Solving Interoperability Problem of SIP, H.323 and Jingle with a Middleware

Rossi Kamal, Muhammad Shoaib Siddiqui, and Choong Seon Hong
Department of Computer Engineering
Kyung Hee University
Gyeonggi Do,Korea
Email: rossi@networking.khu.ac.kr, siddiqui@khu.ac.kr and cshong@khu.ac.kr

***Abstract*—This paper presents a mobile middleware that solves interoperability problems occurred in VoIP protocols (SIP, H.323, Jingle and their different vendor implementations) and uses distributed shared objects to overcome scarce resource problem of mobile devices.**

I. Introduction

VoIP (Voice Over Internet Protocol) is implemented using different protocols like SIP [1], H.323 [2], Jingle [3] etc. There are also variances among each protocol implementations (such as SRTP-MIKEY, ZRTP variances of SIP). Interoperability problems occur when calling party and called party use different protocols.

Existing solutions involve presence of an interim gateway between VoIP implementations.This causes huge overhead to server and keeping a server this way is also a costly solution. But, if we can provide a generic solution in VoIP clients, they can interoperate with each other and also server overhead and cost will be reduced.

Mobile devices are often with scarce resources like minimum memory, limited bandwidth. A mobile middleware based solution can resolve the issue with exploration of distributed shared objects.

Our proposed mobile middleware solves VoIP interoperability problems with its generic framework with support for SIP, H.323 and Jingle and their different implementations. This middleware utilizes distributed shared objects (CORBA[4], RMI[5] and SOAP[6]) to overcome the scarce resources of embedded devices. Here SOAP allows multi-type and multi-stream communication in VoIP signalling and media messages. RMI allows distributed shared object invocation of two VoIP clients of Java platform and CORBA adds platform independency over it. A user agent using this middleware is provided with three options (SIP, H.323, Jingle) and it can choose any option depending on the protocol or the specific implementation of a protocol used at the other end of the session. We have implemented our mobile middleware on JavaME platform with four applications rmobvoip[7], rmoblog[8], mobrmi[9] and mobcorba[10], .

II. Interoperability problems in SIP, H.323 and Jingle

VoIP (Voice Over Internet Protocol) is implemented using different protocols like SIP, H323, Jingle, MCGP etc. There are also variances among each protocol implementations (for example we can say about SRTP-MIKEY, ZRTP variances of SIP). The protocol used by caller has to interoperate with protocol used by the callee. So, if a caller is using SIP but callee is about to be using H323, how they will communicate? Also if a caller is using SRTP-MIKEY and called is using SRTP-ZRTP, how they can recognize each other? The scenario seems quite challenging if we think about broad range of VoIP protocols and their different implementations by vendors around the world.

VoIP clients set a secured session between them in a signaling session and talk to each other in media session. But this session establishment session differs among different protocols and a client implementing one protocol will not recognize other end client who is implementing different protocol. So, they cannot set a secured session on which they can talk to each other in media session.

A. Protocol differences between Jingle and SIP

Jingle, an XMPP extension for IP telephony was designed remembering that it has to work in co-ordination with SIP. Yet, Jingle has some features that yield its interoperability problems with SIP.

1) Message format: SIP uses message formats and header almost like HTTP protocol. XMPP-VOIP uses XML in this case.

2) Payload declaration: SIP uses SDP (Session Description Protocol) in payload declaration stage where Jingle uses XML .But Jingle can map SDP to XML

3) Session Establishment: SIP establishes session in the UDP where Jingle uses TCP .So application layer change or modification does not have any effect.

B. Protocol differences between SIP and H.323

H323 and SIP both have advanced features in a packet voice network. It is a better idea to place them in the same network and internetwork them in the right way. Although they have

good many of functions in common, their implementations are different in both platforms

H323 is a server based IP telephony system on the other hand SIP is a decentralized system in general sense. Data transfer and control are performed in H323 with VOIP gateways SIP clients can talk to each other in media session without the involvement of an interim server. Server first sets a session between them and then clients transfer media data between them.

C. Internetworking between H323 and SIP

Generally, H.323 systems use native key management [H.235.6] or pre-SRTP encryption. But, it also supports SRTP and some SRTP keying scheme like MIKEY [H235.7]. When we need to internetwork between H323 and SIP using SRTP keying mechanism, internetworking device decrypts SRTP (with one key) towards one network and encrypt SRTP (with different key) toward other network. To inter-network between DTLS-SRTP endpoint of SIP and H.323, inter-networking device simply forwards SRTP packet from H.323 network towards DTLS-SRTP key transport layer.

D. Insights on SIP variances: MIKEY, SDES-TLS and ZRTP

Session initiation protocol differs among different implementations based on the way they set secured key between them. Session establishment session can be either MIKEY or SDES-TLS or ZRTP. We have analyzed them in the following

1) MIKEY: Multimedia Internet Keying: MIKEY is used to support key negotiation between two or more peers for SRTP. It can be of three types- Peer to peer (unicast), one to many (multicast), many to many. Three methods are available 1.PSK:Its most efficient way to handle key transport .Yet it does not support group communication 2.PKE:The sender generates a dynamic encryption key and encrypts with receivers public key and sends. This method requires more computation .Its key negotiation for groups and provides scalability where PKI is there 3.Diffie-Helman: This method is most resource-intensive. It provides perfect forward secrecy .But it requires peer to peer communication and existence of PKI

Handshaking is done in signaling channel here MIKEY messages can be exchanged through the signaling protocol that the multimedia application is using. Integrating MIKEY messages within SIP minimizes the number of messages sent between end points to exchange keys. In other words, end points are not required to send separate MIKEY messages and SIP messages to establish a secure session. Therefore, it is desirable to integrate MIKEY messages within the application protocol. When MIKEY uses SIP messages larger than 1300 bytes, we have to use TCP, otherwise UDP. So, when PKI and DH is used, TCP and TLS should be considered .For integrity and confidentiality most of the cases TCP is observed. The MIKEY implementation should use PSK or PKE to respond to other implementation to key management

MIKEY is only limited for peer to peer and simple one to many applications. In the group communication with multimedia support, its performance is not tested better. MIKEY is currently used by minisip.

2) SDES-TLS : Session Description Protocol Security Descriptions for Media -Transport Layer Security: SDES-TLS protocol in based on SDP on background. On SDP offer answer model, caller sends SDP message (set of media streams and codes he wishes to use, ,the IP address and port he uses for receiving message).The answerer generates an answer that is an SDP message. This contains the matching media stream for each offer-indicating whether the accepted or not, along with codec and IP address, port he would use to receive message. We have to be careful about multicasting here. Information is shared between a pair of users but both sides send message to the multicast address rather than unicasting one.

Session Description Protocol Security Description for media streams (SEDS)defines a mechanism to negotiate cryptographic parameters for secured real time protocol(SRTP).A cryptographic attribute may be added to the SDP uni-cast media streams. Using the SDP offer/answer model .the crypto suite to be used can be negotiated as well as other cryptographic parameters (key salts etc) which are necessary for SRTP to secure media stream. Like all SDP messages, SDP message containing security descriptions are conveyed in application protocol SIP. Its the responsibility of the encapsulating protocol to ensure the protection of SDP security descriptions. Therefore, it is required that application uses own security mechanism or invokes lower layer security service as TLS. It is required that this layers security service provides stronger authentication and packet payload encryption, as well as effective replay protection

TLS's primary goal is to provide privacy and data integrity between two communication applications. TLS record protocol provides the connection security in ways. Connection is private. Symmetric keys are used for data encryption (DES,RC4).Keys are generated uniquely for each connection and bases on TLS handshake protocol. Connection is reliable .Secured has function(SHA,md5) are used for MAC computations .This MAC ensures reliable message integrity.TLS record protocol encapsulates the higher level protocols, TLS handshake allows server client to authenticate each other and negotiate encryption algorithm and cryptographic keys before application protocol transmits or receives first bytes of data .TLS handshake starts with peer authentication using RSA or DSS. This is generally optional .But is required for atl east one peer. Negotiation of keys is secured and reliable.

The advantage is that TLS is application protocol independent. Higher layer protocol can layer on top of TLS transparently. How to initialize TLS handshake and how to interpret authentication certificate depend on implementation of upper layer protocols. Most of the sip clients now use SIP over TLS as SJphone, Lynxphone, SNOM, Aastra, PBXnSIP etc. Many of them support SRTP and SDES-TLS as well.

3) ZRTP: The contrast between MIKEY of ZRTP is that cryptography keys are negotiated through media stream (RTP) over the same UDP port. Instead of using signaling channel as its done with MIKEY. Therefore key negotiation is done between pairs without the use of intermediates such as SIP

<table>
<tr><td colspan="4">GUI</td></tr>
<tr><td>SOAP</td><td>RMI</td><td>CORBA</td><td>Extended</td></tr>
<tr><td colspan="2">SIP</td><td colspan="2">H.323</td><td>Jingle</td></tr>
<tr><td colspan="2">DTLS-MIKEY</td><td colspan="2" rowspan="3">Signaling with H.323</td><td rowspan="3">Signaling with Jingle</td></tr>
<tr><td colspan="2">DTLS-SRTP</td></tr>
<tr><td colspan="2">ZRTP</td></tr>
<tr><td colspan="2">Signaling with H.323</td><td colspan="2">Signaling with SIP</td><td>Signaling with SIP</td></tr>
<tr><td colspan="2">Signaling with Jingle</td><td colspan="2">Signaling with Jingle</td><td>Signaling with H.323</td></tr>
<tr><td colspan="5">Media Session</td></tr>
<tr><td colspan="5">Operating System</td></tr>
</table>

Fig. 1. Architechture of mobile middleware

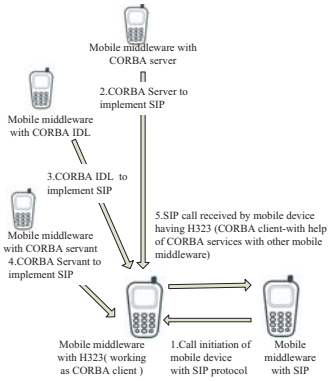

Fig. 2. Mobile middleware with H.323 is making VOIP session with mobile middleware with SIP. In this case, first middleware is working as CORBA client,using three other mobile middleware as CORBA server, servant and IDL

proxies to relay the key material. If necessary ZRTP allows options to exchange keys through signal messages. The protocol uses DH key, it does not require PKI which makes the protocol an alternative to PKI. ZRTP works in two modes. 1. Pre-shared key mode 2.Diffie Hellmann mode. DH is not used at first step, rather pre-shared key is used. The peers use nonce here to calculate the session and then Diffie-Hellman key mode is used.

The initial implementation of ZRTP is Zfone, which interfaces with existing soft phones such as X-Lite, Gizmo, and SJphone, SIPassure has licensed Zfone SDK and integrated ZRTP protocol into them. As a result any SIPassure user can access to the VOIP encryption and fully interoperability with Zfone and other ZRTP compliant client.

III. Proposed Mobile Middleware

A. Architechture

Proposed mobile middleware in Fig. 1 has several layers. The top layer has components for CORBA, RMI and SOAP implementations. There is also an extended component at this layer. It provides the option of integration of new distributed shared object protocol in the future. CORBA component consists of CORBA IDL, CORBA Server, CORBA Servant and CORBA client sub-components. RMI component consists of RMI IDL, RMI Server and RMI Client subcomponents. SOAP component consists of SOAP client and SOAP server sub-components. The next layer consists of components implementing signaling session of VoIP. Our middleware has the support for three protocols SIP, H.323 and Jingle. SIP component on this layer supports signalling with SIP variances:DTLS-MIKEY or DTLS-SRTP or ZRTP, signalling with H.323 and signalling with Jingle. H.323 component supports signalling with H.323, signaling with Jingle and signalling with Jingle. Jingle components similarly supports signaling with Jingle, H.323 and SIP. The next layer has components supporting media session of VoIP session. Two VoIP clients talk in real time in media session based on the signalling session established by upper layer.

B. Functionalities

In Fig. 2, a mobile device with H.323 protocol wants to set a VoIP session with a mobile device with SIP protocol. As two signalling protocol differs, H.323 mobile uses mobile middleware to set session with SIP mobile. It will work as a CORBA client to take help from other three mobile middleware those are working as CORBA IDL, CORBA server and CORBA client to implement SIP protocol for H.323 mobile. Thus, H.323 mobile can set a signalling session with SIP mobile. In the second case, mobile middleware solves the problems for two SIP variances namely DTLS-MIKEY and ZRTP. Mobile middleware with ZRTP works as RMI client to set a VoIP session with DTLS-MIKEY SIP mobile. This mobile middleware takes help of two other mobile middleware working as RMI server and RMI client to implement DTLS-MIKEY on it. Thus ZRTP mobile can set VoIP session with a mobile with DTLS-MIKEY. In the third case, mobile middleware with SIP protocol is communicating with a mobile middleware with SIP. Both middleware are using SOAP component to explore multitype, multistream communication in VoIP session.

C. Implementation

Our mobile middleware is implemented on Java ME platform. Applications namely rmobvoip, rmoblog, mobcorba, mobrmi are developed with J2ME to evaluate the performance of the proposed middleware. Class diagram of the middleware in Fig. 3 describes what the internal operation goes in the middleware. CORBAImpl, RMIImpl and SOAPImpl classes are responsible for distributed shared object communication in the middleware. SipImpl, H323Impl and JingleImpl are responsible for implementing session initiation step of SIP, H.323 and Jingle respectively. MediaImpl class is responsible for implementing media session of three VOIP protocol.Let us see internal operation when a mobile middleware with H323 sets a VoIP session with a SIP mobile device[Fig. 2]. The H.323 mobile creates object of CORBAImpl and implements the CORBAClientImpl() opertion. It invokes three other objects of CORBAIDLImpl and those three objects

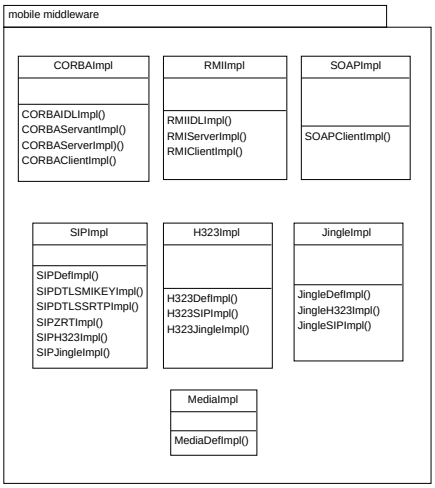

Fig. 3. Mobile middleware implementation

are implementing operations CORBAIDLImpl(), CORBAServantImpl() and CORBAServerImpl(). These three objects invoke H323SipImpl() to implement SIP for the caller mobile middleware with H323. Thus, mobile middleware with H.323 uses three other mobile middleware clients to make a VoIP session with a SIP mobile using CORBA.

Now we can consider the scenerio where two SIP variances want to create a VoIP session. Mobile middleware with ZRTP creates object of RMIImpl and implements RMIClientImpl() operation. It invokes two other objects of RMIImpl and these objects perform operations RMIIDLImpl() and RMIServerImpl(). These three objects invoke SIPDTLSMIKEYImpl() operation of the object SIPImpl. Thus, mobile middleware with ZRTP sets a VoIP session with DTLS-MIKEY mobile with help of two other mobile middleware in RMI communication.

If two mobile middleware are using same protocol, SOAP implementation will allow them multitype and multistream VoIP session. First middleware will create SOAPImpl object and perform SOAPDefImpl() operation. It then creates object of SIPImpl and perform SIPDefImpl() operation to set a VoIP session with other middleware with SIP. After the session is established (any of the three ways), the mobile middleware will invoke MediaImpl object and perform MediaDefImpl() operation to media data transfer.

IV. Conclusion

In this paper, we have proposed a mobile middleware that solves interroperability problems of VoIP protocols namely SIP, H.323 and Jingle. Our middleware uses distributed shared objects with RMI, CORBA and SOAP to overcome the limited energy, memory of mobile devices.

Acknowledgement

This research was supported by Future-based Technology Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (No.20100020728). Dr. CS Hong is the corresponding author.

References

[1] H. Schulzrinne and J. Rosenberg, *The Session Initiation Protocol: Internet-centric signaling*, Communications Magazine, IEEE , vol.38, no.10, pp.134-141, Oct 2000
[2] H. Liu and P. Mouchtaris, *Voice over IP signaling: H.323 and beyond*, Communications Magazine, IEEE , vol.38, no.10, pp.142-148, Oct 2000
[3] XMPP, *XEP-0166:Jingle*, http://xmpp.org/extensions/xep-0166.html.
[4] *OMG:The Object Management Group*,CORBA 3.0 Specification,http://www.omg.org
[5] *T.B. Downing*,Java RMI:The Remote Method Invocation,DG Books Worldwide, ISBN:0764580434.
[6] *w3c*,SOAP (Simple Object Access Protocol) Version 1.2 ,http://www.w3.org/TR/soap12-part1/
[7] *rmobvoip*,http://code.google.com/p/rmobvoip.
[8] *rmoblog*,http://code.google.com/p/rmoblog.
[9] *mobrmi*,http://code.google.com/p/mobrmi.
[10] *mobcorba*,http://code.google.com/p/mobcorba.