Session Mobility of SIP over Multiple Devices

Min-Xiou Chen
Department of Computer Science and Information Engineering,
ChungHua University, Hsinchu, Taiwan, R.O.C.
mxchen@chu.edu.tw

Fu-Ju Wang
Department of Computer Science and Information Engineering,
ChungHua University, Hsinchu, Taiwan, R.O.C.
m9402046@chu.edu.tw

ABSTRACT
Session Initiation Protocol (SIP) has been widely applied to telephony service. SIP is also adopted as the signal communication protocol of VoIP and 3GPP, so the development of SIP receiving much concern. Currently the research of SIP can be divided into four issues, including terminal mobility, session mobility, personal mobility, and service mobility. In this paper, we address the issue of session mobility. We want to solve the problem allowing a user to transfer, split, and retrieve a session over multiple devices by using SIP. In order to provide the service, three new agents have been introduced based on the definition of user agent proposed in RFC 3261. There are session manager, session user, and free node. Based on these agents, we propose a complete mechanism, to transfer and retrieve a session over multiple devices. According to our mechanism, the user will facility transfers, splits, and retrieves a session over multiple devices. Finally, we implement the mechanism by modifying the open source project of "Sip-Communicator".

Keywords: Split session, Session Mobility, SIP, Open Source

1. INTRODUCTION
Recent advances in communication bandwidth have enabled the development of Internet applications, such as WWW, HTTP, FTP and Instant Message. Due to the rapid development of Internet applications, Internet services come into our daily life. Take instant message service as an example, people could only exchange their real time information through the telecommunication network in the past years. But now the Instant Message service can be used to carry multimedia service, such as voice, video or data, anywhere at any time with any devices. Thus, people can exchange their real time information with their friends through the Internet.

Due to the popular application of Internet, Voice over Internet Protocol (VoIP) had been proposed by the researchers in order to reduce the cost of the overseas call. Moreover, in order to carry multimedia service through the Internet, the Session Initiation Protocol (SIP) [1] had been introduced and adopted by the VoIP and 3GPP communities. SIP is an application layer signaling protocol for creating, modifying, and terminating sessions with one or more devices. It is designed to be independent of the underlying transport protocols. Thus, SIP can run on TCP, UDP or SCTP. Moreover, SIP can be used to establish one or multiple sessions that include multimedia streams, such as audio, video and text information. It widely used as a signaling protocol in home networks and personal area networks as discussed in [2-4].

Mobility management is an important issue of SIP. According to the descriptions in [5][6], the mobility management problems of SIP can be divided in two four classes. There are terminal mobility, personal mobility, session mobility and service mobility. The major objective of terminal mobility is to allow a device to move between IP subnets, while continuing to serve any incoming request and continuing to serve sessions across IP subnet changes. The ability of personal mobility is to support a user to access mobility services from anywhere, at anytime, using any device. The main function of service mobility is to provide a user to use the same service even when the user attaches the Internet at different device.

Session mobility is to maintain an ongoing media session from one device to another. Two approaches can be used to support session mobility using SIP. The first approach using the REFER method defined in RFC 3515 [7] to provide the Call Transfer mechanism, and the other one is using the Third Party Call Control (3PCC) mechanism defined in RFC 3725 [8]. Both these approaches can transfer an ongoing media session from one device to another, and can retrieve a transferred media session. However, these approaches cannot be used to transfer an ongoing media session to multiple devices.

The major purpose that transferring a session over multiple devices is to allow the user to transfer, split, and retrieve a session over multiple devices. Therefore, a user can create a session, which contains a video stream and an audio stream, with his friend at his mobile device. Then, when he backs home, he may transfer the video stream to his TV, but keep the audio stream at his mobile device due to the voice echo problem. Moreover, when he walks away, he can retrieve the video stream to his mobile device for continue communication.

In this paper, we address the issue of session mobility over multiple devices. We propose a complete mechanism to transfer and retrieve a session over multiple devices. In

Tridentcom 2008, March 18 – 20, 2008, Innsbruck, Austria.

order to provide session mobility, three new agents had been introduced in our mechanism. There are session manager, session user, and free node. Moreover, we also propose the session manager transformation (SMT) to provide much more flexibility in session mobility. Based on our mechanism, the user can transfer, split, and retrieve a session over multiple devices.

The remainder of this paper is organized as follows. The related literatures of session mobility are discussed in Section II. In Section III, our mechanism is proposed, as well as how the user can transfer and retrieve a session over multiple devices. The implementation architecture is described in Section IV. In Section V, we take an example to show the correctness of our architecture. Finally, the conclusion is given in Section VI.

2. RELATED WORKS

The basic idea of session mobility is to transfer an ongoing media session from one device to another. Two approaches can be used to transfer an ongoing media session from one device to another, and can retrieve a transferred media session. The first approach uses the REFER method defined in RFC 3515 [7], and the other one uses the Third Party Call Control (3PCC) mechanism defined in RFC 3725 [8].

REFER is a SIP extension method, and can be used to enable many applications. The REFER request contains the contract information, which is included in the new defined header field "Refer-To". The contract information also indicates a target device. Thus, the function of the subscription is to keep track of the progress of the REFER request. This means that a REFER request implicitly establishes a subscription to the refer event. In REFER method, all the devices which receive the SIP signals must be the sender or the receiver of a media session.

Contrary to the REFER method, 3PCC allows a device, which may not be a media session sender or a media session receive, to setup and manage a communications relationship between two or more other devices. There are four kinds of call flows in the 3PCC mechanism, and all of them have benefits and drawbacks. All of these call flows can be used to transfer an ongoing media session from one device to another.

However, both these mechanisms cannot be used to transfer or split an ongoing session service over multiple devices. In the First, the REFER method cannot indicate which certain media sessions need to transfer. Moreover, the device has no ability to discern between a request of split session and a request of new session. There are two main drawbacks to the 3PCC control mechanism for session mobility. First, the original session device needs to remain involved in the session, as it will be contacted to change or terminate the session. Second, the 3PCC control mechanism needs a central point of control which may not be desirable in many environments.

In order to support to transfer or split an ongoing session service over multiple devices, we proposed a method, refers to as "Split a SIP session over multiple devices (SSIP)" [9]. We introduced some improvements for the user agent to obtain the ability to split a session. The extension header "Mobility" is used to improve the REFER method and make it transparent to the remote party. The concept of "Association" is used to solve the problem of the user having to terminate all devices separately when a session was split over multiple devices. Moreover, we also implemented a SIP user agent with the SSIP to split a session over multiple devices. However, the session retrieval mechanism was not implemented in SSIP.

Shacham, et al. proposed an integration mechanisms to provide session mobility over multiple devices [10]. Their mechanisms involves the REFER method and the 3PCC control mechanism. In order to efficiently manage the transferred session between multiple devices, the concept of "Virtual Device" was introduced. In the first, all the candidate devices are grouped into a set of "Virtual Device". One of the devices in the Virtual Device will be set as coordinator, refers to as Multi-Device System Manager (MDSM). When a session will be transferred from the user agent device to different devices which are in the same Virtual Device, the user agent will issue a REFER message and send this message to the MDSM. Then, the MDSM uses 3PCC to invite each local device and set up media sessions between correspond node and each device in Virtual Device. Thus, the media sessions between the device and the correspondent node will be set up based on the 3PCC control mechanism.

This mechanism has the advantages of the 3PCC mechanism and the REFER method. However, there are some problems with this mechanism. In the first, because that only the original user agent can retrieve the split session, the split session can only be combined into the original user agent. This means that although there are many devices in the Virtual Device, the split session cannot be combined into any one of the device in the Virtual Device. This problem restricts the flexibility of session mobility. Moreover, in order to efficiently manage the signal messages, the MDSN was introduced into their mechanism. The MDSN is not included in the standard of SIP.

3. SESSION MOBILITY OVER MULTIPLE DEVICES

3.1 System Architecture

In the first, an ongoing session contains a set of media streams. For instance, a user may create a session with a video stream and an audio stream with his friend. Then, we define some common components which will be used in the section. There are four logical network nodes defined in the standard of SIP. There are user agents (UA), registrars, proxy servers, and redirect servers. The major functions of these network nodes can be found in [1]. Six basic requests are also defined in the standard of SIP. There are INVITE, ACK, BYE, REGISTER, OPTION and REGISTER. The function of INVITE and ACK are used to initiate a new session and to confirm a session establishment request (INVITE request), respectively. The REGISTER request is used to register location information into the registrars. The ability of OPTION is used to determine requested capabilities of a session. The ability of BYE is used to terminate a session that has been established. Finally, the CANCEL request is to terminate a session that has not been established yet. All the detail definition of SIP can be found in the standards [1].

In order to provide session mobility, three new logic states are introduced in our mechanism. All of these logic states are the user agent, and there are session manager, session user, and free node. Before we describe the definition and the function of these logic components, we introduce the concept of partial session in the first. The concept of partial session is that one or more media streams of a session are on a user agent, and the others are on the different user agents. As shown in Figure 1, node A, node B and node C are the user agents. Suppose that there is a session with video and audio stream between node A and node C. Then, node A splits the audio stream from the session, and transfer the audio stream to node B. Now, both node A and node B keep the partial session.

The user agents which keep no any partial session are the free node, and the user agent which has partial sessions are the session manager, or session user. The difference between a session manager and session user is that the session manager will has all the information of the partial session, such as configuration, security, and authorization, and the session user just only has the partial session on itself. Thus, based on the information, all the partial session can be transferred, retrieved, and terminated by the session manager. The session user only can transfer itself partial session to the session manager. After transferring the partial session, the updated information should be send to the session manager.

3.2 Protocols

In [9], The SSIP method had been proposed to split a session over multiple devices. However, the session retrieval mechanism was not implemented in SSIP. In this paper, we will design the session mobility mechanism based on the architecture of SSIP. First, we should implement the session retrieval mechanism in our mechanism. The Nested REFER method proposed in RFC 3892 [11] was introduced in order to solve the session retrieval mechanism. The concept of Nested REFER is that the Refer-To URI is a SIP URI indicating the REFER method. This means that the Nested REFER method can be used to request another REFER. Thus, the a user agent A can send a Nested REFER to a user agent B to retrieve the partial session split from A's partial session.

Take the Figure 2 as an example, node A is the session manager, and node B is the session user. Suppose that node A wants to retrieve the partial session on node B, node A will send a Nested REFER to node B. The message format of the Nested REFER from node A to node B is composed as follows:

Refer-To: <sip:A.example;method=REFER?Refer-To="<sip:C.example>">

When node B receives the Nested REFER, node B will send a REFER request to node A, and the Refer-To is set to "sip:C.example". Then, node A will send the INVITE request to node C to retrieve the partial session between node B and node C. The retrieved partial session can be identified by the Mobility header proposed in SSIP. Therefore, the session retrieval mechanism can be solved based on the Nested REFER method and SSIP method.

Although the session retrieval mechanism can be provided by the Nested REFER method and SSIP method, the split session can only be combined into the original user agent. This solution restricts the flexibility of session mobility. Thus the concept of session manager was introduced to provide much more flexibility in session mobility. From the definition of session manager described in previous session, we can know that the session manager will have all the information of the partial session, such as configuration, security, and authorization. This means that a session manager has enough information to send a Nested REFER to any session user to retrieve the partial session. Moreover, suppose the information of session manager can be transferred from the user anger to any other user agent, the partial session can be combined into the user agent, which is at the session manager state.

It is obvious that the session manager has strong control power for session mobility. According to the need for ease of operation and a concern for security, we propose two operation modes. There are the centralized

operation mode and the distributed operation mode. In the centralized operation mode, there is only one user agent in the session manager state. The centralized operation mode is very suitable for the security risk environment. In contrast to the centralized operation mode, in the distributed operation mode, all the user agents which keep partial session are in the session manager state. It means that in the distributed operation mode, all the user agents either are a session manager, or are a free node.

We will introduce the centralized operation mode in the first. In order to provide the flexibility of session mobility, we propose one method, refers to as "session manager transformation (SMT)", used to support the session manager state transformation between different user agents. The ability of the SMT method is to provide a session manager to transfer all the session information to a session user or a free node. After SMT, the session user will become the session manager, and the original session manager will become a session user or a free node.

As shown in Figure 3 and in Figure 4. In the first, the session manager will select a user agent (session user or free node) as a target user agent, and send the SMT message to the target user agent. The authentication information will also be added into the SMT message. As shown in Figure 4, the target user agent will check the authentication information, and will check its own state. If the target user agent is already in session manager state of another session or the SMT message is illegal, the SMT request will be rejected. Otherwise, the target user agent can accept the SMT request or not. If the target user agent accepts the SMT request, an accept message will be responded to the original session manager. Then, the session manager will send the session information to the target user agent, and inform all session user about the new session manager. Finally, the original session manager will become a session user or a free node, and the target user agent will become the new session manager.

Contrary to the centralized operation mode, there is no user agent in session user state in the distributed operation mode. This means that all user agent which keeps the partial session will be in session manager state. There are some differences in SMT method. The old session manager must inform all other session manager about the new session manager. Moreover, after splitting session, the session information should be send from the session manager to the split user agent. This means that, after a session transfer, or a session split, the session information should be send from the session manager to all other session manager Contrast to the session transfer and a session split, when the partial session retrieve to a session manager, the user agent which keeps the original partial session will become free node, and its session information will be released.

4. IMPLEMENTATION

Lately some SIP projects have been proposed. According to these projects and their SIP stacks, the user agent and many SIP applications can be implemented. In the paper, we chose the JAIN-SIP, NIST-SIP and SIP Communicator as the fundamental architecture for our implementation. These softwares are all open sources and are implemented by Java. We will briefly introduce these softwares.

a. Java APIs for Integrated Networks (JAIN)-SIP

The JAIN-SIP Specification [12] as defined by the JAIN Protocol Expert Group Java Community Process Participants for SIP is based on the RFC 3261 [1]. JAIN-SIP is a low level protocol API for SIP. Many interfaces of SIP API have been defined in JAIN-SIP. These APIs allow for the rapid creation and deployment of dynamic telephony services into a Java telephony platform. Thus, based on the JAIN-SIP, the user can rapidly develop, test, and integrate numerous services on the JAVA platform.

b. NIST-SIP

NIST-SIP [13] is a project proposed by the National Institute of Standards and Technology (NIST). NIST-SIP contains a SIP protocol stack and a number of libraries. A number of tools including the official Reference Implementation of the JAIN-SIP, and an implementation of the JAIN-SDP are implemented in NIST-SIP. The goal of NIST-SIP is to facilitate the development of improved VoIP transport mechanisms and to expedite the development of programmable telephony services. Thus, based on NIST-SIP, the user can rapidly build SIP applications and servers

c. SIP communicator

SIP communicator [14] also is a Java based SIP User Agent. It builds on top of the JAIN-SIP-RI and JMF (Java Media Framework). The SIP communicator provides audio and video sessions over IPv4 and IPv6 network. SIP Communicator is completely Open Source and Free Software. It is freely available under the terms of the GNU Lesser General Public License. It started out as the JsPhone example in the NIST-SIP project, but recently is started a life of its own as a separate project on java.net.

In the paper, our implementation is based on the architecture of SSIP [9]. However, the session retrieval mechanism was not implemented in SSIP. Thus, we introduced the Nested REFER method proposed in RFC 3892 [11] to implement the session retrieval mechanism in our mechanism. In order to provide session retrieval

mechanism, we add the "ReferredByHeader" interface into the javax.sip.header package, and implement the ReferredBy method in the NIST-SIP package. Moreover, we also implement the Nested REFER functions into the SipManager. Thus, the session retrieval mechanism can be support in the SIP communicator. The procedures of Nest REFER were shown in Figure 5 and Figure 6.

The concepts of session manager, session user, and free node were also implemented in the implementation. Moreover, in order to provide SMT, the INVITE procedure of CallProcessing should be modified. Therefore, the information of session manager can be inserted into the message body. After SMT, the state of user agent will be shown in the rule table.

Our implementation, as shown in Figure 7, is a SIP UA. The media panel is used to show the video stream. In the Call menu, we can enter the SIP URI of a callee in the contact box and push the dial button. Then, the SIP UA can establish a session and show the session information in the table below the media panel. The device of the notebook can be shown in the Device menu. All the functions are provided in the SIP UA.

5. IMPLEMENTATION RESULTS

In order to ensure the ability of our mechanism, we built an experimental environment as shown in Figure 8. We prepared four notebooks to play the roles of session manager, session user, and free node. All of these notebooks are equipped with a webcam and a microphone. The node A and node B belong to one domain and the others belong to another domain. There are two SIP proxy servers in our experimental environment, but are not shown in Figure 8.

As shown in Figure 8, node B and node C create a session with an audio stream and a video stream. At that time, both node B and node C are the session managers, and both node A and node D are the free nodes. Then, node C splits the partial session of audio stream to node D, and node D is the session user after session splitting. Figure 9 shows the packet traffic of the implementation. From the Figure 9, we can find that node C splits the partial session of audio stream to node D, and the traffic of audio stream is empty after 12 seconds. We also can see the traffic of audio stream between node B and node D is increased after 12 seconds. At 25 seconds, node B splits the partial session of audio stream to node A, and the traffic of audio stream between node B and node D is empty. Then, at 32 seconds, node B retrieves the partial session at node A. Thus, the traffic of audio stream between node B and node D is increased, and at that time, node A is the free node. Finally, After 42 seconds, node C retrieves the partial session at node D, the traffic of audio stream between node B and node C is increased, and node D is the free node.

6. CONCLUSIONS AND FUTURE WORKS

SIP session mobility over multiple devices becomes more important in the near future. In this paper, we proposed our mechanism based on the architecture of SSIP. According to our mechanism, the user can transfers, splits, and retrieves a session over multiple devices. Finally, we implement the mechanism by modifying the open source project of "Sip-Communicator". We also built an experimental environment in order to ensure the ability of our mechanism. Since JAIN-SIP, NIST-SIP and SIP Communicator are based on J2SE, our implementation does not support mobile devices with limited ability, such as PDA or Smart phone. Therefore, we plan to investigate how to port the JAIN-SIP, NIST-SIP and SIP Communicator to the J2ME, and our implementation will also be ported to J2ME in the near future.

7. ACKNOWLEDGMENTS

The work was supported by the National Science Council of Taiwan, R.O.C. under Grant NSC95-2221-E-216-039 and NSC96-2221-E-216-010, and supported by the ChungHua University under Grant CHU95-2221-E-216-039 and CHU96-2221-E-216-010.

8. REFERENCES

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "SIP: Session Initiation Protocol," RFC 3261, IETF, Jun 2002.

[2] S. Moyer, D. Maples, S. Tsang, and A. Ghosh, "Service Portability of Networked Appliances," IEEE Communications Magazine, Vol. 40, No. 1, Jan. 2002

[3] S. Moyer, D. Maples, and S. Tsang, "A Protocol for Wide-Area Secure Networked Appliance Communication," IEEE Communications Magazine, Vol. 39, No. 10, Oct. 2001.

[4] J. Latvakoski, P. Paakkonen, D. Pakkala, A. Tikkala, J. Remes, and P. Valitalo, "Interaction of All IP Mobile Internet Devices with Networked Appliance in a Residential Home," 22nd International Conference on Distributed Computing Systems Workshops, 2-5 July 2002.

[5] E. Wedlund and H. Schulzrinne, "Mobility Support using SIP," in Second ACM/IEEE International Conference on Wireless and Mobile Multimedia (WoWMoM' 99), Aug. 1999.

[6] H. Schulzrinne and E. Wedlund, "Application-Layer Mobility Using SIP," IEEE Service Portability and Virtual Customer Environments, Dec. 2000.

[7] R. Sparks, "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, IETF, April 2003.

[8] J. Rosenberg, J. Peterson, H. Schulzrinne, G. Camarillo, "Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)", RFC 3725, IETF, April 2004.

[9] Min-Xiou Chen, Chen-Jui Peng and Ren-Hung Hwang, "SSIP: Split a SIP Session over Multiple Devices," Computer Standards and Interfaces. Vol. 29, No. 5 pp. 531-545, July, 2007.

[10] R. Shacham , H. Schulzrinne , S. Thakolsri and W. Kellerer , "Session Initiation Protocol(SIP) Session Mobility draft-shacham-sipping-session-mobility-04", IETF , July 2007.

[11] R. Sparks, "The Session Initiation Protocol (SIP) Referred-By Mechanism",RFC 3892 ,IETF, Sep 2004.

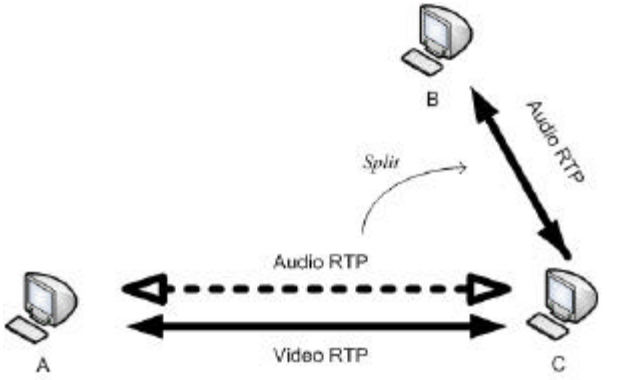

Figure 1. The Concept of Partial Session

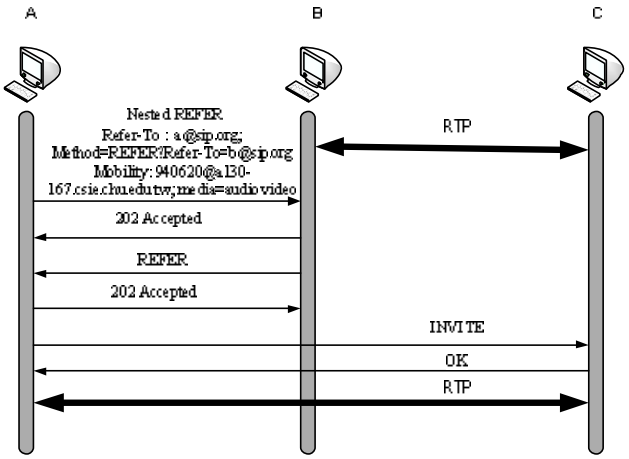

Figure 2. Session Retrieval Mechanism

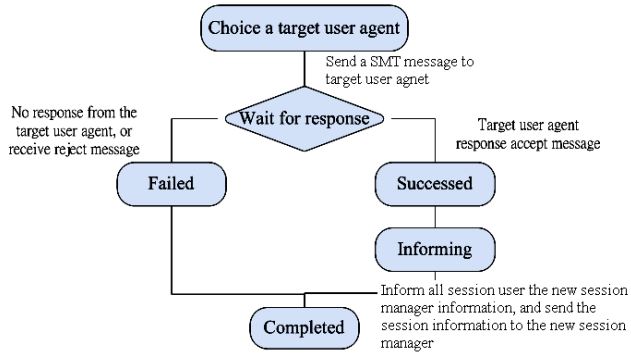

Figure 3. The Flowchart of SMT at Session Manager

[12] JCP, "JSR-32: JAIN SIP Specification," http://www.jcp.org/en/jsr/detail?id=32

[13] NIST, "NIST-SIP 1.2 – SIP Libraries and Tools for the People," http://www-x.antd.nist.gov/proj/iptel/

[14] Network Research Team, "SIP-Communicator A Java Softphone based on JAIN SIP with audio/video and instant," https://sip-communicator.dev.java.net/

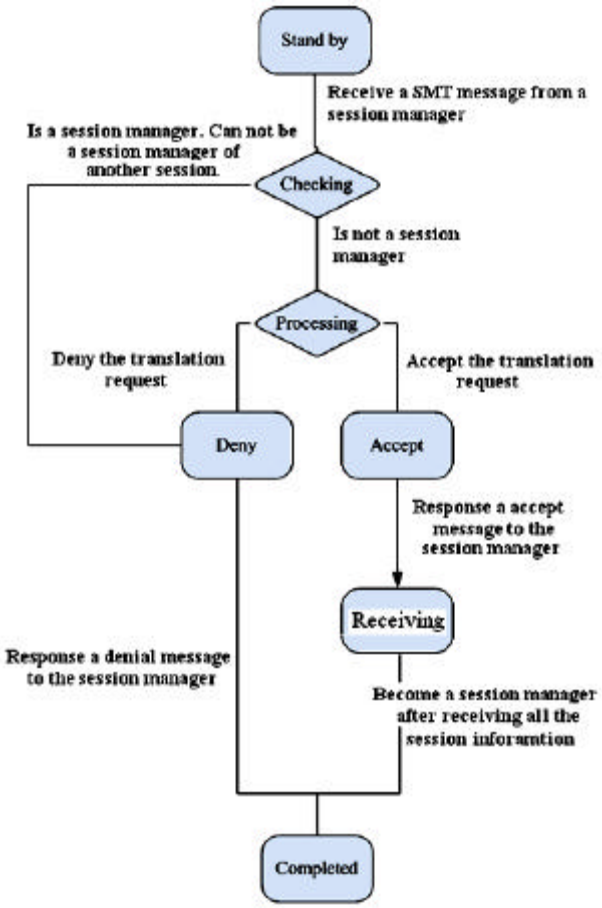

Figure 4. The Flowchart of SMT at Session User or Free Node

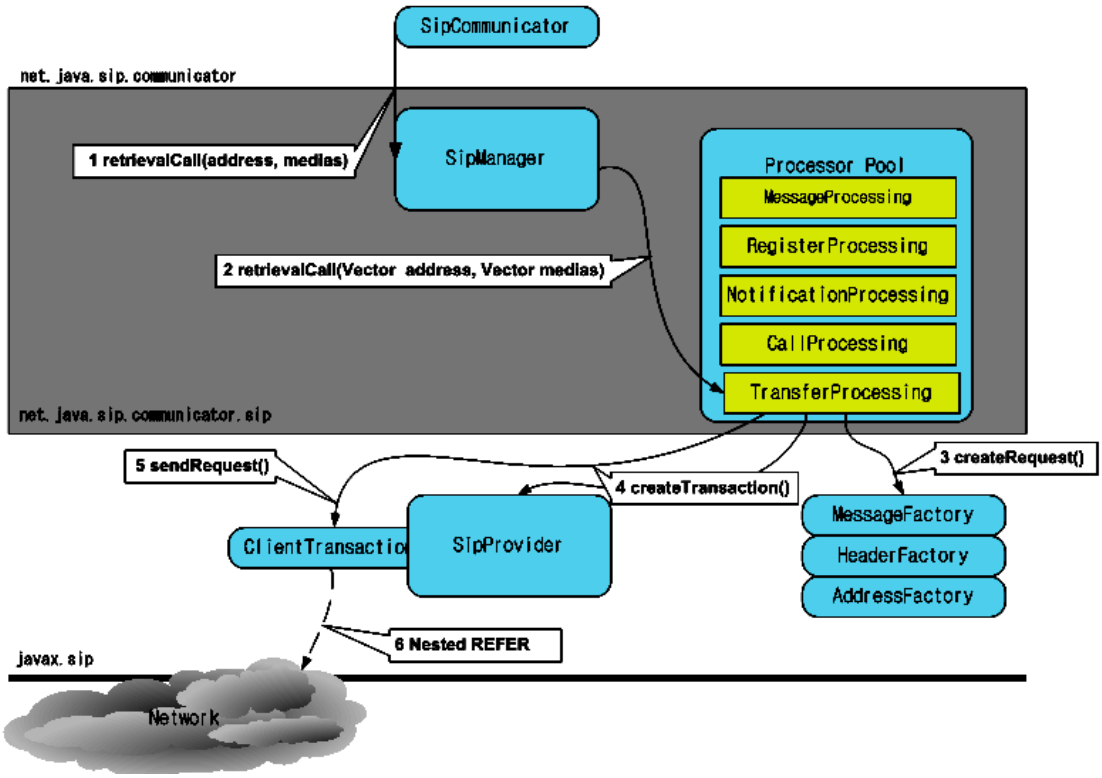

Figure 5. Nested REFER Procedure of Client side

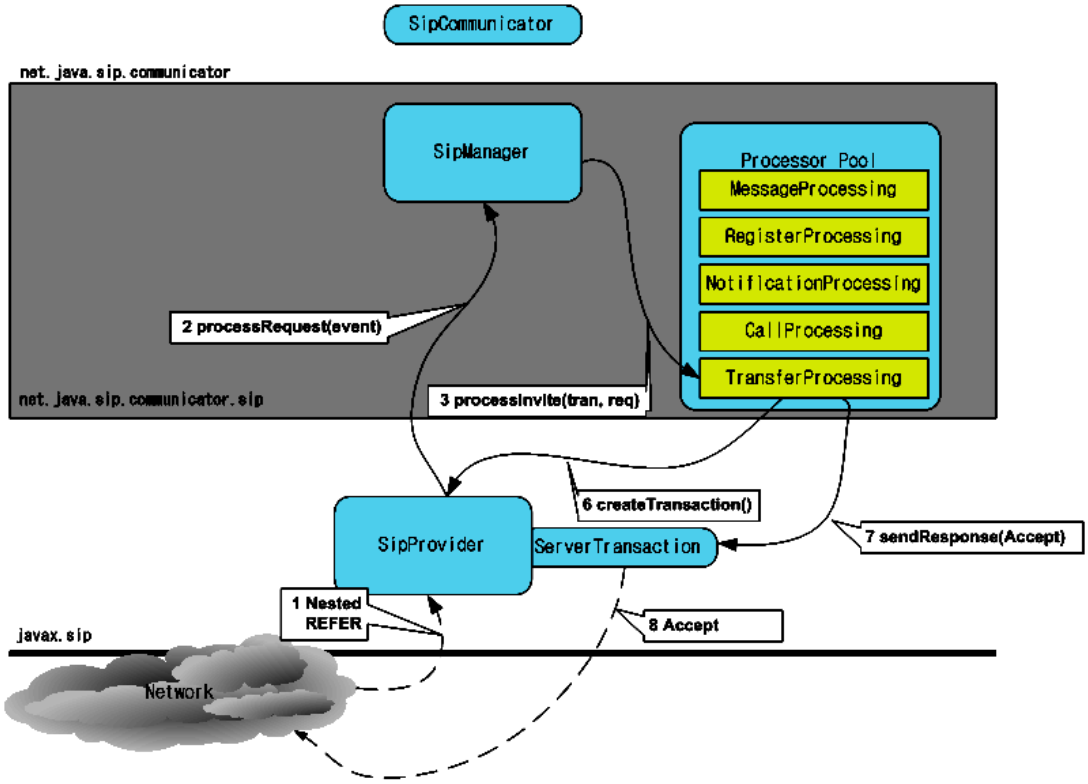

Figure 6. Nested REFER Procedure of Server Side

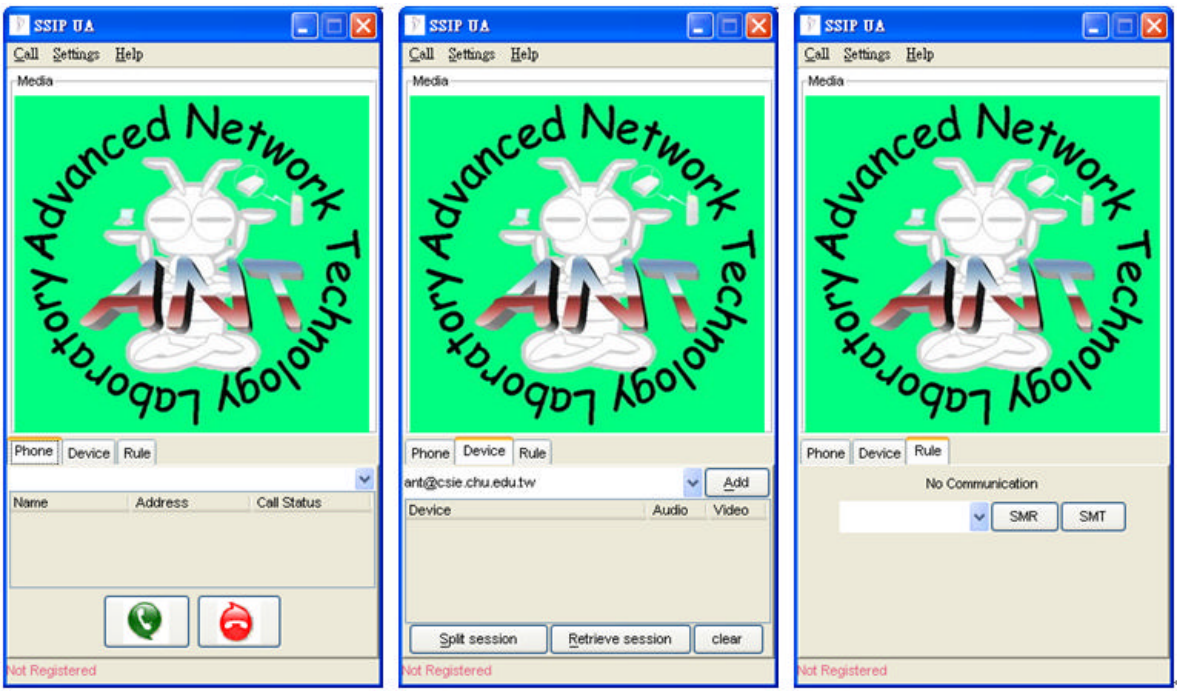

Figure 7. The Interface of User Agent

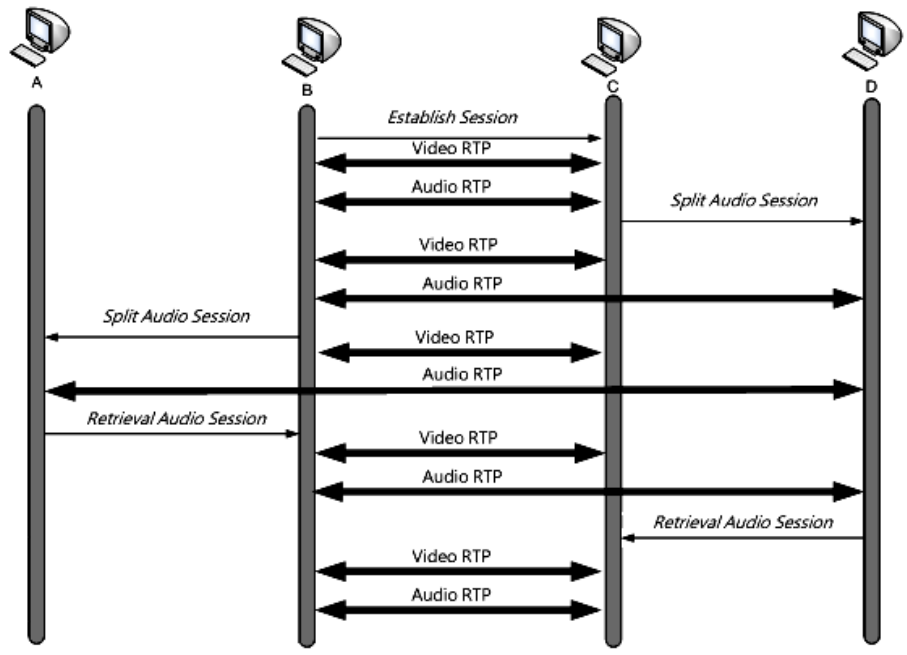

Figure 8. Experimental Environment

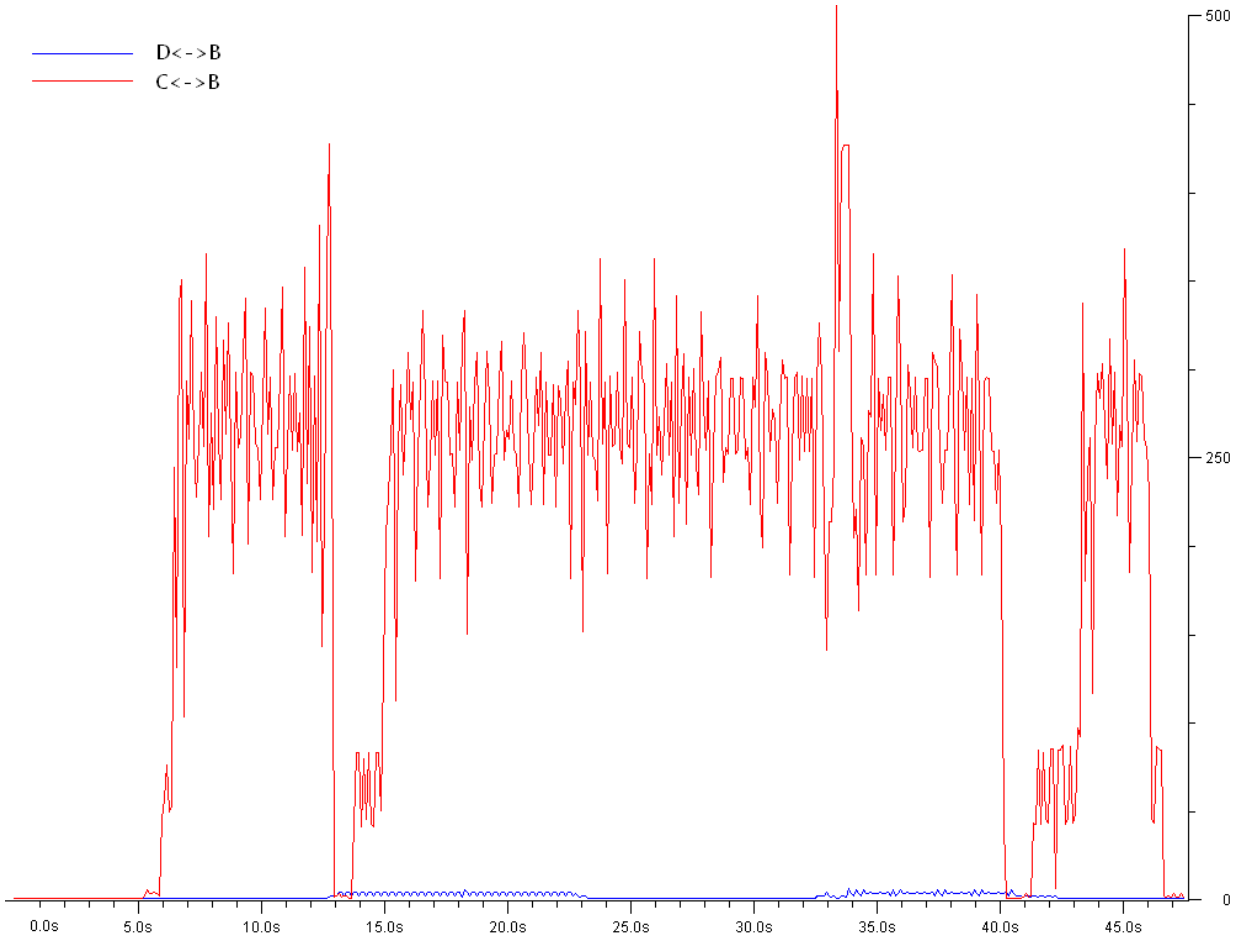

Figure 9. Packet Traffic of Implementation Results