Development of Doubango-based IMS UE for Teaching Experiments

Youzheng Chen, Baozhong Cheng, Xiaoyan Zhang, Yi Sun, Jin Xie and Kun Li
School of Software Engineering
Beijing University of Posts and Telecommunications
Beijing, China
Mr.Youzheng.Chen@ieee.org
chengbaozhong@email.buptsse.cn

***Abstract*—IP Multimedia Subsystem (IMS) is well known for its combination of the capability of universal coverage of mobile network and rich application development environment of IP network. IMS has been welcomed by both industry and academic society, becoming the first choice of the Next Generation Network (NGN) technologies. This paper describes the design and implementation of three IMS teaching lab experiments, based on open source Doubango project. The lab experiments are designed for graduate students and senior undergraduate students to learn how to design and implement an application for IMS user equipment (UE). The paper analyzes the requirements of the experiments, presents an object-oriented software solution, and explains its Doubango-and-OpenIMSCore-based implementation.**

Keywords*—IMS; SIP; Presence Service; Doubango; Teaching case; C#

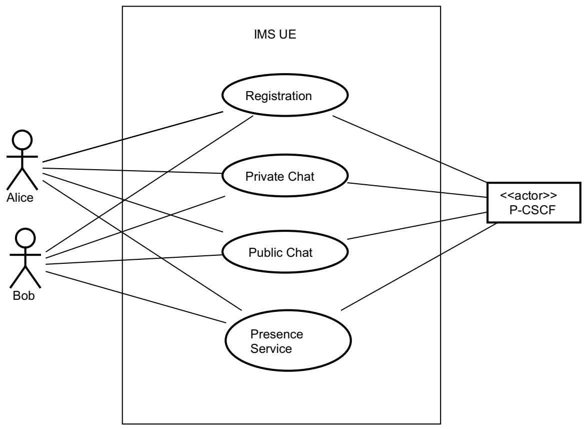

Fig. 1. IMS UE UseCase Diagram

I. Introduction

In the recent years, telecommunication network has faced the challenges of fierce changes on the requirement and technology. Internet Protocol Multimedia Subsystem (IMS) has been proposed by the Third Generation Partnership Project (3GPP) in Release 5 [1]. IMS is a session control system based on Session Initiated Protocol (SIP) [2] which is defined by the Internet Engineering Task Force (IETF), and it is one of the main development directions in the Next Generation Network (NGN). [3]

We designed the lab experiments for graduate students and undergraduate students to study about design and implementation of IMS user equipment (UE) applications. The purposes of the teaching experiments are explained in section two. Section three presents the analysis of requirement. Section four focuses on the design and implementation of the IMS UE development experiment. Section five mainly talks about how to apply and arrange this simple IMS client in teaching cases. Finally, Section six makes a conclusion of this paper and describes the prospect of teaching cases on IMS.

II. Our purpose

As the IMS networks being deployed worldwide, more and more IMS professionals are needed. Some universities have opened IMS technology courses for graduate and undergraduate students, but few papers can be found to describe how to teach students IMS technology in a practical training course. The main purpose of our project is to design and implement some IMS lab experiments, in order to help graduate and undergraduate students to master the IMS UE software development skills in a period of three weeks.

III. Requirement Analysis

In this simple IMS client, in order to meet the requirement of teaching cases which is limited in three weeks, we can decompose the whole implementation into four parts: registration function, private chat function, public chat function and presence service function (shown in Figure 1).

In details, in registration function, a point-to-point connection between the user and the Proxy-Call Session Control Function (P-CSCF) will be established for checking user authorization information in some steps. In private chat function, a point-to-point connection between two IMS clients will be established without forwarded by application server (AS). In public chat function, a client-server connection will be established between the AS and each of the participators who want to join in the specific chat room, so that all the messages will be forwarded to other members in the specific chat room by AS. In presence service function, it is accurately thought to be a simplified presence service. The data format that presented presence information is the Presence Information Data Format (PIDF) according to the RFC3863 [4]. A client-server connection will be established, so that all the users who use the presence business which is defined by user profile in

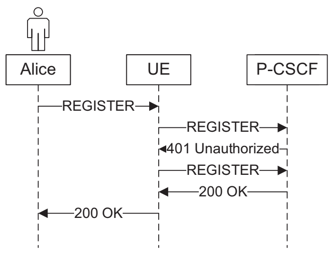

Fig. 2. The Call Flow of IMS Registration

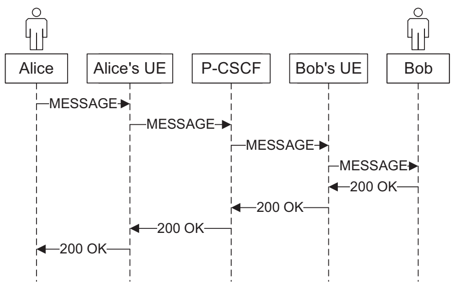

Fig. 3. The Call Flow of IMS Private Chat

Home Subscriber Server (HSS) will exchange presence data with presence service AS.

First of all, an IMS client has to register first for the access to IMS core network, which is shown in Figure 2. We can assume that a user named Alice wants to register in an IMS core network. Her IMS UE should send a REGISTER request to the P-CSCF. Then P-CSCF will return 401 Unauthorized to challenge the UE. With the challenge information, a second request sent by Alice's UE will be returned 200 OK by P-CSCF as the response to Alice's UE if it is authorized. [5]

Secondly, the process of private chat function is shown in Figure 3, assuming that a user named Alice wants to communicate with another user named Bob, and they have already registered in the same IMS realm. Alice's IMS UE will send a message to P-CSCF with Bob's URI and the content. As a SIP Proxy, P-CSCF will forward this message to Bob. This is a simplified model that two clients communicate without INVITE and ACK. [6]

Thirdly, in the scenario about the process of public chat function, a new role will be included - *Chatroom* Application Server (AS). We can assume that a user named Alice and another user named Bob both want to communicate in the same chat room and have already registered in the same IMS realm. Alice's IMS UE will send a message with the URI of the *Chatroom* AS and the content. As a SIP proxy, P-CSCF will forward this message to the AS. Assuming that there has been no member in the chat room before, AS doesn't have to forward this message to others. And then if Bob do the same work in his IMS UE, AS will forward this message to Alice since Alice has been in chat room.

Fourthly, the communication process of presence service function is shown in Figure 4. Presence AS is added in. The precondition is that a user named Alice and another user named Bob both have registered in the same IMS realm, and they are defined to use presence service. In order to simplify the model to fit teaching cases, we ignore the XCAP server which acts as

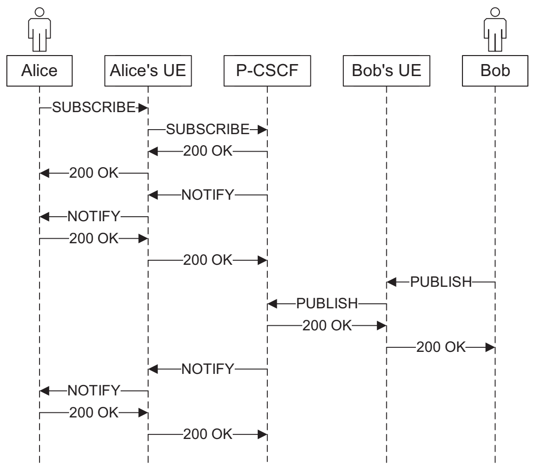

Fig. 4. The Call Flow of IMS Presence Service

the presence rule manager. When Alice uses her IMS client to send SUBSCRIBE request to P-CSCF with Bob's URI, and P-CSCF will forward this request to the Presence AS to response this request. AS will give a NOTIFY response to describe the Bob's status. A moment later, Bob send a PUBLISH request to P-CSCF with his new status, when AS receives this request, if the expired time of Alice's SUBCRIBE is not to, AS will send a NOTIFY message to Alice to update Bob's status. [5]

IV. Design and Implementation

In order to save hours in identifying defects and performance problems, we use Microsoft Visual Studio as our development toolkit to develop simple IMS client. The runtime platform of this development is .NET Framework 3.5 because of the limitation of Microsoft CSharp (C#) Dynamic Link Library of Doubango IMS Framework. [7]

The Fraunhofer Institute FOKUS has developed the Open IMS Core as an Open Source implementation of IMS Call Session Control Functions (CSCFs) and a lightweight Home Subscriber Server (HSS) in 2006 under the open source GNU General Public License version 2 (GPLv2). [8]

The Doubango IMS Framework is a protocol stack designed by Doubango Telecom as an open source project, which supports both voice and SMS over Long Term Evolution (LTE) defined by the One Voice Initiative. The Doubango IMS Framework contains interface for using SIP Message and Presence Service. [9] In Windows 7, developers can choose Microsoft CSharp (C#) as the main language to develop a high-quality IMS client. However, most of the samples and the open source applications are too sophisticated to apply in a teaching case.

The architecture of simple IMS client indicates that in Doubango IMS Framework, we can divide the client implementation into three layers: Application layer, High layer, and Middle & Low layer. Layers below High layer (including High layer) are implemented in Doubango IMS Framework, which helps us ignore the sophisticated construction of SIP stack implementation. In order to design a teaching case of IMS client, we don't have to consider about the implementation of *tinySIP*, *tinyMEDIA*, *tinyDAV* and so on. The aspect we

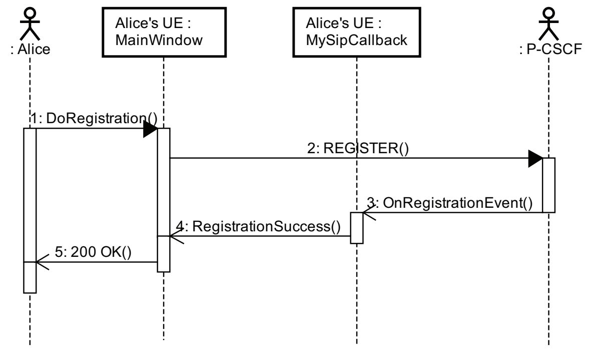

Fig. 6. Sequence Diagram of Registration in Simple IMS Client

should consider is how to call the functions by the interfaces provided by *tinyWRAP*. What we have designed is simply the application layer to match to teaching cases of IMS client. [9]

In Doubango framework, there are several interfaces for us to design the implementation of simple IMS client. *RegistrationSession* is mainly used as a session when the IMS client doing a registration. *MessagingSession* provides the context to send SIP messages to an IMS client or an AS. *PublicationSession* is used to encapsulate a PUBLISH request and send it from IMS client to a specific presence server. *SubscriptionSession* is used to encapsulate a SUBCRIBE request and send it from IMS client to a specific presence server. *SipCallback* provides a programming mechanism that arriving messages can be received conveniently.

Therefore, a simple IMS client can use the service of Doubango IMS stack through a set of well-defined interface provided by Doubango IMS Framework. The class diagram of the simple IMS client is shown in Figure 5. The class *MainWindow* is responsible for the logic of user interface (UI). The class *MySipCallback* is a subclass inherited by *SipCallback*, which is responsible for receiving data from SIP stack. The class *PresItem* is a data structure represents the simplified presence data.

First of all, an IMS client has to initialize the SIP protocol stack. In stack initialization, we should configure the related properties in program, including SIP Callback function, IP Multimedia Public Identity (IMPU), IP Multimedia Private Identity (IMPI), IMS Realm, and P-CSCF address.

Figure 6 shows the sequence diagram in registration of simple IMS client. The function *DoRegistration* is designed to configure personal information and use the information to register user in Open IMS Core. The class *MySipCallback* contains an override function *OnRegistrationEvent* to receive the status of user registration. When receiving the status 200 OK, it will call the function *RegistrationSuccess* in class *MainWindow*.

In the Private Chat function, the function *DoMessaging* is designed to send the message from user to Open IMS Core. The override function *OnMessagingEvent* of class *MySipCallback* will detect the status of message transmission. When Open IMS Core forward the message to another user, the override function *OnMessagingEvent* will be ready for receiving the message, and then call the function *GetMessaging* in class *MainWindow* to display the message in screen.

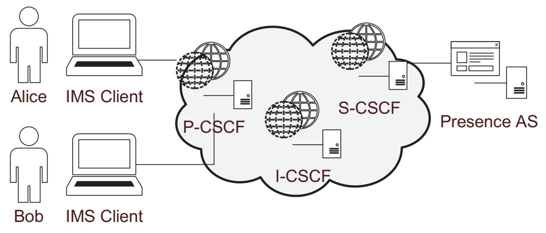

Fig. 8. Network Topology of Presence Service

In public chat function, the additional job is to change property *ToUri* of messaging session to the SIP URI of application server.

The sequence diagram of Presence in Simple IMS Client is shown in Figure 7. We can separate this sequence into three parts: Subscription Session, Publication Session and Notify Callback. First of all, some specific headers-*"Event"*, *"Accept"*, and *"Allow-Events"*-should be added. In Subscription Session, we should send a Reg SUBSCRIBE request to register in Presence Server at first. Then begin subscription by calling function *DoSubscription* in class *MainWindow*. The override function *OnSubscriptionEvent* in class *MySipCallback* will receive the NOTIFY message from P-CSCF and call the function *ChangeStatus* in *MainWindow* to display new status in screen. The function *DoPublication* in class *MainWindow* is designed to send PUBLISH message to publish user status to Presence Server. The status information will be written in PIDF format and serialized to the byte stream. [4]

V. Experiment Series

In teaching cases, we separate the whole simple IMS client into three parts to gradually instruct students to study the design and implementation of IMS client and the basic mechanism of IMS core network:

- IMS-based Windows Half-duplex Chat Room in Console
- IMS-based Windows Full-duplex Chat Room in GUI
- IMS-based Windows Presence Service

In the first teaching case, all the students have to focus on the mechanism of IMS chat client instead of caring much about concurrent threads problem in GUI programming. In console, what they have to do is to implement an IMS-based half-duplex private chat function so that Alice and Bob can communicate with each other. If it can't work well, students could use the tools to detect the network transmission data, such as Wireshark, to check the positions of bugs. When students have known well about the mechanism of IMS chat client, a full-duplex chat room is the new requirement for them in the second teaching case.

Figure 8 indicates the topological structure of Presence Service. Students have to learn some new API for Presence Service. Both the system robustness and the user experience are the evaluation exponents of students' report. The solution of concurrent threads in the second teaching case may help them program efficiently.

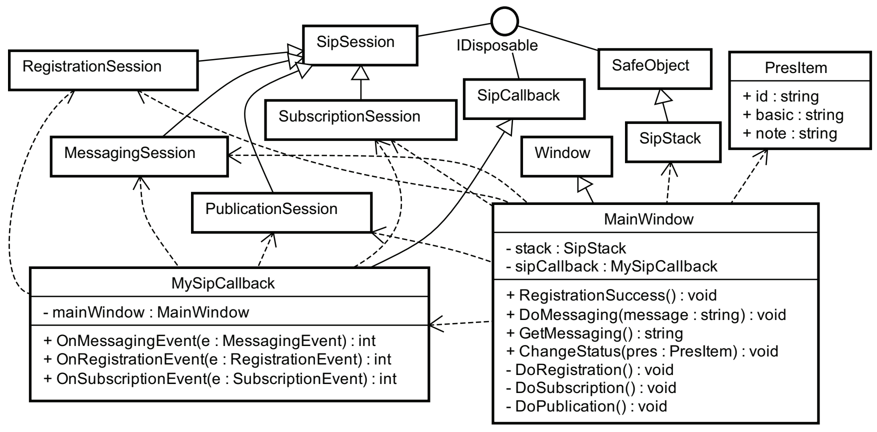
SipSession
IDisposable
SafeObject
PresItem
+ id : string
+ basic : string
+ note : string
RegistrationSession
SubscriptionSession
SipCallback
SipStack
Window
MessagingSession
PublicationSession
MainWindow
- stack : SipStack
- sipCallback : MySipCallback
+ RegistrationSuccess() : void
+ DoMessaging(message : string) : void
+ GetMessaging() : string
+ ChangeStatus(pres : PresItem) : void
- DoRegistration() : void
- DoSubscription() : void
- DoPublication() : void
MySipCallback
- mainWindow : MainWindow
+ OnMessagingEvent(e : MessagingEvent) : int
+ OnRegistrationEvent(e : RegistrationEvent) : int
+ OnSubscriptionEvent(e : SubscriptionEvent) : int

Fig. 5. Class Diagram of Simple IMS Client

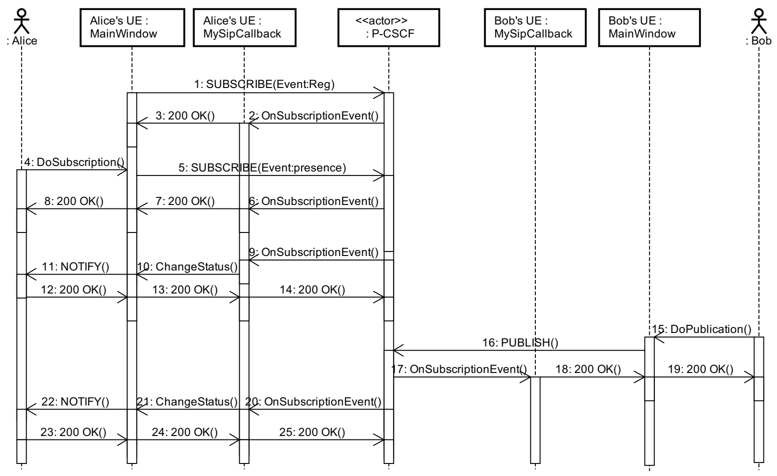

Fig. 7. Sequence Diagram of Presence Service in Simple IMS Client

The three teaching cases gradually build a standard IMS client with simplified model. It not only gives an introduction of mechanism of IMS core network, but also provides a practice method to implement a simple IMS client.

VI. Conclusion

In these teaching cases of simple IMS client, students can learn the mechanism of IMS gradually in a natural way. Although it is only a simplified model instead of a commercial model of IMS, but it perfectly focuses on the main structure in IMS and hides the sophisticated technical details in commercial IMS network, which is not only beneficial for the beginner of IMS, but also helpful for people who want to focus on the main point of IMS network.

Nowadays, IP Multimedia Subsystem begins to grow and becomes mature. IMS is integrating PSTN, Mobile network, and Internet worldwide. Thus, it's very important to spread the knowledge of IMS network. A series of teaching cases efficiently play a necessary part in popularizing knowledge of IMS network.

Acknowledgment

This project is sponsored by Beijing Education Commission.

References

[1] 3GPP, "Ip multimedia subsystem (ims); stage 2," 3rd Generation Partnership Project (3GPP), TS 23.228, Nov. 2012. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/23228.htm

[2] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "Sip: Session initiation protocol," Internet Engineering Task Force, RFC 3261, Jun. 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3261.txt

[3] F. YANG and Q. SUN, *Softswitch and IMS Technology*. Beijing University of Posts and Telecommunications Press, 2007.

[4] H. Sugano, S. Fujimoto, G. Klyne, A. Bateman, W. Carr, and J. Peterson, "Presence information data format (pidf)," Internet Engineering Task Force, RFC 3863, Aug. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3863.txt

[5] E. Inc. (2012, Nov.) Ip multimedia subsystem (ims) call flows. [Online]. Available: http://www.eventhelix.com/ims/

[6] M. Day, J. Rosenberg, and H. Sugano, "A model for presence and instant messaging," Internet Engineering Task Force, RFC 2778, Feb. 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2778.txt

[7] Microsoft. (2012, Nov.) Msdn library. [Online]. Available: http://msdn.microsoft.com/en-us/library/

[8] FOKUS. (2012, Nov.) Openimscore.org — the open source ims core project. [Online]. Available: http://www.openimscore.org/

[9] D. Telecom. (2012, Nov.) Doubango - open source 3gpp ims/lte framework for embedded systems. [Online]. Available: http://doubango.org/