Práctica 3

Vilchis Domínguez Miguel Alonso

1. Contexto:

Alice y Bob siguen trabajando en la misma compañía y las políticas de la empresa siguen siendo restrictivas. Sin embargo, la cantidad de trabajo ha aumentado considerablemente en los últimos días, es por esa razón que ya no les es cómodo escribir por chat, preferirían hacer una llamada a través de tu software para poder seguir platicando sin tener que descuidar el trabajo. Por lo que se te encargó la tarea de agregar la funcionalidad de llamada a tu chat.

2. Objetivo de la práctica

Que el alumno extienda la aplicación de chat de texto a una aplicación que soporte el envío de audio, entre dos equipos que se encuentran en la misma red y con ip fijas.

3. Detalles generales

Es la segunda práctica del proyecto que se irá construyendo durante el semestre, ahora los objetivos se ampliaron a: Mandar texto y audio entre clientes con ayuda de las bibliotecas xmlrpc y pyaudio para conexiones a ubicaciones fijas. El proyecto debe funcionar tanto para equipos dentro de la misma red, como para dos instancias del programa ejecutandose en la misma computadora, en donde cada instancia cuenta con un puerto diferente.

4. Especificaciones:

- En esta práctica se extenderá la funcionalidad de tu aplicación agregando la capacidad de poder realizar una llamada de audio entre dos clientes.
- La manera en la que se creará la interconexión entre los usuarios será usando la biblioteca xmlrpc.
- Es imporante notar que xmlrpc define dos roles, servidor y cliente, en este rol el cliente hace llamadas a procedimientos remotos y el servidor ejecuta lo referente a ese procedimiento remoto y regresa una respuesta.
  Deben tener en mente ese diseño, cuando programen su aplicación.
- El puerto para las conexiones será el 5000 si se está trabajando con computadoras distintas.
- Para el manejo del audio se recomienda el uso de la biblioteca pyAudio:
  Referencia: http://people.csail.mit.edu/hubert/pyaudio/
- La manera en la que se realizará el envío de audio será por medio de dos hilos, uno de ellos grabará el audio de manera continua y guardará el stream en una pila, el otro se dedicará a sacar los datos de la cola y mandarlos por xmlrpc. De acuerdo a nuestro diseño estos threads serán administrados desde la clase canal.

- La estructura del proyecto se encuentra dividida de la siguiente manera:

```
|
|– –Constants
                |– –Constants.py
                |– –AuxiliarFunctions.py
|– –GUI
                |– –LoginWindow.py
                |– –ChatWindow.py
|– –Channel
                |– –ApiServer.py
                |– –ApiClient.py
                |– –Channel.py
                |– –RecordAudio.py
|– –GraphicalUserInterface.py
```

  - *Carpeta Constants:* Dicha carpeta contiene el módulo Constants.py en donde se deben colocar todas las constantes que se utilicen a lo largo del proyecto, con el fin de evitar número mágicos y fomentar la legibilidad del código.
    AuxiliarFunctions: En este archivo encontraran funciones auxiliares cuyo propósito es facilitar el desarrollo del sistema.
  - *Carpeta GUI:* En esta carpeta se desarrollaran el código referente a las interfaces gráficas a través de las cuales el usuario interactuará con el sistema. Deben desarrollarla por completo. Al ejecutar el archivo main de esta práctica se desplegará una ventana, la cual, si se está trabajando de manera local, pedirá el puerto del cliente y el puerto del contacto. Seguido de esto mostrará una ventana de chat para permitir la comunicación. En el caso de trabajar con instancias en distintas computadoras solo se preguntara por el campo ip del contacto, y seguido de esto se mostrará la ventana del chat.

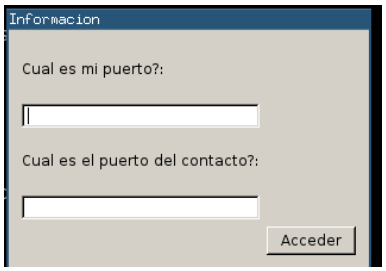

Figura 1: Imagen que muestra la etapa de Login para uso local

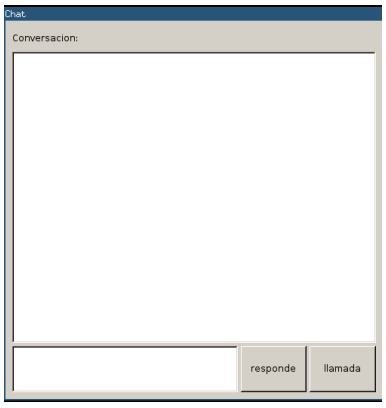

Figura 2: Imagen que muestra la ventana del chat despues del login, con el botón de llamada

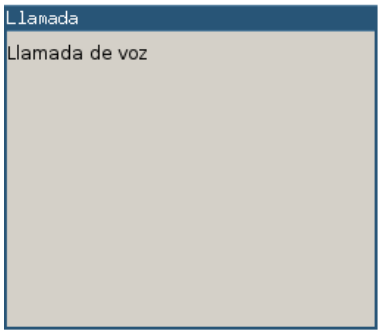

Figura 3: Imagen que muestra la ventana que se muestra al presionar el botón de llamar

- *Carpeta Channel:* En este sitio se implementará todo lo referente a los procedimientos remotos con los que el sistema funcionará. La definición en particular de los archivos es el siguiente:
  - *Channel*: Este archivo contiene la clase Channel, el objetivo de esta clase es organizar los threads que darán soporte a los diferentes servicios que se irán agregando en el chat, el primero de ellos es el thread para dar servicio al servidor de xmlrpc que del cual hara uso el contacto con el que nos estámos comunicando. En esta clase se declaran e inician los threads, además se crean los metodos necesarios para interactuar con la instancia de la clase ApiClient de manera indirecta, desde la clase Channel. La definición del servidor y cliente de xmlrpc como clase se realizará en los siguientes archivos.
  - *ApiServer* : En este archivo se define la clase MyApiServer, la cual representa el servidor de procedimientos remotos de este cliente. A través de la definición de los métodos que se encuentren en la clase FunctionWrapper, otro cliente de chat podrá mandarle datos a este cliente.
    Ahora hay que agregar un método que permita reproducir el audio cuando el cliente le mande el audio en forma de datos binarios.
    Hint: Se debe crear un thread cuyo target sea una instancia de la clase MyApiServer en la clase Channel.
    Hint2: Se recomienda que la reproducción del audio se lleve acabo en otro thread.
  - *ApiClient*: En este archivo se especifica la clase MyApiClient, el cual define la instancia de xmlrpc que tendrá contacto con el Servidor xmlrpc del cliente con el que estamos hablando.
- *GraphicalUserInterface.py*: Este archivo tiene como propósito funcionar como main, es decir el usuario ejecutar $\$pyhtonGraphicalUserInterface.py$ se debe de desplegar su ventana de login, con la instancia que soporta la comunicación entre dos computadoras, es decir, solicitando la ip del contacto. Si el usuario ejecuta $\$pyhtonGraphicalUserInterface.py \quad -l$ entonces se debe de preguntar por el puerto del cliente y el puerto del contacto para poder tener dos instancias dentro de la misma computadora trabajando ambas en localhost.

- Se utilizará la misma estructura que en la práctica anterior por lo que no se les proporcionará una nueva carpeta.

5. Preguntas para el reporte:

- Tu programa funciona bien dentro de la misma red. ¿Si le pasas el proyecto a alguien que se encuentre del otro lado del mundo, podrían comunicarse ?
- En el ejemplo anterior ¿Bajo que restricciones podrían comunicarse?
- Si Alice y Bob están en la misma oficina pero Alice está conectada por ethernet y Bob por wifi. ¿Se pueden comunicar sin problema ? ¿Bajo que restricciones se pueden comunicar?

- Además de las preguntas, se debe agregar un pequeño reporte de las particularidades de su código, como son:
  - Cuál es el flujo del programa y para qué sirve cada archivo dentro de la carpeta GUI y Channel.
  - Principales problemas que se encontraron y cómo los solucionaron.
  - Si un usuario presiona múltiples veces el botón de llamar, ¿qué sucede?
  - Problemas que no fueron solucionados.

6. Entrega

Fecha de entrega: 29 de Agosto del 2016

La entrega es por parejas, la práctica con la estructura mencionada debe estar almacenada en un depósito de github llamado Redes2017, en la rama Practica3, el readme debe contener el nombre de los 2 integrantes del equipo.

Deben mandar el link del depósito con el asunto *[Redes-17]Practica3* al siguiente correo:

mvilchis@ciencias.unam.mx

Nota: Basta con que un integrante del equipo suba su código al depósito.
Nota: A las 11:59 del díal de la entrega se descargará el contenido de los depósitos, para evaluación, asegurense que se encuentre su versión final para esa hora.