Dialogic®

Application Note

Using Ethereal to Debug SIP and RTP on Dialogic® Voice over IP (VoIP) Products

Executive Summary

This application note explains how to set up, configure, and use the Open Source, PC-based network protocol analyzer Ethereal with Dialogic® voice over Internet Protocol (VoIP) telephony products.

Setting up and using Ethereal in a telephony hardware and software environment involves some special considerations, which this application note addresses. This note also serves as a beginning installation, configuration, and user's guide for Ethereal with Dialogic VoIP products. It explores the use of Ethereal in a VoIP test environment by describing how to get started debugging VoIP protocols using Ethereal, providing debugging guidelines, and showing examples of real-world problem scenarios.

Table of Contents

About Ethereal

Ethereal is an Open Source, PC-based protocol analyzer project providing network analysis features that can help a developer understand the behavior of a VoIP installation. Ethereal can monitor and collect Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) packets from a network and sort and display them in an intelligible way.

Ethereal is widely used by network professionals for troubleshooting, analysis, software and protocol development, and education. Its Open Source license allows experts in the networking community to add enhancements.

Ethereal runs on many popular computing platforms, including the Unix, Linux, and Windows® operating systems.

For more information and to download Ethereal, visit http://www.ethereal.com.

Test Environment

The test environment consists of four different VoIP components that interoperate with one another.

Two of the selected VoIP components are based on hardware and software products from Dialogic. The first Dialogic component is a Dialogic® architecture server running Dialogic® Host Media Processing (HMP) Software. (To learn more about HMP software, go to http://www.dialogic.com/products/ip_enabled/hmp_software.htm.) With HMP software, all Session Initiation Protocol (SIP) signaling, Real Time Protocol (RTP) streaming, and media processing are handled in host software and the IP session. Voice streaming data is transmitted through the host Ethernet Network Interface Card (NIC). In addition to SIP, the H.323 protocol is also supported by VoIP hardware and software products from Dialogic, with its functionality exposed via the Global Call API. While SIP is used for the purposes of this paper, H.323 could also be used in the test environment.

The second Dialogic component is a Dialogic architecture server using a Dialogic® telephony board that provides embedded RTP streaming and media processing features.

The third component is a Cisco Analog Telephone Adaptor (ATA) 188 that provides analog to VoIP media gateway features and is attached to two common analog phones.

Finally, a SIP phone from Snom Technology AG is also included in the test environment.

The SIP endpoints are illustrated in Figure 1.

The communications service framework shown in Figure 1 is a C++ framework developed at Dialogic. It is used to develop communications applications using the Dialogic® Standard Runtime Library. While this framework is used in the test environment described in this application note, developers can also use an application or middleware written to the R4 and Global Call APIs instead of this framework.

The example configuration shown in Figure 1 represents a commonly used VoIP application development environment. However, the configuration also contains several features that may be less obvious to newcomers to Dialogic® telecom solutions or to SIP/RTP. These features will be explained in the "General Setup and Use" section that follows.

Methodology

To debug any SIP/H.323/RTP VoIP problem, you closely monitor the operation of the network. Resolving a problem often requires taking an accurate look at all the packets involved during the course of a call. You can purchase expensive "sniffers" that do this based on proprietary hardware, but Ethereal, a freely-available Open Source product, provides comparable functionality for identifying most common (and many less common) issues.

Note that although Ethereal is available for many operating systems, this application note describes its use only on the Windows and Red Hat Linux operating systems.

Windows® Installation

The Windows installation is done by downloading the binary version of Ethereal from http://www.ethereal.com/distribution/win32/.

Systems may require version 3.0 or later of the WinPCap Packet Capture Drivers. Links on the binaries download page point to the driver download. You should install the drivers before proceeding with the Ethereal installation directions. The installation is complete when an Ethereal startup icon appears on the desktop of the target system.

Linux Installation

Ethereal can be installed on Red Hat Linux using Remote Packet Modules (RPMs). Locate the Ethereal version

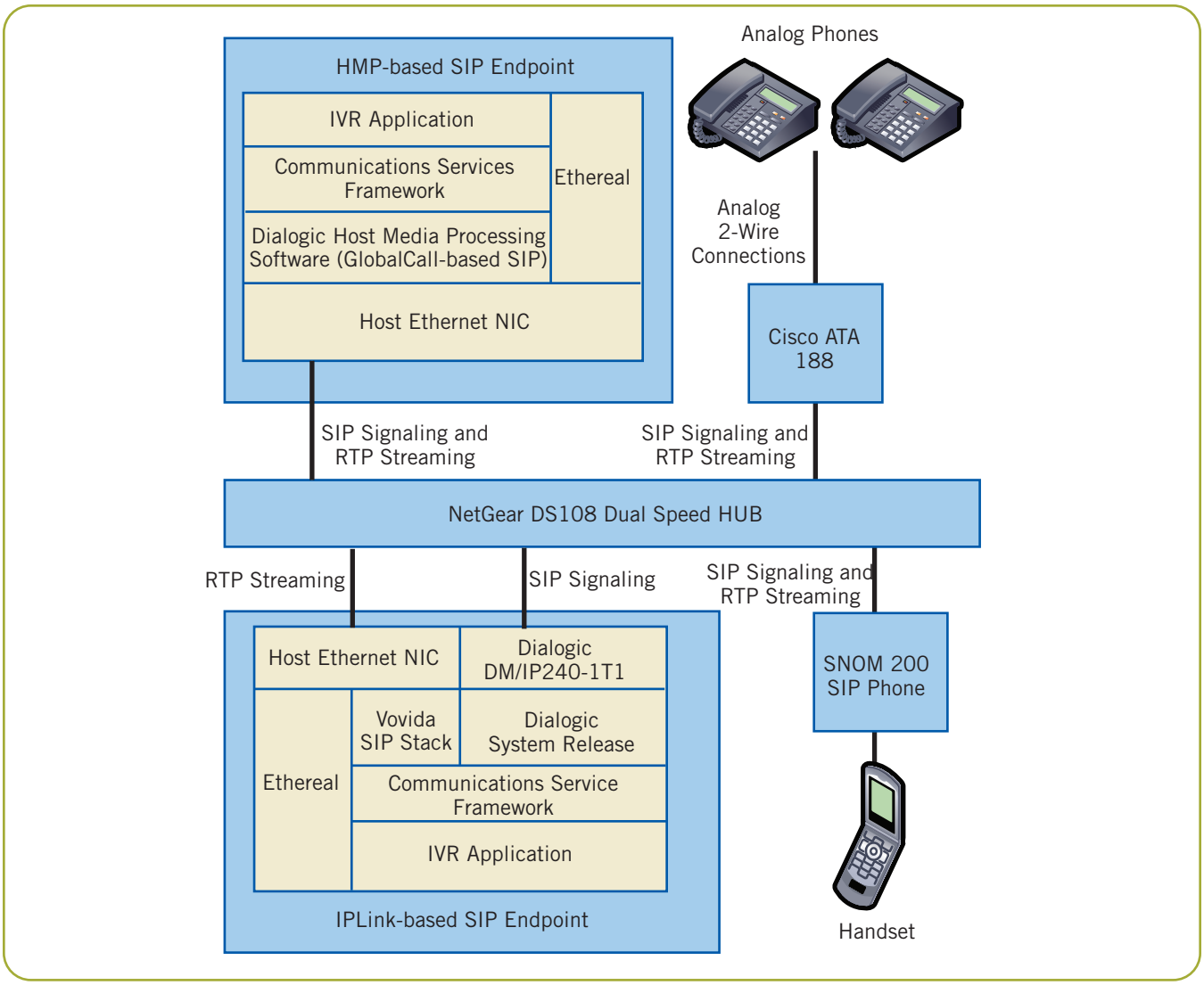

Figure 1. SIP Endpoints

compatible with the Red Hat Linux version you are using (go to http://www.rpmfind.net to locate these and other RPMs.) There are four packages you must install for the full GUI-based product. Install them in this order:

1. **Ethereal-base**
2. **Ethereal-gtk+**
3. **Ethereal-usermode**
4. **Ethereal-kde (or ethereal-gnome, depending on the window manager used)**

You can then start the utility by simply typing "ethereal" in a shell.

Openxtra Ethereal Installation

An extended version of Ethereal, Ethereal-XTRA, is also available under GNU Public License from Openxtra at http://www.openxtra.co.uk/freestuff/ethereal-xtra.php.

Ethereal-XTRA is available only for the Windows operating system, with installation based on an installation wizard.

A useful feature for VoIP work is RTP stream analysis. Once an RTP stream has been captured, it is possible to easily look at relationships between packets. For example, you can peruse delay between packets by scrolling down the list. If a 20 msec packet size has been specified, you can note how close to the 20 msec boundary each packet is delivered. Jitter, which is a variation in packet arrival

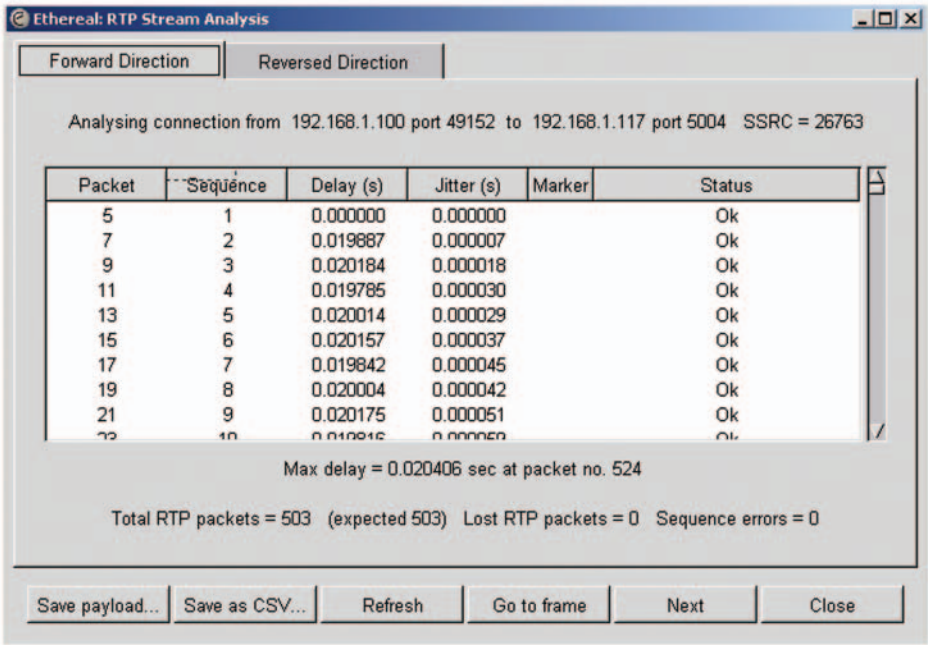

Figure 2 .OpenXtra RTP Analysis

time, is calculated for each packet as well. Figure 2 shows an example of RTP stream analysis.

Note that the two streams — one for each direction — are shown separately. Another feature makes it possible to dump the payloads for either RTP stream into a file so that the audio data it is carrying can be heard.

General Setup and Use

You should run Ethereal in "promiscuous mode," meaning that it will grab and display any packet that it sees on the Ethernet NIC, not just packets destined for its own address. This is important for two reasons:

1. The Dialogic telephony board has its own address and NIC. (The host NIC is not used for RTP.)
2. The RTP (or SIP) packets could be headed elsewhere, not to the system that you are expecting, where Ethereal is running.

You can set promiscuous mode in either of two locations:

- In the Edit -> Preferences -> Capture screen
- In the Capture -> Start -> Capture options window that pops up when starting packet capture

We recommend that you plug in all Ethernet connections into an Ethernet *hub*, rather than a *router*. A router has the intelligence to not waste system resources by forwarding packets not destined for its address. A hub simply broadcasts all packets to all connected NICs. Access to IP packets on the system's LAN can be beneficial when using a host-based Ethernet monitor such as Ethereal. RTP packets may be headed for the NIC on the Dialogic telephony board, but they need to be caught and displayed on the system running Ethereal. Even with an Ethernet router, Ethereal can still be used to monitor the traffic coming in and out of the Dialogic VoIP system (see Figure 1) to debug or to troubleshoot problems specific to that particular system (for example, a connection with no RTP audio).

The operation of Ethereal is described in an online manual, which you can download at http://www.ethereal.com/docs/user-guide. Here we will only cover the basics for setting up and using Ethereal to get you going as quickly as possible with the utility. For general use, set these display options when a Capture -> Start is used to begin packet capture:

X Update list of packets in real time

X Automatic scrolling in live capture

This allows you to get an idea of what is happening in real time (although any in-depth analysis must be done after a call is made and the trace stopped).

Ethereal provides a full set of filters to control collection or display of protocols. Initially, ensure that the SIP protocol is enabled:

Edit -> Protocols -> Enable All

Adding "SIP" to the filter text box on the bottom left side of the screen and then selecting the Apply button will either suppress collection (pre-capture) or display (post-capture) of RTP packets. This is desirable due to the number RTP packets generated. You can deactivate filtering by pressing the Reset button.

RTP packets are normally displayed simply as UDP. To view them as RTP:

- Select a UDP packet
- Decode by selecting Tools-> Decode As -> Transport -> RTP -> Apply

VoIP Problem-Solving

In general, approach any VoIP problem by first looking at the SIP signaling. Only investigate RTP once the SIP protocol trace looks correct. Try a test run in which the inbound or outbound call is made, then look closely at the results. These guidelines describe a quick checklist:

- Are the IP addresses correct for the source and destination systems or phones?
- The default SIP port is 5060 unless otherwise specified. Check the port number in the UDP header.
- Are the addresses and ports correct for the Dialogic telecom board Ethernet NIC?
- For a simple User-Agent-to-User-Agent session, the basic message flow will be similar to:
  - Caller ---- INVITE --→ Called
  - Caller ←-- Trying ---- Called
  - Caller ←-- Ringing --- Called
  - Caller ←-- OK -------- Called
  - Caller ---- ACK -----→ Called

If you see this message flow, then the SIP signaling is likely correct.

- Is there a match in the choice of media (given by Media Attribute) in the Session Description Protocol (SDP) portions of the INVITE and OK messages?
- Does RTP streaming start after the SIP ACK acknowledgement message?
- Are the source and destination addresses for the RTP packets correct?
- Take a look at the RTP payload type. Does the payload type match that requested by the call originator?
- Look at the RTP payload itself. Does it look like a "random" mixture of values that would likely be describing digitized voice? Or is it a more a repetitive set of values, such as 0xff, that could indicate silence?
- If Ethereal-XTRA is used, listening to the RTP-based audio may also be useful. To do this:
  - Select a UDP packet
  - Decode as RTP by selecting: Tools-> Decode As -> Transport -> RTP -> Apply
  - Invoke RTP stream analysis by selecting Tools-> Statistics -> RTP Streams -> Analyze. This will bring up the analysis window, with each stream under a different tab.
  - Streams may be saved separately or mixed as .au files (8000 HZ, 8-bit, mono PCM) by selecting the Save Payload button. This brings up the Save window, where it is possible to select Forward, Reverse, or Both for the stream(s) to save. You must specify a directory in which to save and a file name.
  - The resulting .au audio file can then be heard on a system with a sound card with any audio player such as Windows Media Player or CoolEdit.

Troubleshooting Scenarios

This section presents in detail four real-world problems to demonstrate common failure scenarios and how to address them using Ethereal as an Ethernet packet monitor.

SIP Signaling Working but No Voice

Symptom: An inbound call is made into an interactive voice response (IVR) application on the system that uses the Dialogic telecom board. The SIP stack is external to the board. Based on the host SIP message, flow looks

normal (INVITE/Trying/Ringing/OK/ACK). However, no audio is heard when the SIP connection is complete.

Analysis and Resolution: Initial trace examination shows the expected SIP packets but no RTP packets. The SDP portion of the OK message returned by the IVR system to the SIP call initiator (Cisco ATA) contains the address and port for the RTP stream back to the application. On examination, the connection information field shows an address of 192.168.1.9. However, the IP address of the system itself, used for SIP, is 192.168.1.101. This is not unexpected, since the Dialogic telecom board has its own Ethernet NIC, with an address of 192.168.1.9, assigned when the board was configured.

At the same time, there is a string of Address Resolution Protocol (ARP) packets asking "Who has 192.168.1.9? Tell 192.168.1.115". This means that the Cisco ATA (192.168.1.115) is requesting the MAC (unique hardware address) address that goes along with the IP address of 192.168.1.9. The NIC on the Dialogic telecom board itself would normally supply this information, but for some reason it is not responding.

Examination of the cabling shows that the RJ-45 connector on the Dialogic telecom board has worked itself loose. When it is reattached, normal operation resumes on the next SIP call.

Packets Lost to Local Loopback Interface

Symptom: On receipt of an INVITE method, all further SIP communication stops. RTP streaming does not start.

Analysis and Resolution: If packets were going out, even to an unwanted address, they still should be seen on the NIC of the Ethereal system. However, application traces indicate that the outgoing messaging is taking place. Examining the CallID in the SIP Message Header in the INVITE reveals that it contained, as part of the ID, an address of 127.0.0.1, which is the standard address of any system's local loopback interface. This leads to the source system's /etc/hosts file, where it is found that the system's name, bigsys, appears on the line specifying the local loopback address:

```
127.0.0.1       localhost localhost.localdomain bigsys
192.168.1.100   bigsys
```

Removing the name from the first line resulted in an address of 192.168.1.100 appearing in the CallID and its use as the return address for subsequent messages.

Address Changed for Host Media Processing Software Host System

Symptom: A SIP call inbound to an IVR application on a system based on HMP software proceeds normally. RTP streaming starts, but no audio is heard.

Analysis and Resolution: At first glance, SIP signaling looks normal. RTP streaming starts. However, closer inspection reveals that packets are going only in one direction – from the HMP software IVR system to the Cisco ATA that initiated the call. There are some packets heading in the other direction, but they are ARP Broadcast packets. ("Who has 192.168.1.100? Tell 192.168.1.115". 192.168.1.115 is the IP address of the Cisco ATA.) That makes sense. However, 192.168.1.100 is an old address, not used anymore since the HMP software host was recently moved and its address changed to 192.168.1.200.

Why is the ATA trying to find an old, unused address? Looking more closely at the SDP portion of the OK sent from the HMP software host to the ATA reveals an address of 192.168.1.100 in the Connection Information field. That is how the ATA was instructed to stream to that address, which it is unsuccessfully trying to find.

When HMP software is installed, the system's IP Address is entered as part of the process. Perhaps the RTP stack does not get its NIC address from the IP subsystem itself? That is indeed the case. The address entered as part of the install is kept in the Registry. When changing the system's TCP/IP address, the registry entry got out of synch with the system's actual address.

Reinstalling HMP software and specifying the new IP address will solve the problem. A quicker solution is to modify the address entry with Regedit, which you can find at HKEY_LOCAL_MACHINE -> SOFTWARE -> SBLabs -> dm3ssp -> IP_Addr0.

rfc2833 Type DTMF Not Working Properly

Symptom: An IVR application and Snom SIP phone have been set up to work with rfc2833 DTMF tone generation, but DTMF recognition is not working.

Analysis and Resolution: rfc2833 DTMF is a special type of payload carried by an RTP packet. This type of DTMF does not rely on an audio representation of the tone, and is thus more compact and reliable. However, both parties in a call must agree that rfc2833 is being used for the tone representation.

The Snom SIP phone has been configured for out-of-band DTMF (rfc2833) and a Dialogic® application set up to handle the same sort of DTMF recognition. One side (or possibly both) is not working correctly.

The Snom SIP Phone has been configured to handle out-of-band DTMF with a payload type of 100. The application should be doing the same. But what do these particular DTMF packets look like? After running even a short test call, hundreds or thousands of RTP packets are captured. Ethereal's filter must be used to separate out the voice from rfc2833 packets. This can be done as follows:

- Stop the call's trace.
- Select a UDP packet.
- Use Tools-> Decode As -> Transport -> RTP -> Apply to interpret the packets as RTP.
- Create a filter for RTP payload type 100 with:
  - Edit -> Display Filters.
  - Add a Name for a new filter.
  - Select "Add Expression" button and scroll down and expand RTP entry.
  - Select "Payload type", "==" and enter a value of 100 for the payload.
  - Accept and Save the new entry.
- Apply the filter by selecting the filter button on the bottom left corner of the screen. This will bring up the list of predefined filters, including the one just created. Select and apply it.

The RTP packets remaining in the trace are those with a payload type of 100. The first two digits of the payload correspond to the DTMF key pressed during the test call – 01, 02, etc. So, the SIP phone seems to be doing its job.

This leads to the other SIP endpoint. Examining the application code reveals that it is expecting rfc2833 DTMF, but is actually looking for a different payload type. The Dialogic® RTP streaming interface, once open, has a number of parameters that can be set to define the RTP sessions that will be used. Here, ulParmValue in IPM_PARM_INFO has been set to 96, indicating that a type 96 payload is expected. This does not match the payload type sent by the phone. ulParmValue is set to 100, the application is recompiled and rerun, and subsequent test calls now have working DTMF detection.

Conclusions

The Open Source, PC-based network protocol analyzer Ethereal can be a great help in an environment that uses telephony hardware and software. By understanding the special considerations for this environment as outlined in this document, it is easy to get started debugging Dialogic VoIP protocols using Ethereal.

Product List

Dialogic® Host Media Processing Software System

- Microsoft Windows operating system
- Dialogic Host Media Processing Software
- SIP under Global Call
- Ethereal

Dialogic® Telephony Hardware-Based System

- Tech Support Chassis
- Dialogic® DM/IP241-1T1-PCI-100BT IP Board
- Red Hat Linux operating system
- Dialogic® System Release Software
- Vovida SIP
- Ethereal

Cisco ATA 188

- Two-line analog phone to IP gateway
- SIP software

Snom 200

- SIP phone
- Snom200-SIP

Networking Equipment

- NetGear 8 Port Pro Series Dual Speed Hub, DS108

References

Dialogic product information —
http://www.dialogic.com

Dialogic telecom support —
http://www.dialogic.com/support/

Communications Services Framework Project —
http://sourceforge.net/projects/commsvcfw

Ethereal Project —
http://www.ethereal.com

Linux RPM Locator Website —
http://www.rpmfind.net

Openxtra Ethereal —
http://www.openxtra.com/products/ethereal_xtra.htm

Acronyms

ACK	SIP message sent by the caller after a final response has been received for an INVITE request.
API	Application programming interface
ARP	Address resolution protocol
ATA	Analog telephone adaptor
DTMF	Dual tone multiple frequency
GNU	Open source software licensing from the GNU Project (http://www.gnu.org)
GUI	Graphical user interface
HMP	Host Media Processing
IP	Internet protocol
IVR	Interactive voice response
LAN	Local area network
NIC	Network interface card
rfc2833	RTP request for comment proposal for out-of-band DTMF transmission
RPM	Remote packet module
RTP	Real time protocol
SDP	Session description protocol
SIP	Session initiation protocol
TCP	Transmission control protocol
UDP	User datagram protocol
VoIP	Voice over internet protocol

Dialogic

To learn more, visit our site on the World Wide Web at **http://www.dialogic.com**

Dialogic Corporation
9800 Cavendish Blvd., 5th floor
Montreal, Quebec
CANADA H4M 2V9

9008-02 06/07