A Mobile Middleware to Solve Interoperability Problems in VOIP Streaming Session

Rossi Kamal

Department of Computer Engineering,
Kyung Hee University,
1 Seocheon, Giheung, Yongin,
Gyeonggi, 446-701 Korea
Fax: +82 31 204-9082
E-mail: rossi@networking.khu.ac.kr

Mosaddek Hossain Kamal

Department of Computer Science and Engineering,
University of Dhaka,
Dhaka-1000, Bangladesh
Fax: +88
E-mail: tushar@univdhaka.edu

Muhammad Mostafa Monowar

Department of Computer Engineering,
Kyung Hee University,
1 Seocheon, Giheung, Yongin,
Gyeonggi, 446-701 Korea
Fax: +82 31 204-9082
E-mail: monowar@ieee.org

Choong Seon Hong*

Department of Computer Engineering,
Kyung Hee University,
1 Seocheon, Giheung, Yongin,
Gyeonggi, 446-701 Korea
Fax: +82 31 204-9082
E-mail: cshong@khu.ac.kr
*Corresponding author

Abstract:
SIP, H.323 and Jingle have been widely used as VoIP protocols. As the session establishment period differs in these protocols and their different implementations, interoperability problems occur. This paper presents a middleware on mobile devices with a generic framework supporting the three protocols and their different implementations. The applications (rmobvoip, rmoblog, mobrmi, mobcorba) implement distributed protocols to solve interoperability problems and to

overcome the scarce resources of mobile devices. A peer with this middleware is provided with three options(SIP, H.323 and Jingle) and it can choose any of the protocols or implementations depending on the protocol type used by other peer of the session.

Keywords: SIP; H.323; Jingle; interoperability; middleware; streaming session;VoIP.

Reference to this paper should be made as follows:Kamal, R., Kamal, M.H.,Monowar, M.M. and HONG, C.S. (2010) 'A Mobile Middleware to Solve Interoperability Problems in VOIP Streaming Session', *International Journal of Communication Networks and Distributed Systems (IJCNDS)*

Biographical notes: Rossi Kamal is currently doing masters at Networking lab, Department of Computer Engineering, Kyung Hee University (KHU), Korea and he received B.Sc. in Computer Science and Engineering from University of Dhaka, Bangladesh in 2009. His research area includes middleware and distributed computing, sensor network.

Mosaddek Hossain Kamal received his B.Sc and M.Sc degree from University of Dhaka, Bangladesh. He received another M.Sc degree from University of New South Wales of Australia. He has been working as an associate professor in deparment of Computer Science and Engineernig of University of Dhaka. His research area includes distributed computing, enterprise application development

Muhammad Mostafa Monowar is currently doing Ph.D. at Networking lab, Department of Computer Engineering, Kyung Hee University (KHU), Korea and he received B.Sc. in Computer Science and Information Technology (CIT) from Islamic University of Technology, Bangladesh (IUT) in 2003. He is an Assistant professor (On leave) at CSE department, University of Chittagong, Bangladesh. He is a student member of IEEE. His research interest includes MAC and Transport layer solutions for Wireless Sensor Networks

Choong Seon Hong received his B.Sc. and M.Sc. degrees in electronic engineering from Kyung Hee University, Seoul, Korea, in 1983, 1985, respectively. In 1988 he joined KT, where he worked on Broadband Networks as a member of the technical staff. From September 1993, he joined Keio University, Japan. He received the Ph.D. degree at Keio University in March 1997. He had worked for the Telecommunications Network Lab, KT as a senior member of technical staff and as a director of the networking research team until August 1999. Since September 1999, he has been working as a professor of the School of Electronics and Information, Kyung Hee University. He has served as a Program Committee Member and an Organizing Committee Member for International conferences such as NOMS, IM, APNOMS, E2EMON, CCNC, ADSN, ICPP, DIM, WISA, BcN and TINA. His research interests include ad hoc networks, network security and network management. He is a member of ACM, IEEE, IPSJ, KIPS, KICS and KISS

1 Introduction

In a distributed computing system, middleware is defined as the software layer that lies between the operating system and the applications on each side of the system. Middleware makes application development easier, by providing common programming abstractions, masking the heterogeneity and the distribution of the underlying hardware and operating systems and by hiding the low level programming details. A middleware on the embedded device can solve the interoperability problems of different protocols and their implementations on different devices.

VoIP means transmitting voice data over internet. VoIP is implemented by different protocols. H.323 (Hersent, 2010), a standard of ITU (International Telecommunication Union) is the mostly widely used. The comprehensive and also complex protocol was originally designed for video conferencing. Specifications like real time , interactive video-conferencing, data sharing and audio applications such as VOIP were included to a suite of protocols named H.323. It incorporates many individual protocols for specific applications. The problem with H.323 is that it is not specially tailored to VoIP. An alternative to H.323 emerged with the development of SIP (Zenner, 2004). SIP is a more streamlined protocol, developed specially for VoIP applications. Smaller and more efficient than H.323, SIP takes advantage of existing protocols to handle certain parts of the process. XMPP (Liuhto and Mantyasaari, 2005) also provides a signaling protocol, typically via the Transmission Control Protocol. Given the significant differences between XMPP and SIP, it is difficult to combine the two technologies in a single user agent. Therefore, developers wishing to add media session capabilities to XMPP clients have defined an XMPP-specific negotiation protocol called Jingle (Zenner, 2009). Voice over Inter Protocol is implemented using SIP(session initiation protocol), H.323 and Jingle. An intermediate server establishes a secure communication session between the peers and then they exchanges voice data in real time transport protocol. As the session establishment period differs in the three and there are variations in protocol implementations, interoperability problem occurs.

The solution could be server based or client based or an integration of both. We have studied existing open source and commercial implementations of SIP. We have found interoperability within SIP, H.323 and Jingle as a major problem. Existing solutions involve presence of an interim gateway between VOIP implementations. This gateway can interoperate VoIP calls of different types of VoIP clients. If a client with protocol A wants to make a VoIP call with a client with protocol B, the gateway encodes the VoIP message of protocol-A client and decodes it for protocol-B client. As the VoIP gateways handle a large numbers of VoIP calls and huge voice traffic is associated with it, they have to undergo a huge overhead. For example, generally, H.323 systems use native key management or pre-SRTP encryption. But, it also supports SRTP and some SRTP keying scheme like MIKEY. When we need to interoperate between H.323 and SIP using SRTP keying mechanism, gateway device decrypts SRTP (with

one key) towards one network and encrypt SRTP (with different key) towards other network. To interoperate DTLS-SRTP endpoint of SIP and H.323, interim device simply forwards SRTP packet from H.323 network towards DTLS-SRTP key transport layer. These cause huge overhead to server and keeping a server in this way is also a costly solution. But, if we can provide a generic solution in VOIP clients, they will 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 protocols. A mobile client software should have support for VOIP protocols SIP, H.323 and Jingle and their different implementations. If it does not have that support, it has to use implementations of other clients in network. We are motivated towards utilization of distributed SOAP (Zenner, 2007), RMI (Downing, 1998) and CORBA (Schmidt and Kuhns, 2004) on mobile to resolve that problem. RMI is a simple method for distributed computing with Java objects where objects can be called from other Java Virtual Machines. In order to support NON-JVM context, CORBA was introduced. CORBA allows heterogeneous, distributed collections of objects to interoperate. Client and object implementations are portable in CORBA. Code written for one vendor's CORBA could be with a minimum effort written to other vendor's product. CORBA's interoperability is ensured with IIOP (Internet Inter ORB protocol) . IIOP is also used in transport protocol of RMI. Programs written for RMI can interoperate with each other and even with programs written for CORBA. SOAP allows multi-stream and multitype data transfer among VoIP clients. Implementing CORBA and RMI to mobile platform is a challenging issue. Existing approaches use interim servers (co-existing with mobile device) to implement server or servant or IDL needed for RMI or CORBA. But, we were interested on implementing RMI or CORBA only on mobile devices without involvement of any server. We moved toward a generic VOIP solution that will explore distributed protocol implementation (CORBA, RMI and SOAP) overcoming limited resource available on mobile devices.

Our proposed mobile middleware solves VoIP interoperability problems with its generic framework with support for the SIP, H.323 and Jingle and their different implementations. This middleware utilizes distributed protocols to overcome the scarce resources of embedded devices (such as low battery power, slow CPU-speed and little memory; they are recurred to react to frequent and unannounced changes in the environment, such as high variability of network bandwidth and in the remote resource availability and so on). SOAP, RMI, CORBA explore remote object and resources to solve the interoperability problem. A user agent using this middleware is provided with three options (SIP, H.323, XMPP-VoIP) and it can choose any option depending on the protocol or the specific implementation of a protocol used by other end of the session.

This paper is organized as follows. Section 2 decribes the interoperability problems of SIP, H.323 and Jingle. Section 3 describes the need for a mobile middleware, its comparison with existing middleware, key fetures of a mobile middleware. Section 4 describes our proposed mobile middleware- its architechture, functionalities and implementation details. In functionalities and implementation, we have considered three VoIP interoperability scenerios: A. VoIP client with H.323 is setting a VoIP session with SIP client, B. VoIP client with SIP ZRTP

Subject	SIP	Jingle
Message format	SIP uses message formats and header almost like HTTP protocol.	XMPP-VOIP uses XML in this case.
SDP feature	SIP uses SDP (Session Description Protocol) in payload declaration stage	Jingle uses XML .But Jingle can map SDP to XML
Session establishment period	SIP establishes session in UDP	Jingle uses TCP .So, application layer change or modification does not have any effect.

Table 1: Protocol Differences between Jingle and SIP

is setting a VoIP session with SIP DTLS-MIKEY client, C. VoIP client with SIP is setting a VoIP session with another SIP client. We have implemented our mobile middleware on Java ME (Muchow and John, 2001) platform with four applications rmobvoip (Kamal, 2009a),rmoblog (Kamal, 2009b), mobrmi(Kamal, 2009c), mobcorba (Kamal, 2009d). In section 5, we have described related works and finally we have concluded in section 6.

2 Interoperability problems in VoIP

VoIP (Voice Over Internet Protocol) is implemented using different protocols like SIP, H.323, Jingle, MCGP etc. There are also variances among each protocol implementations (for example SRTP-MIKEY, ZRTP variances of SIP). The protocol used by caller has to interoperate with protocol used by the callee. The scenario seems quite challenging for the 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.

2.1 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. We have shown the protocol differences between SIP and Jingle in Table 1.

2.2 Protocol differences between SIP and H.323

H.323 and SIP both have advanced features in a packet voice network. It is a better idea to place them in the same network and interoperate them in the right way. Although they have good many of functions in common, their implementations are different in both platforms. We have shown the protocol differences between SIP and H.323 in Table 2.

Subject	SIP	H.323
Basic feature	SIP clients can talk to each other in media-session without any involvement of an interim server. Server first sets a session between them and then clients transfer media data between them.	H.323 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 H.323 with VOIP gateways

Table 2 : Protocol Differences between SIP and H.323

2.3 Analysis 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 the three in Table 3.

3 Need for Mobile Middleware

With the advancement of wireless network and mobile computing, new middleware system with support for mobility has become challenging concept. Traditional middleware systems are based on the concept that physical locations of hosts and connection among them do not change. In a mobile environment, this is inappropriate. Mobile devices face loss of network connectivity while on move, may have limited capabilities like slow CPU, little memory. Mobile middleware system has to adapt to those challenges.

3.1 Traditional middleware and mobile middleware

Traditional middleware systems have been successfully implemented in distributed applications but it will not be able to meet the needs imposed by mobile computing. Stationery middleware system is built with fixed network and requires high communication bandwidth. But mobile middleware has to deal with unreliable, low bandwidth connection. Synchronized paradigm of communications is used by traditional system and client and server should run autonomously. Client and server are not kept connected due to mobility of nomadic users and devices. Transparency causes traditional middleware to shield context information of execution environment from application and to make decision on application behavior. But this is inefficient in dynamicity and heterogeneity of mobile environment. Applications must have enough information that allows mobile middleware to take better decision based on scenarios of context.

3.2 Mobility, mobile computing and mobile middleware

Mobility is a major issue in middleware technology. Mobility can be of logical or physical entity. Logical entity involves movement of mobile computing entities like sample data, objects, mobile agents or whole applications.

Subject	MIKEY	SDES-TLS	ZRTP
Feature	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 the most resource-intensive. It provides perfect forward secrecy .But it requires peer to peer communication and existence of PKI	SDES-TLS protocol is based on SDP on background. 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 unicast 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. It is 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 secured and reliable.	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 it is done with MIKEY. Therefore, key negotiation is done between pairs without the use of intermediates such as SIP 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.

Table 3: SIP variances MIKEY, SDES-TLS and ZRTP.

In mobile computing where computational components may change their locations, locations must allow mobility of users, devices or code. Users can access service resources regardless of physical location or movement-behavior.

Mobile distributed system has to face the challenges of mobile computing but the critical point is which layer it will use for the purpose. Mobile middleware concept comes in this case and to support mobility, it needs to implement lots of complex information. But these information is kept hidden in middleware itself, mobility is transparent to applications. But, in addition mobile middleware may interact with underlying network operating system and takes adaptive decisions based on context information.

3.3 Key features of a mobile middleware

Mobile systems are based on extremely dynamic contexts and they need to be lightweight system, with features of synchronization and application developers aware of execution context.

Mobile applications are depended on low featured devices. So, mobile middleware has to be lightweight for a feasible solution. Mobile devices connect to network opportunistically for some periods of time to access data or request service. An asynchronous form of communication is necessary. Bandwidth is not stable and service may not be always available. Middleware itself does not consider possible contexts and behave to corresponding situation. It interacts with application that will be aware of context change. Middleware adapts to change based on the information given by application

3.3.1 Environmental Adaption

Dynamicity of mobile environment and limited bandwidth have made adaptation as essential element of a mobile middleware system. Adaptation techniques involve systems and applications respond to environmental changes and resource limitations

3.3.2 Flexible Computing Mode

Mobile environments bring wide variations and rapid changes in network connections and local resource-availability. To cope with dynamic environment, mobile middleware adaptively adjusts functionality of computation between clients and servers in response to changes in dynamic network environment.

3.3.3 Loose Couple Computing

A loose couple style of distributed computing exists in mobile middleware system. This style allows middleware to cope with intermittent connections and low network bandwidth. If we consider application view, this loose coupled system is observed with a new asynchronous paradigm of traditional client server system.

3.3.4 Resource Discovery

Resource discovery is an essential attribute as dynamic environment will move and face resource limitations. How to publish resource information, how to cope with invalidation of information and how to query neighbor for resources-these are considered in resource discovery process. Centralized and distributed approaches are used in this context. Centralized approach maintains dynamic resource information in global directory and distributed approach tracks neighbor availability in mobile areas.

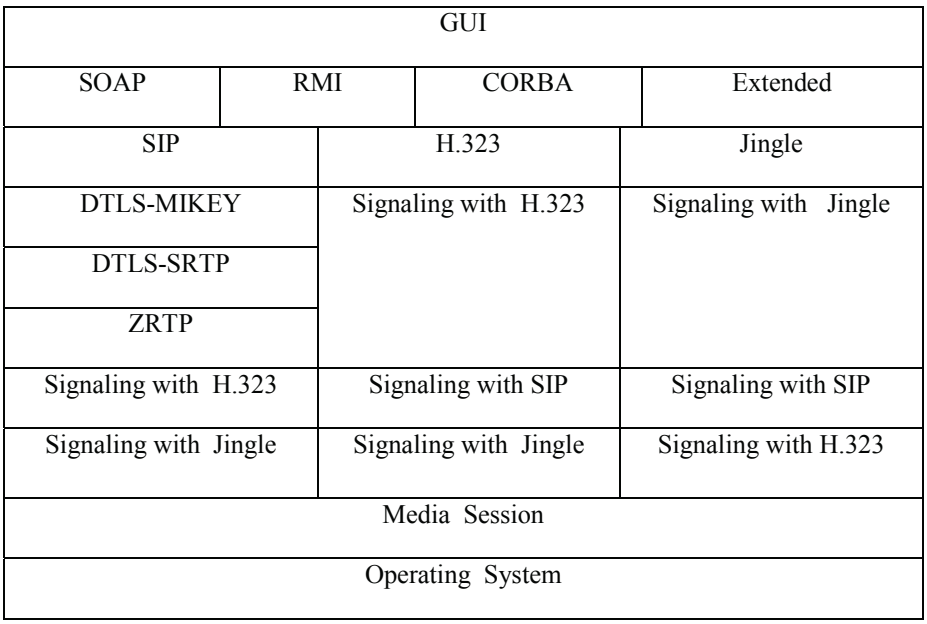

Figure 1 Architechture of proposed mobile middleware

4 Proposed Mobile Middleware

4.1 Architechture

Proposed mobile middleware in Figure 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 the integration of new distributed protocol in 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 sub-components. 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 signaling with SIP variances namely DTLS-MIKEY, DTLS-SRTP or ZRTP or with H.323 and Jingle. H.323 component supports signaling with H.323, Jingle and SIP. 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 signaling session established by upper layer.

4.2 Functionalities

We discuss the functionalities of the middleware with some problem-scenarios. In Figure 2, a mobile with H.323 protocol wants to set a VoIP session with a mobile with SIP protocol. As two signaling protocols differ, H.323 mobile uses

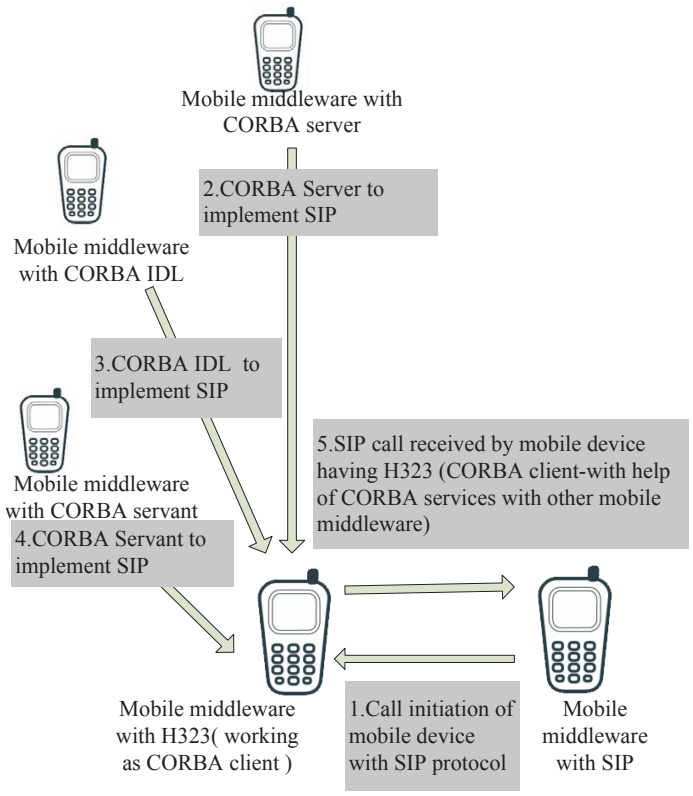

Figure 2 Mobile middleware with H.323 is making VOIP session with mobile middleware with SIP. In this case, first middleware is working as CORBA client,using three other mobile middleware as CORBA server, servant and IDL

mobile middleware to set session with SIP mobile. It works 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 signaling session with SIP mobile. In Figure 3, 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 Figure 4, mobile middleware with SIP protocol is communicating with a mobile middleware with SIP. Both middleware are using SOAP component to explore multi-type, multi-stream communication in a VoIP session.

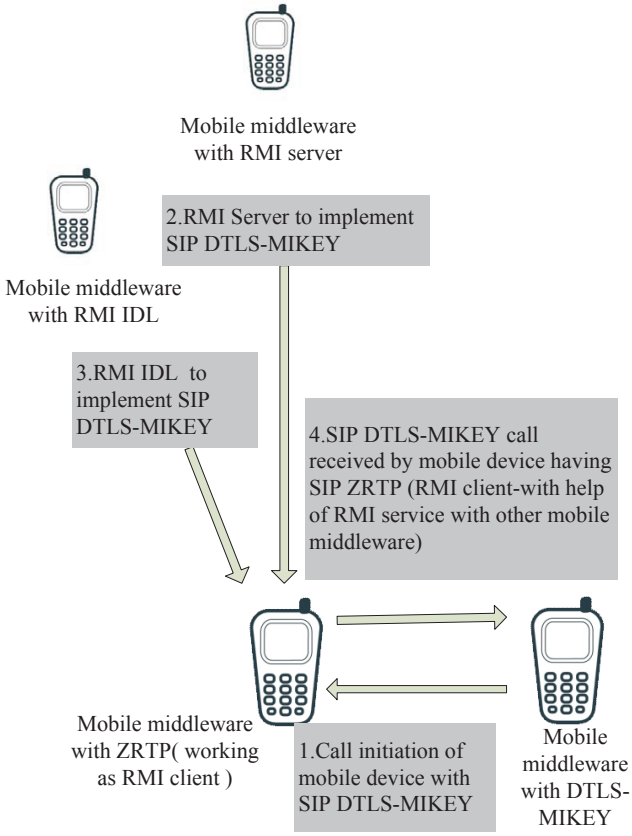

Figure 3 Mobile middleware with SIP-ZRTP is making VOIP session with mobile middleware with SIP DTLS-MIKEY. In this case, first middleware is working as RMI client,using two other mobile middleware serving as RMI server and IDL

4.3 Implementation

We have implemented our mobile middleware on Java ME platform. Four applications rmobvoip (Kamal, 2009a),rmoblog (Kamal, 2009b), mobrmi(Kamal, 2009c), mobcorba (Kamal, 2009d) are developed with j2me programming language to evaluate the performance of the proposed middleware. Class diagram of the middleware in Figure 5 describes how the internal operation goes in the middleware. CORBAImpl, RMIImpl and SOAPImpl classes are responsible for distributed 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 protocols. We consider the internal operation when a mobile middleware with H.323 sets a VoIP session with a SIP mobile in Figure 2. The H.323 mobile creates object of CORBAImpl and implements the CORBAClientImpl() opertion. It invokes three other objects of CORBAIDLImpl and those three

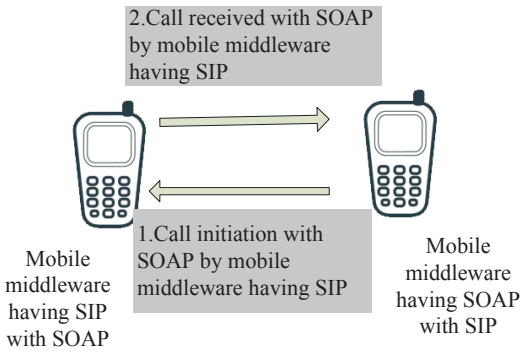

Figure 4 Mobile middleware with SIP is making VOIP session with mobile middleware with SIP . In this case, two middleware are communicating with help of SOAP

objects are implementing operations CORBAIDLImpl(), CORBAServantImpl() and CORBAServerImpl() respectively. These three objects invoke H323SipImpl() to implement SIP for the caller mobile middleware with H.323. Thus, mobile middleware with H.323 uses three other mobile middleware clients to make a VoIP session with a SIP mobile using CORBA. We can consider the scenerio of Figure 3 where two SIP variances want to make VoIP session with each other. 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 in Figure 4, SOAP implementation allows them multitype and multistream VoIP session. First middleware creates SOAPImpl object and perform SOAPDefImpl() operation. It then creates object of SIPImpl and perform SIPDefImpl() operation to set a VoiP sesion with other middleware with SIP. After the session is established, the mobile middleware invokes MediaImpl object and perform MediaDefImpl() operation to talk in the media session of a VoIP call.

5 Related Works

SMILE-JS (Bartolomeo, Salsano and Polidoro, 2008) is an implementation of a new middleware for mobile VOIP devices exploring distributed protocols for SIP and JSON (Java-script Object Notation). This has been an attempt to port SMILE (Bartolomeo et all., 2008) framework to embedded device. It deals with SIP only and J2ME environment is considered in the middleware architecture. Collaborative multimedia middleware (Kim et all., 2001) describes a middleware architecture composed of basic real-time service components for teleconferencing, screen-sharing and collaborative browsing and tele-presentation so that a variety of

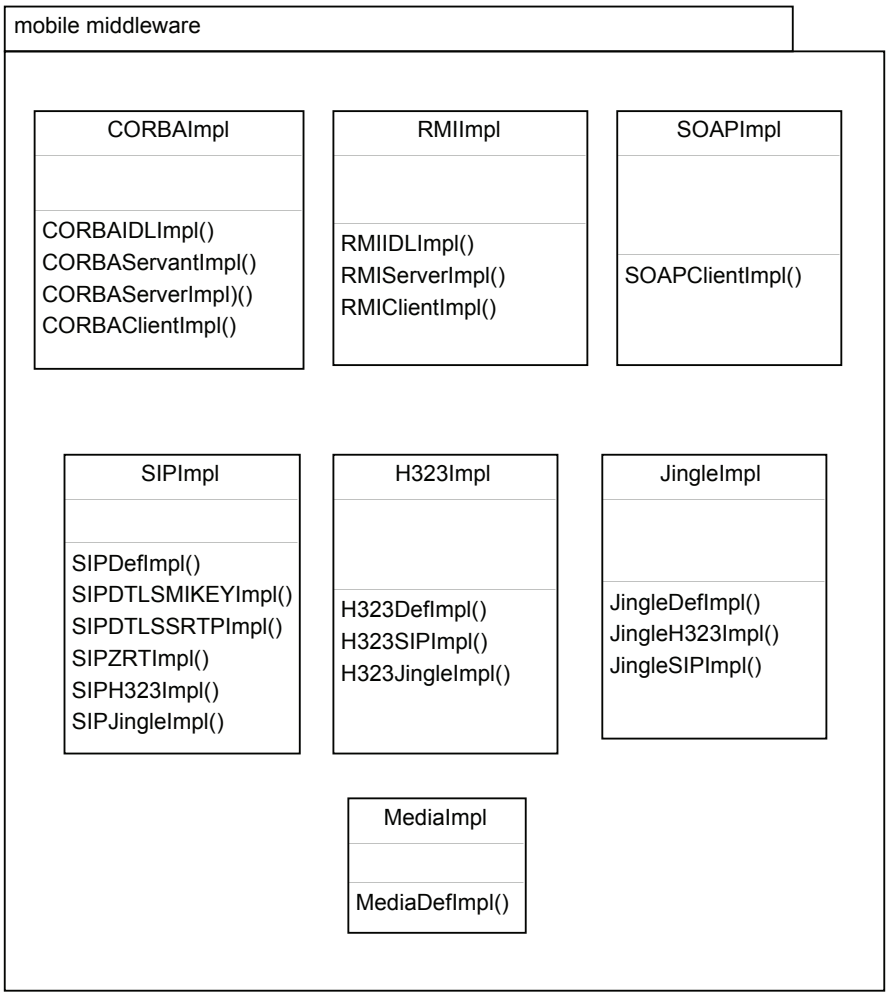

Figure 5 Class diagram of mobile middleware implementation

application services can be built on it in an easy and inter-operable manner. Mobile computing middleware (Mascolo, Capra and Emmerich, 2002) presents 'Middlware aims at facilitating communications and coordination of distributed components, concealing difficulties raised by mobility from application engineers as much as possible', 'characteristics of mobile distributed systems and distinguish them from their fixed counterpart', 'a framework and a categorization of the middleware systems designed to support mobility'. Sun Micro System has recently introduced Mobile Enterprise Platform(MLP), a framework for building mobile applications synchronized with enterprise databases. The framework sits between Java based client front end, deployed to a mobile device and a traditional server application'. Communications between the mobile device and application are performed with data synchronization from Open Mobile Alliance, CLDC and MIDP.

6 Conclusion

In this paper, we have proposed a mobile middleware that solves interoperability problems of VoIP protocols namely SIP, H.323 and Jingle. Our middleware uses distributed protocols RMI, CORBA and SOAP to overcome the limited energy, memory of mobile devices. Our middleware is provided with software stack of SIP, Jingle and H.323. Any instance of middleware may have full or partial (any of three protocols) implementation depending on the memory and power of the mobile device. If a mobile device has H.323 implementations but wants to communicate with SIP mobile, it will use other mobile devices (having middleware instances) implementing SIP. The middleware is lightweight in this sense. In future, we will work on how efficiently we can import software stack from other middleware instances. Inspiration from mobile middleware for VOIP services has motivated us for an open source VOIP solution. This will be platform independent and will be with XML web services.

Acknowledgements

This work was supported by the IT R&D program of MKE/KEIT [10035245: Study on Architecture of Future Internet to Support Mobile Environments and Network Diversity].

Appendix

rmobvoip

rmobvoip is an application implementing VoIP in mobile.

rmoblog

rmoblog is an application implementing SOAP on mobile device.

mobcorba

mobcorba is an application implementing CORBA on mobile device

mobrmi

mobcorba is an application implementing RMI on mobile device

References

Bartolomeo, G., Salsano, S. and Polidoro, A. (2008) 'SMILE-JS, a SIP-based Middleware for J2ME Devices', *Proceedings of MobMid '08, the 1st workshop on Mobile middleware*, Leuven, Belgium, pp.1-6.

Bartolomeo, G., Salsano, S., Melazzi, N.B. and Trubiani, C.(2008) 'SMILE - Simple Middleware Independent LayEr for Distributed Mobile Applications', *Proceedings of IEEE WCNC08, Wireless Communications and Networking Conference*, Las-Vegas, U.S.A, pp.3039-3044.

Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, F., Thatte, S., Winer. D.(2007) 'Simple Object Access Protocol (SOAP) 1.1', *http://www.w3.org/TR/soap/, Accessed on 13 October, 2010.*

Downing, T.B.(2010) 'Java RMI:The Remote Method Invocation', *DG Books Worldwide.*

Hersent, O.(2010) 'H.323: Packet-Based Multimedia Communications Systems', *IP Telephony*, John Wiley & Sons, Ltd.

Kim, D.H, Park, S.M., Kim, J.,Y., Sul, D.,M., Lee ,K.,H.(2001) 'Collaborative multimedia middleware architecture and advanced Internet call center', *Proceedings of the 15th International Conference on Information Networking 2001*, Oita, Japan, pp.246-250.

Mascolo, C. , Capra, L. and Emmerich, W. (2002) 'Mobile computing middleware', *Advanced lectures on networking*, Springer-Verlag, NY, USA, pp. 20-58.

Liuhto, L. and Mantysaari, V. (2005) 'Extensible Messaging and Presence Protocol (XMPP)', *Proceedings of the seminar on instalnt messaging and presence architectures in the internet*, Finland.

Ludwig, S., Beda, J., Andre, P., McQueen, R., Egan, S. and Hildebrand, J. (2009) 'XEP-0166: Jingle', *http://xmpp.org/extensions/xep-0166.html, Accessed on 13 October, 2010.*

Muchow and John, W. (2001) 'Core J2ME Technology and MIDP', *Prentice Hall PTR*, Upper Saddle River, NJ, USA.

Kamal, R. (2000) 'rmobvoip - An implementation of VOIP on embedded device', *http://code.google.com/p/rmobvoip, Accessed on 13 October, 2010.*

Kamal, R. (2000) 'rmoblog - An implementation of SOAP enabled blogging on embedded device', *http://code.google.com/p/rmoblog, Accessed on 13 October, 2010.*

Kamal, R. (2000) 'mobrmi - An implementation of RMI on embedded device', *http://code.google.com/p/mobrmi, Accessed on 13 October, 2010.*

Kamal, R. (2000) 'mobcorba - An implementation of CORBA on embedded device', *http://code.google.com/p/mobcorba, Accessed on 13 October, 2010.*

Schmidt, D.G.; Kuhns, F. (2000) 'An Overview of the Real-time CORBA Specification', *Computer*, vol.33, no.6, pp.56-63.

Zenner, G. J. (2004) 'Session Initiation Protocol', *Bell Labs Technical Journal*, vol.9, pp.1-3.