WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2008; **8**:245–253
Published online 25 September 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/wcm.458

SIPv6 analyzer: an analysis tool for 3GPP IMS services

Whai-En Chen, Yueh-Hsin Sung and Yi-Bing Lin*,†

Department of Computer Science and Information Engineering, National Chiao Tung University, Taiwan, Republic of China

Summary

The *3rd Generation Partnership Project* (3GPP) defines *IP Multimedia Core Network Subsystem* (IMS) to support IP-based multimedia services. IMS utilizes protocols such as *IP version 6* (IPv6), *Session Initiation Protocol* (SIP), and *Real-time Transport Protocol* (RTP) to deliver the multimedia content. This paper proposes an analysis tool referred to as SIPv6 Analyzer to investigate the IMS-related protocols during IMS service deployment. The SIPv6 Analyzer not only dissects the protocol headers but also provides user-friendly functions such as message flow generation and audio/video replay for IPv6, SIP, and RTP. We describe the design of the SIPv6 Analyzer and use examples to illustrate how it works. Copyright © 2006 John Wiley & Sons, Ltd.

KEY WORDS: 3GPP; IMS; IPv6; SIP; RTP

1. Introduction

The *3rd Generation Partnership Project* (3GPP) defines *IP Multimedia Core Network Subsystem* (IMS) to support IP-based multimedia services [1]. 3GPP IMS utilizes *Session Initiation Protocol* (SIP) [2] and *Real-time Transport Protocol* (RTP) [3] to transfer the *Voice over IP* (VoIP) signaling and multimedia information. *IP version 6* (IPv6) is also employed in IMS to provide large address space and new features not found in IPv4, including security, *Quality of Service* (QoS), and Plug-and-play [4].

During IMS service deployment, it is essential to utilize analysis tools for debugging and investigation of the IMS-related protocols. A potential analysis tool is Ethereal [5], an open-source analyzer which can dissect more than 700 protocol headers including IPv6, SIP, and RTP. However, Ethereal has the following disadvantages: (1) Ethereal only provides limited SIP message flow illustration capability (see Section 4 for a complete illustration supported in our solution). (2) Ethereal does not provide audio/video replay for IMS services. (3) Ethereal cannot automatically retrieve SIP messages that are not delivered in default port number 5060.

To support these functions for IMS service investigation, this paper develops a tool called *SIPv6 Analyzer*. This tool provides the following functions.

- The SIP dialog list function organizes the related SIP messages in the same SIP dialogs. For an SIP dialog, the SIP message flow generation function plots message flow and therefore provides effective message path tracking for IMS multimedia calls.
- The audio/video replay function organizes the RTP packets into the RTP connections and plays the audio or the video delivered in these RTP connections. Through this function, the users can evaluate

*Correspondence to: Yi-Bing Lin, Department of Computer Science, National Chiao Tung University, 1001 Ta Hsueh Rd., Hsinchu, Taiwan 300, Republic of China.
†E-mail: liny@cs.nctu.edu.tw

Copyright © 2006 John Wiley & Sons, Ltd.

the quality of the captured IMS calls. Moreover, the audio/video replay function can be used to wiretap the IMS calls for lawful interception.
- The statistic function collects the packet counts, the inter-arrival distributions, and the jitter statistics for the IMS protocols.

This paper is organized as follows. The system architecture of the SIPv6 Analyzer is described in Section 2. Section 3 presents the design and operation of the key component (i.e., the SIP/RTP Processor) of the SIPv6 Analyzer. Section 4 demonstrates the SIP message flow and RTP replay functions through an IMS call example.

2. System Architecture

Figure 1 illustrates the SIPv6 Analyzer architecture that consists of three major modules. The *WinPcap Module* (Figure 1(1)) utilizes the open-source *Windows Packet Capture* (WinPcap) library [1] to capture the packets in the kernel level. The *Packet Processing Module* (Figure 1(2)) obtains the packets from the kernel-level WinPcap Module, classifies the packets, and provides an interface for the user-level applications to analyze the captured packets. The *User Interface* (UI) Module (Figure 1(3)) presents the captured packet information through dialog/session list, message flow, audio, and video.

2.1. The Packet Processing Module

The Packet Processing Module consists of five components. Upon receipt of a packet from the WinPcap Module, the *Packet Preprocessor* (Figure 1ⓑ) retrieves the IP address and the port number information from the headers and stores this information together with the original packets into the *Packet Buffer* (Figure 1ⓐ). The Packet Preprocessor maintains a *Filtering Rule Table* (Figure 1ⓒ) and invokes the WinPcap Module to filter the specific packets based on the filtering rules. For a captured packet, the Packet Preprocessor forwards the packet pointer to the *Ethereal Packet Processor* (Figure 1ⓓ) and *SIP/RTP Processor* (Figure 1ⓔ). The Ethereal library [5] is utilized to implement the Ethereal Packet Processor for packet header dissection. The SIP/RTP Processor retrieves the SIP and the RTP packets from the captured IP packets and organizes them into the corresponding sessions. The *Statistic Processor* (Figure 1ⓕ) collects the packet counts, the inter-arrival distributions, and the jitter statistics of the captured packets.

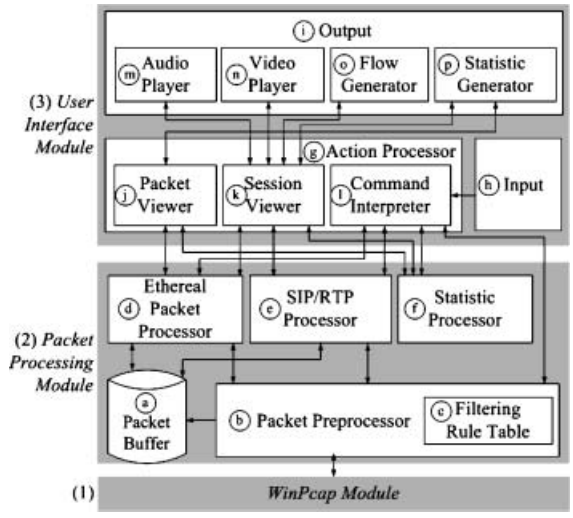

Fig. 1. System architecture of the SIPv6 Analyzer.

Most analyzers (e.g., Ethereal and Sniffer) set the filters 'udp port 5060' and 'udp port 9000' to capture the SIP and the RTP packets, respectively. However, a SIP User Agent (UA; i.e., an IMS user equipment) may use TCP to transmit SIP messages. The SIP UA may also utilize an arbitrary port number for SIP message delivery. Therefore, if the fixed-port filter (i.e., udp port 5060) is used to capture the SIP messages, the SIP messages carried by TCP and/or flexible port numbers will be ignored and 'wrong packets' may be captured for IMS service analysis.

The above issue can be resolved as follows. We note that the first portion of a SIP message is the request-line or the status-line. The last word in the request-line and the first word in the status-line are the ASCII keyword 'SIP/2.0' that is used to identify the SIP version. To prevent mis-filtering any of the SIP messages, the SIP/RTP Processor checks the TCP or UDP payload to see if the first line contains the keyword. If so, an SIP message is identified.

Similarly, setting the fixed filters (e.g., udp port 9000) may capture non-RTP packets that use port number 9000 or ignore the RTP packets that do not use port number 9000. We resolve this issue as follows. Since the RTP connection information is exchanged through the SIP messages, the SIP/RTP Processor retrieves the correct IP address and port number information of the RTP packets from the *Session Description Protocol* (SDP) [7] *c* and *m* fields in a SIP message. Then the SIP/RTP Processor can accurately recognize the RTP packets based on this information. After a RTP packet is identified, the SIP/RTP Processor organizes it into the corresponding

Copyright © 2006 John Wiley & Sons, Ltd.

RTP connection based on the SDP fields and the *Synchronization Source* (SSRC) value [3].

2.2. The User Interface Module

The UI Module consists of three components. The *Input Component* (Figure 1ⓗ) receives the instructions from the user and requests the *Action Processor* (Figure 1ⓖ) to execute the instructions. In the Action Processor, the *Command Interpreter* (Figure 1ⓛ) invokes the Ethereal Packet Processor, SIP/RTP Processor, Statistic Processor, or Packet Preprocessor to execute the instructions from the Input Component. After a command execution, the Action Processor may obtain the packet and session information through the *Packet Viewer* (Figure 1ⓙ) and the *Session Viewer* (Figure 1ⓚ). It then invokes the *Output Component* (Figure 1ⓘ) to present the results through Windows-based multimedia tools. For example, the *Audio Player* (Figure 1ⓜ) and *Video Player* (Figure 1ⓝ) replay the RTP voice and the video. The *Flow Generator* (Figure 1ⓞ) plots the SIP message flows based on the IP addresses or the SIP header fields. The *Statistic Generator* (Figure 1ⓟ) displays the packet related statistics by using tables, tree views, and bar charts.

Through the above components, the UI Module provides the following convenient functions.

- Since the captured SIP messages of various sessions are usually interleaved, the UI Module obtains the SIP dialogs form the SIP/RTP Processor and lists them into a table (Figure 6ⓐ). Through this function, a user can conveniently observe the sorted SIP messages in a SIP dialog instead of the interleaved messages.
- For each SIP dialog, the UI Module can generate a SIP message flow (Figure 6ⓑ) based on the IP addresses or the SIP header fields. Through this function, a user can identify all network nodes that are visited in the SIP signaling path.
- The UI Module supports the H.263 video codec and the audio codecs including G.711u, G.711a, and GSM. Therefore, one can replay the audio or the video carried in the selected RTP connection (Figure 6ⓒ).

3. The SIP/RTP Processor

This section describes the SIP/RTP Processor, which is the major component of the SIPv6 Analyzer.

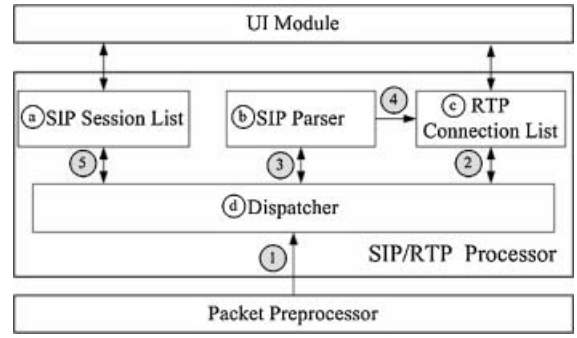

Fig. 2. The SIP/RTP Processor.

As illustrated in Figure 2, the SIP/RTP Processor consists of a *SIP Session List* (Figure 2ⓐ), a *SIP Parser* (Figure 2ⓑ), a *RTP Connection List* (Figure 2ⓒ), and a *Dispatcher* (Figure 2ⓓ).

The SIP Session List is a linked-list structure that stores the SIP session information (see Figure 3). Each entry in the SIP Session List includes a Session Identifier (Figure 3①) to indicate the SIP session and a linked list which stores the pointers to the packet associated with that session (Figure 3③–⑦). The Session Identifier of a SIP session consists of the SIP *From*, *To*, and *Call-ID* header fields.

An SIP call setup transaction consists of an *INVITE* message (Figure 3③), zero or more provisional messages (e.g., *100 Trying* message and *180 Ringing* message; see Figure 3④ and ⑤), a *200 OK* message (Figure 3⑥), and an *ACK* message (Figure 3⑦). In this transaction, both the *INVITE* and the *200 OK* messages contain the destination IP address and port number information in the SDP *c* and *m* fields. When the call setup transaction is complete, the RTP connections between the calling party and the called party are established.

An RTP session consists of two one-way connections. The RTP Connection List (Figure 4(a)) maps the RTP packets to the corresponding RTP connections. Each connection entry in the RTP Connection List includes a source (SRC) IP field, a destination (DST) IP field, an SRC Port field, a DST Port field, a SSRC field, and a linked list which stores the packet pointers belonging to the corresponding RTP

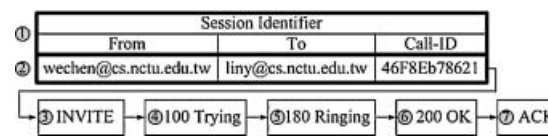

Fig. 3. A SIP Session List example.

Copyright © 2006 John Wiley & Sons, Ltd.

	SRC IP	DST IP	SRC Port	DST Port	SSRC	
①	3ffe:3600:1::1	3ffe:3600:1::2	9002	9000	29696	⑤ RTP
②	3ffe:3600:1::2	3ffe:3600:1::1	9002	9000	22607	⑥ RTP
③	3ffe:3600:1::3	3ffe:3600:1::4	9002	9000	25002	RTP
④	3ffe:3600:1::4	3ffe:3600:1::3	9000	9002	23141	RTP

(a) The RTP Connection List After Receiving the RTP Packets

	SRC IP	DST IP	SRC Port	DST Port	SSRC
①	3ffe:3600:1::1	3ffe:3600:1::2	*	9000	*
②	3ffe:3600:1::2	3ffe:3600:1::1	*	9000	*

(b) The RTP Connection List After Processing of SIP 200 OK Message

Fig. 4. A RTP Connection List example; (a) The RTP Connection List after receiving the RTP Packets; (b) The RTP Connection List after processing of SIP 200 OK Message.

connection. The SRC IP, the DST IP, and the DST port fields are used to check if a packet is for RTP, and all above five fields are used to classify the RTP packets into their corresponding connections.

Two RTP transmission methods, *asymmetric* and *symmetric*, are defined in Internet-Drafts [8]. In Figure 4(a), Entries 1 and 2 represent the pair of two connections for an asymmetric RTP session, and Entries 3 and 4 represent the pair of two connections for a symmetric RTP session. In an asymmetric connection, an SIP UA uses a port (e.g., 9000) to receive RTP packets and another port (e.g., 9002) to send RTP packets. In a symmetric connection, a SIP UA uses the same port (e.g., 9000) to send and receive the RTP packets.

The SIP Parser (Figure 2ⓑ) cannot determine whether a connection is asymmetric or symmetric based on the SIP messages. Therefore, before the first packet of a RTP connection is captured, the SIP Parser inserts a void '*' mark into the SRC Port fields (see Figure 4(b), Entries 1 and 2) of the RTP Connection List. The '*' mark represents that the source port number is not specified and should be retrieved from the first captured packet of the RTP connection. Since SIP messages do not contain the SSRC value, the SSRC field is also filled by '*'. After receiving the first RTP packet of a connection, the Dispatcher fills these values into the SRC Port and the SSRC fields.

Consider the following SIP/RTP packet handling example. Assume that the IP address of the calling party (i.e., wechen@cs.nctu.edu.tw) is 3ffe:3600:1::1, and that of the called party (i.e., liny@cs.nctu.edu.tw) is 3ffe:3600:1::2. The calling party and the called party use port number 5060 to send and receive SIP messages. Assume that the calling party and the called party use port number 9000 to receive RTP packets and port number 9002 to send RTP packets. After initialization of the SIP/RTP Processor, the SIP Session List (Figure 2ⓐ) and the RTP Connection List (Figure 2ⓒ) are empty. The *INVITE* message handling is described as follows.

Step 1.1. (Figure 2①) Upon receipt of the *INVITE* message, the Packet Preprocessor forwards the packet pointer to the Dispatcher.

Step 1.2. (Figure 2②) The Dispatcher retrieves the source IP address (3ffe:3600:1::1), the destination IP address (3ffe:3600:1::2), the source port number (5060), and the destination port number (5060) from the IP and UDP headers of the packet. This information is used to search the RTP Connection List. The Dispatcher does not find any matched entry, which implies that this packet is not an RTP packet.

Step 1.3. (Figure 2③) The Dispatcher forwards the packet pointer to the SIP Parser to verify that this packet carries an SIP message (i.e., an *INVITE* message). The SIP Parser then generates a session identifier by combining the *From* (i.e., wechen @cs.nctu.edu.tw), *To* (i.e., liny@cs.nctu.edu. tw), and *Call-ID* (i.e., 46F8Eb78621) header fields. To obtain the RTP connection information of the calling party, the SIP Parser retrieves the IP address 3ffe:3600:1::1 from the SDP *c* field and the port number 9000 from the SDP *m* field. At this point, the call setup transaction is not complete. The SIP Parser keeps the IP address and port number information in a temporary buffer. This information will be stored in the RTP Connection List at Step 2.4.

Step 1.4. (Figure 2③) The SIP Parser returns the session identifier information (i.e., {wechen@cs.nctu. edu.tw}{liny@cs.nctu.edu.tw}{46F8Eb78621}) to the Dispatcher.

Step 1.5. (Figure 2⑤) The Dispatcher uses the identifier to search the SIP Session List. Since this packet is for a new SIP session, no matched session is found. A new SIP session entry is created (Figure 3②) and the pointer of the *INVITE* message is inserted into the linked list (Figure 3③) of this SIP session.

At this point, a new SIP session has been created and the SIP/RTP Processor proceeds to process next packet pointer. The subsequent SIP provisional messages (e.g., *100 Trying* message and *180 Ringing* message) are inserted into the SIP Session List (Figure 3④ and ⑤) following Steps 1.1–1.5. Suppose that the *200 OK* message corresponding to the *INVITE* message is captured by the Packet

Copyright © 2006 John Wiley & Sons, Ltd.

Preprocessor, the SIP/RTP Processor takes the following actions.

Steps 2.1 and 2.2. These steps are the same as Steps 1.1 and 1.2.

Step 2.3. (Figure 2②) The SIP Parser detects that the packet is a SIP message. The SIP Parser then retrieves the IP address 3ffe:3600:1::2 from the SDP *c* field and the port number 9000 from the SDP *m* field to obtain the RTP connection information of the called party.

At this point, the SIP Parser has obtained the IP address and port number information for the RTP connections. For the calling party, the IP address 3ffe:3600:1::1 and the port number 9000 are obtained at Step 1.3. For the called party, the IP address 3ffe:3600:1::2 and the port number 9000 are obtained at Step 2.3.

Step 2.4. (Figure 2④) The SIP Parser inserts the IP addresses and the port numbers into the RTP Connection List (see Entries 1 and 2 in Figure 4(b)). The IP address 3ffe:3600:1::1 is inserted in the SRC IP field in Entry 1 and the DST IP field in Entry 2. The IP address 3ffe:3600:1::2 is inserted in the SRC IP field in Entry 2 and the DST IP field in Entry 1. The port number 9000 and the '*' mark are inserted into the DST Port and SRC Port fields of Entries 1 and 2. The '*' mark is inserted into the SSRC field of Entries 1 and 2.

Step 2.5. (Figure 2③) The SIP Parser returns the session identifier to the Dispatcher.

Step 2.6. (Figure 2⑤) The Dispatcher uses the session identifier to search the SIP Session List. A matched session (created at Step 1.5) is found. The Dispatcher then inserts the pointer of the *200 OK* message into the linked list of the matched SIP session (Figure 3⑥).

At this point, the SIP *200 OK* message handling is complete. The SIP *ACK* message corresponding to the *200 OK* message is inserted into the SIP Session List (Figure 3⑦) following Steps 1.1–1.5. Suppose that the calling party starts to send the first RTP packet to the called party. This RTP packet is captured, and the SIP/RTP Processor takes the following actions.

Step 3.1. This step is the same as Step 1.1.

Step 3.2. (Figure 2②) The Dispatcher searches the RTP Connection List and finds the matched entry (i.e., Entry 1 in Figure 4(b) created at Step 2.4). In this entry, the source port number is '*' and the SSRC value is '*'. The Dispatcher retrieves the SSRC value (i.e., 29696) from the RTP header of the packet, and then replaces the void port number '*' and the SSRC value '*' by '9002' and '29696', respectively. The modified RTP Connection List is shown in Figure 4(a)①. The Dispatcher then inserts the packet pointer into the linked list of the matched entry (Figure 4(a)⑤).

Step 3.3. (Figure 2②) After the Dispatcher successfully stores the packet pointer into the RTP Connection List, it proceeds to retrieve the next packet pointer.

Upon receipt of a RTP packet sent from the called party (i.e., 3ffe:3600:1::2) to the calling party (i.e., 3ffe:3600:1::1), the SIP/RTP Processor executes Steps 3.1–3.3. After the RTP packet is processed, the source port number and the SSRC values in entry Figure 4(b)② are determined and the values are shown in Figure 4 (a)②. The RTP packet is inserted into the RTP Connection List (Figure 4(a)⑥).

During a packet pointer processing, if the Dispatcher cannot find a matched entry in the RTP Connection List and the SIP Parser indicates that this packet does not carry any SIP message, the SIP/RTP Processor drops this packet pointer and proceeds to retrieve the next packet.

4. SIPv6 Analyzer Demonstration

Figure 5 illustrates an IMS environment for SIPv6 Analyzer demonstration, where UA1 (Figure 5ⓐ) is the calling party and UA2 (Figure 5ⓘ) is the called party with the IP addresses 3ffe:3600:1::1 and 3ffe:3600:1::2, respectively. These two UAs communicate through *Universal Mobile Telecommunications System* (UMTS; Figure 5ⓑ), IMS (Figure 5ⓒ), and *Packet Data Network* (PDN; Figure 5ⓕ). UA1 and UA2 register the accounts wechen@cs.nctu.edu.tw and liny@cs.nctu.edu.tw, to the *Call Session Control Function* (CSCF; Figure 5ⓓ) in the IMS network. After a call is set up, the RTP packets are transmitted through the Media Gateway (MGW; Figure 5ⓔ). The SIPv6 Analyzer (Figure 5ⓗ) can be inserted in either IMS or PDN to analyze the SIP and the RTP messages. In this demonstration, the SIPv6 Analyzer and UA2 connect to the PDN through a hub (Figure 5ⓖ).

The SIPv6 Analyzer captures all packets sent from/to UA2 through the hub, organizes the SIP messages by following Steps 1.1–1.5 and Steps 2.1–2.5, and

Copyright © 2006 John Wiley & Sons, Ltd.

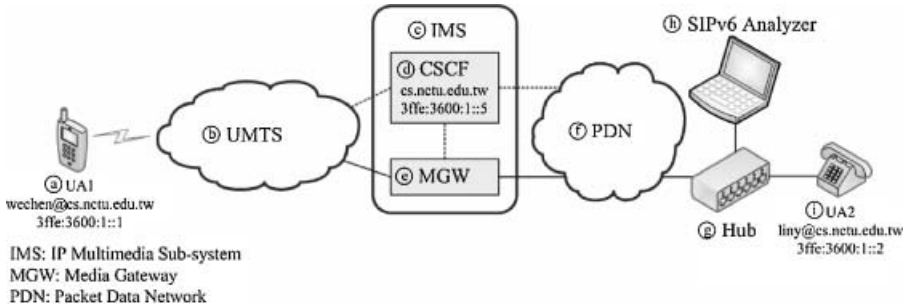

Fig. 5. An IMS environment for VoIP calls.

classifies the RTP packets into the corresponding sessions following Steps 3.1–3.5. After the capturing process, a user can analyze the SIP message flow and replay the multimedia content.

The user can request the SIPv6 Analyzer to draw the SIP message flow by selecting the 'Draw Message Flow' or the 'Draw Message Flow from Headers' items (Figure 6①). If the first item (i.e., Draw Message Flow) is selected, the SIPv6 Analyzer plots the SIP message flow based on the source and destination IP addresses. If the second item (i.e., Draw Message Flow from Headers) is selected, the SIPv6 Analyzer plots the SIP message flow based on the SIP header fields (e.g., SIP *Via* or *Route* header field). Figure 6ⓑ shows the SIP message flow that is plotted by using the SIP *Via* header field. In this message flow, the dashed lines represent the SIP message deliveries that are generated based on the SIP *Via* header field. For example, the *Via* header field of the SIP INVITE message, which is captured by the SIPv6 Analyzer, contains two *Uniform Resource Identifiers* (URIs [7])—3ffe:3600:1::1 and 3ffe:3600:1::5. The SIPv6 Analyzer plots a dashed line from 3ffe:3600:1::1 (i.e., UA1) to 3ffe:3600:1::5 (i.e., CSCF) based on this information. Through this unique function not provided in Ethereal, the SIPv6 Analyzer can provide message interaction (dashed lines) between UA1 and CSCF even if the analyzer is not connected to these two nodes.

To analyze the captured RTP packets, the user selects the 'RTP Viewer' tab (Figure 6③) to enable the Statistic Generator (Figure 1ⓟ) to present the RTP Session List (Figure 6④). To play the audio or video, the user adds the corresponding entry (Figure 6④) to the Media Instance (Figure 6⑦). Then the user plays the content carried in the RTP packets by pressing a control button (i.e., Play/Hold/Stop buttons in Figure 6⑧). Figure 6⑤ shows the video generated by the Video Player (Figure 1ⓝ). Through this function, the user can evaluate the quality of the audio/video carried by the captured RTP packets.

To evaluate the voice/video quality under different jitter buffer sizes, the user clicks the configuration button (Figure 6②) to set up the length of the jitter buffer (Figure 6⑥). The status line of the RTP Viewer (Figure 6⑨) presents the current jitter buffer length (i.e., 100 ms), the number of dropped packets (i.e., 0 packets), and the mean jitter (i.e., 60.09 ms) of the selected RTP connection. Increasing the length of jitter will decrease the number of dropped packets, but will increase the communication delay. The user can observe the effects of different jitter buffer sizes through this function.

5. Conclusions

This paper describes the SIPv6 Analyzer, a tool that investigates the IMS-related protocols during IMS service deployment. The SIPv6 Analyzer provides the SIP dialog list, the SIP message flow generation, and the audio/video replay functions for evaluating the IMS-related protocols. Through these unique functions, the users can effectively observe the SIP message flows and the audio/video qualities of the IMS calls. We demonstrate these functions by using an IMS call example. The SIPv6 Analyzer won the top prize at the Japan's IPv6 Appli-Contest in 2004.

Acknowledgment

This work was sponsored in part by NSC Excellence project NSC 94-2752-E-009-005-PAE, NSC 94-2219-E-009-001, NSC 94-2213-E-009-104, NTP VoIP Project under grant number NSC 94-2219-E-009-002, NTP Service IOT Project under grant number NSC 94-2219-E-009-024, Chung Hwa Telecom, IIS/

Copyright © 2006 John Wiley & Sons, Ltd.

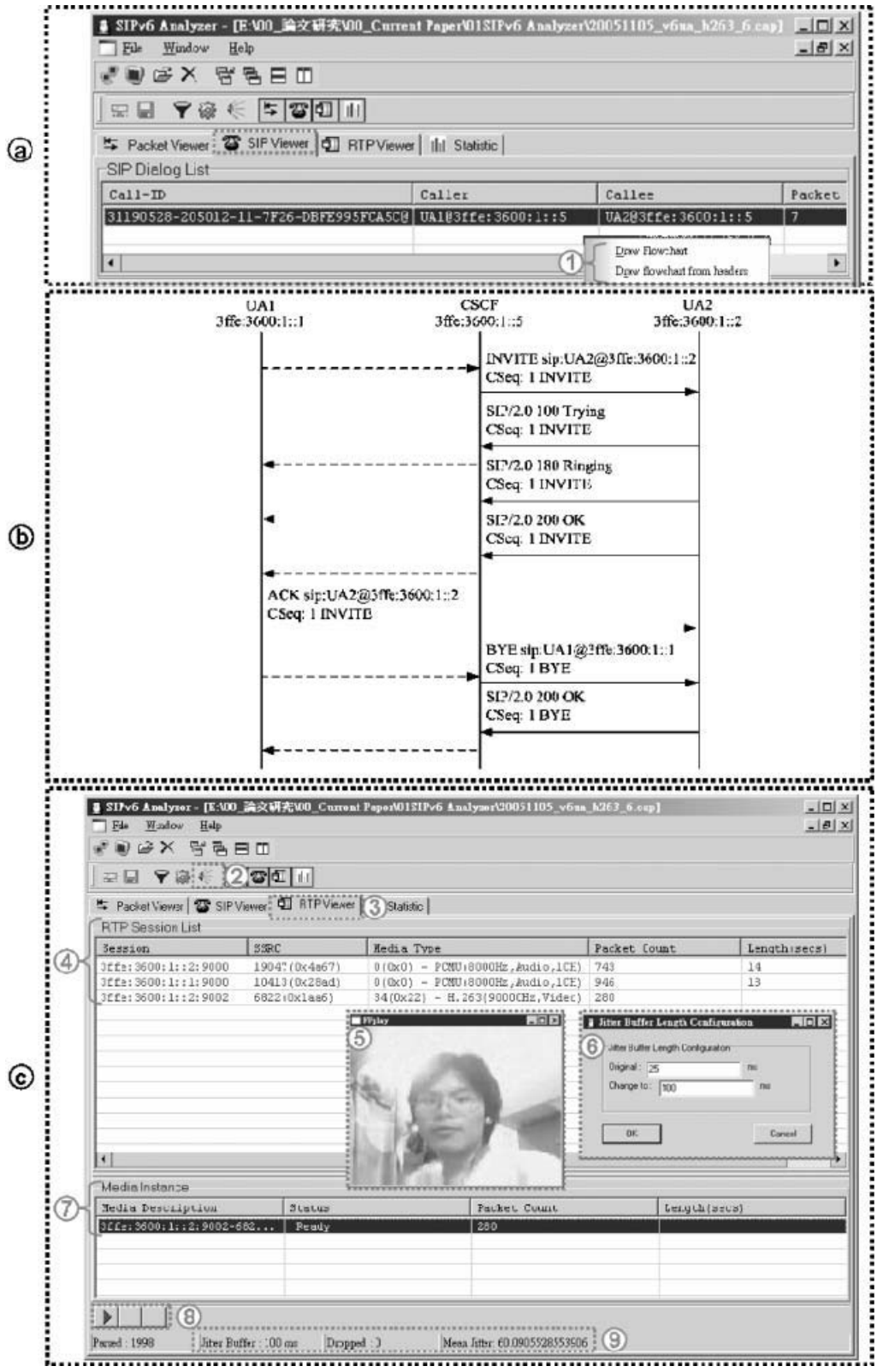

Fig. 6. The snapshot of the SIP Viewer, the SIP Message Flow, and the RTP Viewer.

Copyright © 2006 John Wiley & Sons, Ltd.

Academia Sinica, ITRI/NCTU Joint Research Center, and MoE ATU.

References

1. 3GPP. 3rd Generation Partnership Project; Technical Specification Group Services and Systems Aspects; IP Multimedia Subsystem Stage 2. Technical Specification 3G TS 23.228 version 5.1.0 (2001-06), 2001.
2. Rosenberg J, Schulzrinne H, Camarillo G, *et al.* SIP: Session Initiation Protocol. RFC 3261, IETF, June 2002.
3. Schulzrinne H, Casner S, Frederick R, Jacobson V. RTP: A Transport Protocol for Real-Time Applications. IETF RFC-3550, July 2003.
4. Deering S, Hinden R. Internet Protocol, Version 6 (IPv6) Specification. IETF RFC 2460, December 1998.
5. Ethereal: A Network Protocol Analyzer. http://www.ethereal.-com/.
6. WinPcap: The Free Packet Capture Library for Windows. http://winpcap.polito.it/.
7. Handley M, Jacobson V. SDP: Session Description Protocol. RFC 2327, IETF, April 1998.
8. Wing D. Common Local Transmit and Receive Ports (Symmetric RTP). IETF Internet-Draft draft-wing-behave-symmetric-rtprtcp-01, June 2005.

Appendix

A Comparison of VoIP Analysis and Test Tools

In this appendix, we compare the SIPv6 Analyzer with several existing VoIP analysis and test tools including Ethereal, SIP Scenario Generator, WinSIP and RTP Tools.

Ethereal is a protocol dissector, which can analyze more than 700 protocols. Ethereal provides the VoIP analysis functions including the SIP message flow illustration and the RTP packet statistics (e.g., the jitter statistics). *SIP Scenario Generator* processes a trace file that can be created by Ethereal or the SIPv6 Analyzer. Based on this file, SIP Scenario Generator retrieves the SIP messages and then generates the SIP message flows. *WinSIP* establishes the bulk SIP calls for performance evaluation of VoIP devices. *RTP Tools* include the rtpdump, the rtpsend, and the rtpplay tools. rtpdump listens on a specific IP address and port pair to receive the RTP packets, and then stores these packets into an output file. rtpsend and rtpplay generate the received RTP packets based on the output file. We compare these tools with the SIPv6 Analyzer as follows.

SIP Message Flow Illustration: The SIPv6 Analyzer can plot the SIP message flow based on the IP addresses or the SIP header fields. Through this function, a user can identify all network nodes that are visited in the SIP signaling path. In Ethereal and SIP Scenario Generator, the SIP message flows are generated based on the source IP address and the destination IP address. Therefore, only the source and the destination of the captured SIP message can be identified. Both WinSIP and RTP Tools do not support the SIP message flow illustration function.

Audio/Video Replay: The SIPv6 Analyzer can play the audio/video carried in the RTP payload for evaluating the quality of the captured calls. Other tools do not provide this function.

IPv6 Support: The SIPv6 Analyzer and Ethereal can capture and analyze IPv6 packets. On the other hand, SIP Scenario Generator, WinSIP and RTP Tools do not support IPv6.

Packet Generation: The SIPv6 Analyzer, Ethereal, and SIP Scenario Generator do not include the packet generation function. On the other hand, WinSIP generates the bulk SIP calls including the SIP and the RTP packets to test a VoIP device. Like WinSIP, RTP Tools can test VoIP devices except that the packets for testing are collected by RTP Tools.

Authors' Biographies

Whai-En Chen received a Bachelor of Science degree in Electric Engineering from Tam Kang University in 1997, and received a Ph.D. in Computer Science from National Tsing Hua University (NTHU) in 2002. He began serving as a Research Assistant Professor in National Chiao Tung University (NCTU) and joined National Telecommunications Program (NTP) to deploy a SIP-based VoIP Platform from 2002. His research interests include 3GPP IP core-network Multimedia Subsystem (IMS), SIP-based VoIP services, IPv6 translation mechanisms, Mobile IP and high-speed lookup/classification engines.

Yi-Bing Lin is Chair Professor and Vice President of Research and Development, National Chiao Tung University. His current research interests include wireless communications and mobile computing. Dr. Lin has published over 190 journal articles and more than 200 conference papers. Lin is the co-author of the book *Wireless and Mobile Network Architecture* (with Imrich Chlamtac; published by John

Copyright © 2006 John Wiley & Sons, Ltd.

Wiley & Sons). Lin is an IEEE Fellow, an ACM Fellow, an AAAS Fellow, and an IEE Fellow.

Yueh-Hsin Sung received the B.S. degree from the Department of Computer Science and Information Engineering, National Chiao Tung University (NCTU), Hsinchu, Taiwan, R.O.C., in 2005. His research interests include design and analysis of a VoIP network, Internet telephony integration, and mobile computing.

Copyright © 2006 John Wiley & Sons, Ltd.