Voice Over IP Implemented on Peerster

Michael Wu, Ruan Ramos Monteiro Silva, Vincent Hu

CPSC 426 - Building Decentralized Systems, Final Project - Fall 2013

Project Description

Summary

In our final project, we extended Peerster to include voice over IP capabilities and encrypted private messages. Specifically, Peerster now have the following features:

- Group calls
- Encrypted private calls
- Encrypted private text chat

How to compile: In the second Peerster folder, type qmake, then make.
How to run the program: ./peerster

Project Architecture

We created two new types of messages: audio messages and private audio messages. Both of them are being created using the QtMultimedia library to encode audio from an input and decode it to an output. We record the user's voice continuously, switching a data buffer every second to send a new audio message with the data recorded. In one second we record 8K of data, so that is the size of the audio messages we are sending to the network. We use the same UDP socket we were using for the other Peerster functionalities.

For group audio messages, we use a flooding algorithm to spread the message throughout the network, because we want every user to receive it as soon as possible. Each user has a status table with the audio messages he received so far, and we use the timestamp (QDateTime) of the moment that the message was sent as the message ID. So when a user receives a new group audio message, he plays it and forward to all his neighbors. If the message is old, he still forwards it to his neighbors while ((current time) - (timestamp) <= threshold). When ((current time) - (timestamp) > threshold), the client simply drops the package.

Private audio messages are handled the same way as private text messages. They also have "Dest" and "HopLimit" fields and they are forwarded to the destination using the same routing table we were using for text chat. The audio data is contained in the "AudioData" field.

We also created an UI for VoIP so that the clients can exchange audio messages (both group and private). We added a “Mute All button” that turns the group chat off and each private messaging window also has a “Mute” button when you want to mute an specific user.

Here is a description of the files we added to Peerster in order to implement these features:

- We created a VoIP panel class (VoipPanel.cc + VoipPanel.hh) that contains all the methods related to audio formatting, sending and receiving audio messages, processing audio input/output and the VoIP UI.

- We also created a different type of messages for audio (AudioMessage.cc + AudioMessage.hh) that contains the constructors to these messages and the methods to serialize group audio messages and private messages before these messages are sent to the UDP socket.

Audio Data Encryption

Encryption Method

Private messages are encrypted over the network using AES. We chose this method because symmetric-key encryption is faster, and we wanted to add as little delay as possible to the voice messages.

The symmetric key used to encrypt and decrypt AES-encrypted messages is exchanged using the Diffie-Hellman key exchange method. To exchange keys, all clients use the same logarithmic group with the same g and p values (generator and modulo) since this is public information. A private value y is then randomly generated to create the public key, which is g^y mod p. This number is sent to the peer with whom you want to do private communication, and the peer sends you his public key as well. As per the Diffie-Hellman method, a symmetric key is generated by combining the public key b you received and your private y value through b^y mod p. The symmetric key is then used to encrypt and decrypt all private audio messages.

Architecture

We created a new file DHKeyMessage in order to pass public key information to private message peers. The DHKeyMessage contains origin, destination, hop limit, and the public key value. Before initializing the private message window, the peers both send this DHKeyMessage to each other in order to calculate the symmetric key, which is later used in this private session to encrypt and decrypt private audio messages.

Division of Work

Michael Wu - group VoIP, input buffer switching, simultaneous playback, audio message flooding & status table

Ruan Ramos Monteiro Silva - piggyback private audio chat on private messaging interface, private mute functions, private chat/audio initializes new private chat window on receiver

Vincent Hu - fixed audio output header problem, DH Key Exchange, AES Symmetric Key encryption of private audio chat