IK2213 - Network Services and Internet-based Applications Assignment 2 - SIP Speaker

Ganapathy Raman Madanagopal, Tien Thanh Bui

1. Introduction

In this project, we have developed a SIP Server/Speaker (SIP UA), which can be considered as a robot which automatically answers to the incoming SIP calls. The SIP speaker will waits for the incoming call and answers when call is received from the Client. When the call is answered, a pre-configured voice message will be played out to the client and finally the call is terminated. A web server is also provided, where the administrator or the UA on the SIP Speaker end can type a customized message by accessing the web portal. This new message will be saved and played out to the SIP clients then on. Also the administrator can view, delete the existing message through the web page.

2. Problems & Our Solutions

In this section, we will describe how the application works, the problems we had to solve and our solutions.

2.1. The SIP Server

The SIP server was implemented to listen to all requests destined to the configured interface and port. For each request, it extracts the SIP and SDP information from the request to analyze and answer them correspondingly. If it detects that someone is trying to make a call, a session will be created. After the SIP handshake process, the server will play the pre-configured voice message for the client and end the call after that. The problems that the SIP server had to solve and our solutions are described in this section.

2.1.1. SIP Protocol Implementation

For each request, the SIP header information was extracted using regular expression. The extracted information included: the request type, the call-id, the call sequence, the source's address, the source's username, and the source's tag, the destination's address, the destination's username and the branch. Depending on the request type, each request is then processed as follows:

- **INVITE request**: someone is trying to make a call, so a session will be created based on the call-id. If the destination's username is matched to the preconfigured username, three messages will be then send back to the client to notify him/her that the call has been established: TRYING, RINGING, and OK, in which the OK message containing the media parameters of the call presented in a SDP packet. Otherwise, a 404 NOT FOUND message will be sent back to the client.
- **OPTIONS request**: someone is trying to query the capabilities of the server. In this application, a OK message will be sent back to the client.

- **ACK request**: the client confirms that it has received a final response to an INVITE request. If the server find a session corresponding to this request, it will start sending voice message back to the client using RTP protocol. The call will be ended after that.
- **BYE request**: the client wants to end the call. The server will terminate both SIP and RTP session that have been established for the call.

2.1.2 SDP Protocol Implementation

The information of the media codec of the client was extracted from the SDP packet inside the INVITE SIP request using regular expression. The information included the list of codecs and the client's RTP port. These information was analyzed and used to generate a SDP response inside the OK message that the server replies to the client.

2.1.3 Voice Generation

The default/customized message which is in text form has to be converted into a synthesized sound file before playing out to the client UA. To convert the text message into sound format, we need text-to-speech converter. We used Freetts jar files to convert text into speech format. Also "kevin16" voice is used as the speaking voice. The synthesized voice is then stored into (.wav) file format.

2.1.4 RTP Implementation

After the client has confirmed with ACK request, a RTP session will be created to send voice message back to the client. In order to guarantee that the voice is not modified during the call, a copy of the media file is created and the RTP session takes data from the copy. The RTP session was developed using the Java Media Framework.

2.2. The Web Server

The web server is configured to host the SIP Speaker's or administrator web portal from which the administrator can configure the customized message that they want to play out. The administrator can view and delete the existing message using the web portal. There should always be a default message present to be played out for the clients when there is no customized message. So when the administrator tries to delete the default message an error is thrown. The web server is written in Java and the web portal was designed using HTML.

2.3. Configuration

Some parameters of the application can be configured, including: the SIP interface, the SIP port, the SIP username, the HTTP interface, the HTTP port, the default media file's name, the default message, the current media file's name. These parameters could be loaded from a configuration file using the java.util.Properties class. They could also be extracted from the arguments of java command when running using regular expression. If any of the parameters is not specified, a default value will be used for the parameter. After loading the configuration, all parameters are validated and the user will be notified if any parameter is invalid.

2.4. The SIP Client

Any SIP softphone (PC-PC) can be used as User Agent to call the SIP Speaker. The standard SIP URI format is followed to call the SIP Speaker. The format is follows
<protocol>:<username>@<ip/domain name>:[port].
Eg: "sip:robot@192.168.3.123:5060" or "sip:root@ik2213.lab:8888".
When called to the SIP Speaker, the speaker automatically answers and plays out the pre-configured sound if the client was able to connect to SIP speaker properly. After completion of the voice message the call will be automatically terminated by the SIP speaker (BYE Message will be sent out). Even the client/UA can terminate in between the call.

3. Discussion & Conclusion

Our SIP Speaker service performs basic SIP User Agent functionalities quite well. However, it is limited by the fact that our SIP Server can serve the request only if the request comes from the SIP UA present within the same domain or the network. NAT/STUN services are not implemented. Also in our SIP server we have not implemented SIP proxy, registrar or redirect functionality. We will try to include these functionalities in the future. However, all the required basic functionalities of a SIP UA were met, and the project has given us a solid background in the key areas of SIP, SDP and RTP.