JMS, HTTP, TCP, MQ, SMTP, etc.).

4. Two-Way Web Service Integration

In two-way web service interaction, each web service endpoint is both a client and a server, and it requires two appropriate WSDLs, one at each endpoint to conduct two-way web service interaction. These two WSDL interfaces need to be correlated in a certain way to enable the desired full duplex web service interaction. This motivated the research efforts on two-way web services. In particular, the two-way web service framework and its WSDL interface design were studied in [4], and applications of two-way web services in communication were described in [1][2] [19].

From an endpoint perspective, two-way full duplex web service interaction involves both client initiated service request and proactive server push for event notification. From service interaction perspective, we consider three types of generic web service interaction patterns. We use the subscript to indicate the service contact initiator, and use a generic client and server to separate two web service endpoints, although each of them is both a client and a server.

- Type I (RC): Requests initiated by the client, with or without response.
- Type II (ES): Event reports from the server, with acknowledgment (solicitation) or without

(notification).

- Type III (RS): Request initiated by the server, with or without response.

When Type II outbound (asynchronous) operations are needed, client has to specify the event sink for server to send event notification when the subscribed service event happens. A theoretical framework to systematically design and verify client interface in two-way web service interaction is studied in [4] [19], where properties and use cases of tightly coupled (TC) interface and loosely coupled (LC) interface are described therein.

4.1 WS-Addressing and WS-Eventing

WS-Addressing is an important web service standard for one-way service addressing [11]. It specifies an extensible endpoint reference (EPR) structure that can address services on the web service endpoint with an enhanced resolution that is beyond the traditional URI or IP address. This property makes WS-Addressing well suited for web service based communication over IP, because today's Internet is inherently peer-to-peer and based on IPv4 with limited IPv4 addresses. From web service communication perspective, WS-Addressing specifies a message level service resource extension which does not require the physical migration to IPv6 addresses. Because of the extensibility of the EPR in WS-Addressing, web service endpoints and their services can be uniquely identified through WS-Addressing EPR mechanism, and it alleviates the physical address resource limitations of IPv4. It makes web service suitable for communication over IP.

Moreover, event notification is critical for communication services, and WS-Eventing [17] specifies a light weight web service eventing mechanism based on WS-Addressing. It introduces the following web service constructs:

- Event Source: A web service that generates events and accepts event subscriptions.
- Event Sink: A web service that receives event notification messages.
- Subscriber: A web service that subscribes to event sources on behalf of the event sink, although they are usually identical.
- Subscription Manager: A web service that manages the subscriptions, such as renewal, termination, pause and resume; and it can be the same web service as Event Source.

WS-Eventing is a generic web service protocol to establish the relation between event source and sink. It provides web service based event notification mechanism that can be used to enable communication over IP.

5. Web Services Initiation Protocol (WIP)

WIP is based on a single stack of web services to enable service-oriented communication over IP. It uses two-way web service interactions to establish communication signaling for multimedia and voice communication.

5.1 Communication Flow of WIP

Figure 5 illustrates the call flow and operations of WIP using WS-Session, WS-Eventing and CSTA web service operations to establish P2P communication over IP. As illustrated in Figure 5, WIP is a fully web services based communication paradigm and each WIP endpoint is a standard (SOAP/WSDL) based web service endpoint well suited for existing and future web services platforms.

5.2 WIP Session Establishment

Figure 2 is a sample template of web service session establishment SOAP message in WIP. WIP is based on WS-Addressing for one-way service addressing for communication over IP, and it uses WS-Addressing <wsa:ReferenceParameters> to address the service endpoints. The session establishment in WIP is based on WS-Session and it uses <StartApplicationSession> operation in Figure 1 to establish an application session with the server. A sessionID element is returned by the server in the positive response message. The sessionID element is used in the subsequent service interaction to address the session as long as the session exists.

```
<s11:Envelope><s11:Header>…</s11:Header>
<s11:Body>
<StartApplicationSession xmlns="http://www.ecma-
international.org/standards/ecma-354/appl_session" >
<applicationInfo>
  <applicationID>${wip_ua}</applicationID>
  <applicationSpecificInfo>
    <wip:call
xmlns:wip='http://research.avayalabs.com/dialog/wip'>
    <wip:subject>${subject}</wip:subject>
    <wip:sender name="${sendername}" url="${senderurl}" />
    <wip:receiver name="${receivername}" url="${receiverurl}" />
    <wip:priority>${priority}</wip:priority>
    </wip:call>
  </applicationSpecificInfo>
 </applicationInfo>
<requestedProtocolVersions>
 <protocolVersion>http://www.ecma-
international.org/standards/ecma-323/csta/ed3
 </protocolVersion>
</requestedProtocolVersions>
</StartApplicationSession>
</s11:Body>
</s11:Envelope>
```

Figure 2 StartApplicationSession request message

In WIP, an application session can be terminated by

session duration timer or by stopApplicationSession operation. If the established session terminates, all subscribed services within the session will cease to exist. Figure 3 illustrates a positive response SOAP message from the server for a successful application session establishment, in which an element sessionID is returned by the server for other services to address the session. A service requester can join the session and use the session context by including the sessionID in its service request SOAP message header section [16]. From web service perspective, sessionID in WS-Session has a well defined semantics in WS-Addressing framework as a special application dependent endpoint reference parameter. This relationship is an important feature in WIP.

5.3 Media Negotiation Signaling in WIP

The media negotiation signaling in WIP is based on two-way web service interaction message exchanges. A key idea in WIP is to model media transmission relationship as a special "event" subscription in web services. The receiving port is modeled by an "event sink", whose IP endpoint is specified by its event sink EPR (endpoint reference) of WS-Addressing. Moreover, the source of the media is modeled by an "event source", whose IP endpoint is specified by its event source EPR. Therefore, the media negotiation in WIP is modeled by finding the matching event (media) sink-source between two WIP endpoints for a selected media type which itself is also modeled as part of the EPR. A successful event subscription will allow the source to send (transmit) the "event" (media) to the matching sink, and consequently, it establishes the media communication over IP.

```
<s11:Envelope><s11:Header>...</s11:Header>
<s11:Body>
<StartApplicationSessionPosResponse
xmlns="http://www.ecma-international.org/standards/ecma-
354/appl_session" >
        <sessionID>${session_id}</sessionID>
        <actualProtocolVersion>http://www.ecma-
international.org/standards/ecma-
323/csta/ed3</actualProtocolVersion>
        <actualSessionDuration>-1</actualSessionDuration>
</StartApplicationSessionPosResponse>
</s11:Body>
</s11:Envelope>
```

Figure 3 StartApplicationSession response message

The approach of WIP is full web service based according to a novel use of WS-Eventing for media negotiation and transmission. Figure 6 and Figure 7 illustrate one pair of SOAP message templates for such web service interaction. In WIP, the session and media negotiation can be completed with only three web service interactions [1]. It should be noted, although the generic web service event based media negotiation signaling framework of WIP is described using WS-Eventing, it can be realized with other web service methods, such as WS-Notification. The use of WS-Eventing here is to make WIP approach consistent with the roadmap towards converging web service standards for resources, events, and management [21].

5.4 WIP: A Stackable SOA Based Protocol

The communication service enablement of WIP follows the web service based SOA paradigm. It is based on web service (SOAP/WSDL) message exchanges to invoke services. This makes WIP endpoints interoperable with each other, and also interoperable with other web service based SOA platforms. WIP provides a consistent SOA based extensibility framework by enabling communication as services. It utilizes WS-Session to bundle services, including media, channels, controls, etc., and utilizes a generic web service method based on WS-Eventing for media negotiation. This approach in WIP is not tied to any pre-defined applications or services.

Moreover, services at WIP endpoints are stackable, and services can be added, deleted and provisioned dynamically in a self-describing fashion without changing the WIP structure.

6. Extensibility and Protocol Interaction in WIP System Realization

It is clear from the operations of WIP that there are actually two levels of web services: the base services that provide specific functions in a particular domain, and meta-services that provide services or enforce rules for the base services. Without a base service, a meta-service has no effect on real world. For example, the sessions generated by WS-Session service must be used by some base services. On the other hand, meta-services are independent of base services and they can be combined with different or multiple base services.

In addition to WS-Session, WIP specifies a generic media negotiation web service interaction pattern based on WS-Eventing. This generic media negotiation interaction pattern is not tightly coupled with any particular media, and it is applicable to any media type following the event source-sink modeling used in WIP. This mechanism is extensible. A new media source will be treated as a new event source whose media type and related media information will be specified following the WS-Eventing EPR extension mechanism.

6.1 Protocol Interactions

To realize these relationships at runtime, we must allow services to interact with each other at two levels: resource and message.

At resource level, a service needs to access the resources created by another service or create certain relationship between resources across services. In this case, the service can be modeled as a "factory" that manages the lifecycle and persistence of its resources through well-defined APIs.

TABLE 1
TIME MEASUREMENTS IN WIP CALL SETUP

	Client (round trip) (ms)	Server (processing time) (ms)
Start Application Session	44.67 (16.45)	< 10
Subscribe to Session	45.67 (10.07)	< 10
CSTA Monitor Start	30 (7.07)	< 10
Subscribe Offer	59 (23.64)	5.56 (16.67)
Subscribe Confirm	85 (29.03)	7.78 (8.33)
Total	256.11 (86.27)	13.34 (25)

At the message level, a service needs to access or change the messages as well as call flows of another service. For example, the ECMA-348 service must analyze WS-Eventing subscribe message to determine if a session is the target of the subscription. If the session is valid, the link between the subscription and session must be established. Otherwise, it must force a fault message. To permit this kind of interaction, the service should separate its core logic from extensions so that the extension can be manipulated while the core is protected and can be invoked from different context. For extension, we found the Interceptor design pattern is a most flexible choice as it leaves the client, service and the interceptors loosely coupled and fits well with current SOAP framework. We call this kind of interceptor, integration interceptor, as it is used to integrate various services in WIP implementation.

7. WIP Endpoint Implementation

Table 1 lists the averaged time of each web service interaction and the total time spent during call setup for P2P communication using WIP. It is averaged for multiple test runs and the number in the parentheses is the standard deviation of the measurements. The results indicated that WIP is quite efficient. The average call setup time was in the range of 256 ms which is well within the requirement of call setup for real-time communication in existing communication protocols. The server processing time for received web service SOAP messages is around 10 ms because the message payload in WIP is quite small in a P2P application using voice and video as illustrated by the sample messages in Figure 6 and Figure 7 of WIP.

7.1 WIP Endpoint Service Extension

In the framework of WIP, new applications can be added to the base service stack of WIP endpoint as services in a service oriented manner. These new services are not limited to media services and they can include advanced call control services, presence services, monitoring services, event services, etc.

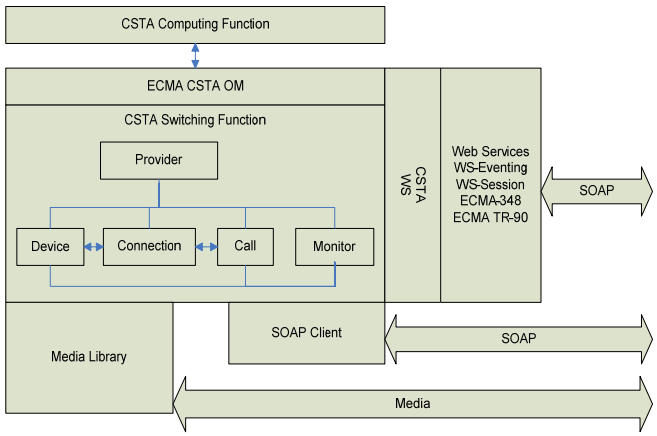

Figure 4 Architecture diagram of an extended WIP endpoint

To verify the service-oriented architecture of WIP, we extended our basic WIP endpoint from P2P real-time communication with voice and video to include CSTA call control and switching capabilities, e.g. single-step-transfer, monitor, etc. and CSTA call control events, e.g. transferred-event, offered-event, etc. We implemented the CSTA object model for these functionalities, exposed them as web services following ECMA-348, and then added them as services to the base service stack of WIP endpoint. These new services were incorporated into WIP following the protocol interaction patterns of meta-service

and base-services as described in Section 6 and ECMA TR-90 [19]. Figure 4 illustrates the high level architecture embodiment of the extended WIP endpoint in our implementation.

The new call control services, e.g. the added ECMA CSTA call control web services, can be invoked and inter-working with the existing services. Our basic WIP endpoint supports voice and multimedia communication over IP through web services without resorting to any non-web service methods. All media services, e.g. voice, video, chat, etc., can be subscribed and managed following the web service based media subscription framework described in Section 5.3.

The new services that we added to the base service stack of the WIP endpoint can enable the WIP endpoint to perform advanced switching and call control functions. This extension mechanism of WIP is service-oriented without changing the WIP structure. The extended WIP endpoints were demonstrated in an IP call center application where a multimedia WIP call with voice and video were transferred from one agent to another utilizing the presence aware services on the WIP endpoint and without a loss of any media.

8. Summary

In this paper we presented communication over IP based on web services. We described the generic web service based application session establishment, WS-Session; the two-way web service framework for web service and SOA based communication enablement; the full featured web service based communication protocol, WIP (Web Service Initiation Protocol) and its generic web service based media negotiation signaling framework. Testing results presented in this paper indicated that WIP for real-time voice and multimedia communication over IP is not only feasible but advantageous. WIP is a full-featured web service based communication protocol for service-oriented communication. It can be extended to support advanced call control features for sophisticated enterprise communication services. The full web service nature of WIP makes it well suited for the infrastructure of the Internet and interoperable with existing and future web service platforms.

REFERENCES

[1] W. Chou, L. Li and F. Liu, "WIP: Web Service Initiation Protocol for Multimedia and Voice Communication over IP", pp. 515-552, Proceedings of IEEE International Conferenc on Web Services (ICWS'06), Sept. 2006.

[2] W. Chou, L. Li and F. Liu, "Web Service Enablement of Communication Services", Proceedings of IEE International Conference on Web Services (ICWS'05), vol. 2, pp. 393-400.

[3] M. Haines, "Web service as Information Systems Innovation: A Theoretical Framework for Web Service Technology Adoption", Proc. of ICWS'2004, pp. 11-16

[4] L. Li, and W. Chou, "Two Way Web Service: From Interface Design to Interface Verification", Proceedings of IEEE International Conference on Web Services (ICWS'05), vol. 2, pp. 525-532, July 2005

[5] L. Li, W. Chou, F. Liu and D. Zhuo, "Semantic Modeling and Design Patterns for Asynchronous Events in Web Service Interaction", pp. 223-230, Proceedings of IEEE International Conference on Web Services (ICWS'06), Sept. 2006

[6] F. Liu, W. Chou, L. Li and J. Li, "WSIP - Web Service SIP Endpoint for Converged Multimedia/Multimodal Communication over IP", Proceedings of ICWS'04, pp. 690-697, July, 2004

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

[8] Extensible Messaging and Presence Protocol (XMPP) Core, IETF, Oct. 2004.

[9] JEP-0166: Jingle Signaling, Dec. 15, 2005

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

[11] Web-Addressing 1.0, W3C recommendation, June, 2006, http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/

[12] WS-Security: SOAP Message Security 1.0 (WS-Security 2004) – OASIS Standard 200401, March 2004

[13] ECMA-269: Services for Computer Supported Telecommunications Applications (CSTA) Phase III, 6th edition (June 2004), http://www.ecma-international.org/publications/standards/Ecma-269.htm

[14] ECMA-348, Web Services Description Language (WSDL) for CSTA Phase III, 2nd Edition (June 2004): http://www.ecma-international.org/publications/standards/Ecma-348.htm

[15] ECMA-354, Application Session Services (june 2004)

[16] WS-Session, ECMA-366, http://www.ecma-internal.org/publications/standards/Ecma-323.htm

[17] Web Service Eventing (WS-Eventing), March 2006, http://www.w3.org/Submission/WS-Eventing/

[18] WS-BPEL 2.0, OASIS Web Services Business Process Execution Language, Sept. 2006

[19] TR-90, "Session Management, Event Notification, and Computing Function Services – an amendment to ECMA-348", Dec. 2005, ECMA International.

[20] Open Access, Parlay X Web Services, ETSI ES 202 3912-12 (2005-03) Multimedia Conferencing

[21] "Toward Converging Web Service Standards for Resources, Events, and Management", a joint white paper from HP, IBM, Intel and Microsoft, March, 2006.

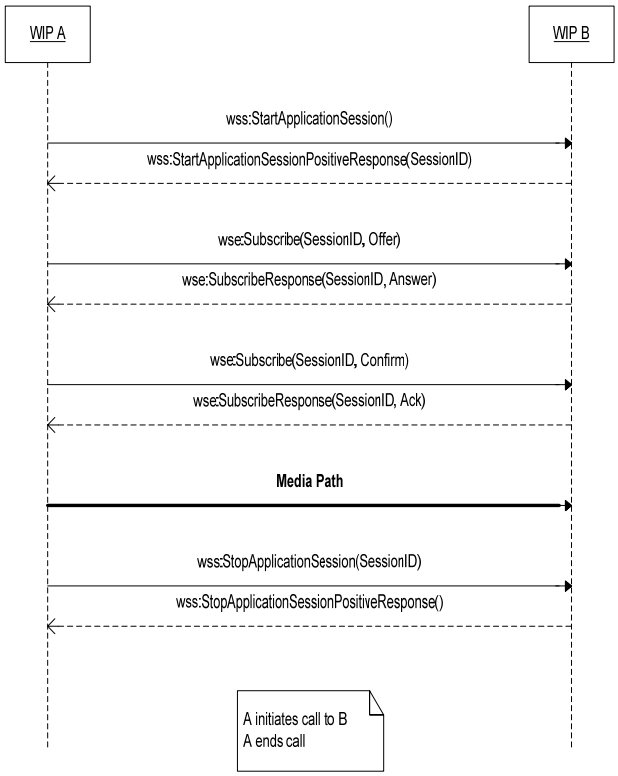

Figure 5 Diagram of call flow between two WIP endpoints in communication over IP

```
<s11:Header>
   …
   <wsa:To>${csta_service}</wsa:To>
   <aps:sessionID>${session_id}</aps:sessionID>
   <csta:deviceID>${callee_device}</csta:deviceID>
</s11:Header>
<s11:Body>
<wse:Subscribe>
   <wse:EndTo>
      <wsa:Address>${sink_URL}</wsa:Address>
   </wse:EndTo>
   <wse:Delivery>
      <wse:NotifyTo>
         <wsa:Address>
            ${my_service_URL}
         </wsa:Address>
         <wsa:ReferenceParameters>
            <csta:deviceID mediaClass="audio
video">${caller_device}</csta:deviceID>
            <wip:transport
media="audio">${audio_transport_a}</wip:transport>
            <wip:transport
media="video">${video_transport_b}</wip:transport>
         </wsa:ReferenceParameters>
      </wse:NotifyTo>
   </wse:Delivery>
</wse:Subscribe>
</s11:Body>
</s11:Envelope>
```

Figure 6 Media subscribe-offer template in WIP

```
<s11:Envelope …>
<s11:Header>
   …
   <wsa:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/a
   nonymous</wsa:To>
<aps:sessionID>${session_id}</aps:sessionID>
<csta:deviceID>${caller_device}</csta:deviceID>
</s11:Header>
<s11:Body>
            <wse:SubscribeResponse>
            <wse:SubscriptionManager>
              <wsa:Address>
                ${SubscriptionManager_url}
              </wsa:Address>
              <wsa:ReferenceParameters>
              <wse:Identifier>${subscription_id}</wse:Identifier>
              </wsa:ReferenceParameters>
            </wse:SubscriptionManager>
      <csta:deviceID>${callee_device}</csta:deviceID>
            <wip:transport
media="${media}">${transport_b}</wip:transport>
            <wip:selection>
            <wip:transport
media="${media}">${transport_a}</wip:transport>
             </wip:selection>
            </wse:SubscribeResponse>
</s11:Body>
</s11:Envelope>
```

Figure 7 Media subscribe-answer template in WIP