Reporte Práctica 3 [Redes 2017]

Nombre(s): Angie Daniela Velásquez Garzón

Bernal Cedillo Enrique Antonio

- **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 estado actual del Software, aun no es posible.

- **En el ejemplo anterior ¿Bajo que restricciones podrían comunicarse?**

  Debido a que hasta ahora unicamente funciona en una misma red (ya sea de forma Local o Remote), hace falta más información para poder comunicar ambos equipos finales a través de Internet.

- **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?**

  Sí.
  Siempre y cuando esten dentro de la misma red con IP fijas.
  Pues con la funcionalidad actual se puede obtener la dirección IP de ambos y establecer la conexión bajo una misma red.

Reporte código

- **¿Cuál es el flujo del programa?**
  1. Inicia la GUI del *Login* la cual de acuerdo al modo en que se inició la sesión, sea local o remota, pedirá al usuario:
     - **Local:** Número del puerto por el cual desea establecer su conexión y el número del puerto al que desea conectarse.
     - **Remoto:** Dirección IP del equipo con el cual desea conectarse
  2. El usuario ingresa los datos solicitados, el canal se encarga de iniciar los hilos para las instancias del cliente y servidor de cada usuario y se establece una conexión si la información ingresada es válida, de lo contrario se informa al usuario que la conexión no se ha podido establecer, paso seguido se presenta la pantalla del chat donde el usuario puede intercambiar mensajes de texto o audio.
  3. El usuario tendrá disponibles dos modalidades para la comunicación, a través de mensajes de texto o de audio:
     - **Mensajes:** Si el usuario escribe un mensaje y presiona el botón *enviar* el canal a través de su cliente establece una conexión al contacto indicado cuando inició sesión y envía el mensaje al servidor del usuario receptor y este se encarga de mostrarlo en la ventana de dialogo.
     - **Audio:** Si el usuario presiona el botón de *llamar* el canal iniciará dos hilos, uno que se encargara de ir grabando el audio e ir encolando la información en una cola y el otro que se encargará de ir tomando lo que se encuentra en la cola y enviarlo al servidor del contacto el cual se encargará de reproducirlo, esté procedimiento se llevará a cabo todo el tiempo que la llamada se encuentre activa y no se permitirá el envío de mensajes.

       Si el usuario selecciona la opción de terminar llamada, se detendrá la grabación de audio, su envío y la llamada concluirá.

- **¿Para qué sirve cada archivo dentro de la carpeta GUI y Channel?**
  - **GUI/ChatGUI.py**

    Clase que a través de una interfaz gráfica le permite al usuario hacer uso de las funcionalidades de comunicación: *envío de mensajes* y *llamadas.* Está clase contiene el canal quien se encarga de establecer la comunicación, enviar y recibir información entre los usuarios del chat.
  - **GUI/LoginGUI.py**

    Contiene clase LoginGUI que se encarga de solicitar la información correspondiente al usuario para establecer la conexión entre los usuarios. La información solicitada varía de acuerdo al modo en que se indique por consola si la conexión es local o remota.

- **Channel/ApiClient.py**
  Clase encargada de representar al cliente de la conexión, permite establecer la comunicación con el servidor del usuario contacto para el envío de mensajes o inicio de una llamada.
  Cuando se desea enviar un mensaje, se establece la conexión con el servidor del contacto y se envía el mensaje.
  Cuando se inicia una llamada, el cliente inicia una instancia de un cliente audio e inicia un hilo que permite grabar de forma continua el audio y guardar dicha información en una cola mientras de forma paralela el cliente establece comunicación con el servidor del usuario de contacto y repetitivamente extrae la información de la cola y la envía al servidor de contacto.
- **Channel/ApiServer.py**
  Clase encargada de recibir los mensajes y reproducir los audios recibidos por parte del cliente del contacto y disponérselos a su usuario. El servidor cuenta con dos servicios, uno que permite recibir los mensajes y mostrarlos en la pantalla de su usuario y otro que se encarga de reproducir el audio capturado por el cliente del contacto.
- **Channel/RecordAudio**
  Archivo que contiene un cliente y servidor para la transferencia de audio entre dos usuarios, la clase de *AudioClient* cuenta con los servicios necesarios para la grabación del audio de su usuario; la clase *AudioServer* provee el servicio necesario para reproducir la información que recibe.
- **Channel/Channel.py**
  Alberga la clase *Channel* la cual administra los hilos del cliente y servidor de cada usuario, gestiona el envío de mensajes, llamadas (para la cual inicia un nuevo hilo del cliente que se encarga de enviar continuamente al servidor de contacto el audio) y finalización de llamadas. Maneja los hilos para iniciar el cliente y servidor, además de gestionar la llamada.

- **Principales problemas encontrados**

  Falta encontrar cómo detener los hilos encargados del audio, pues se mantienen vivos a pesar de querer terminar la llamada (Esto es, porque el main obviamente sigue corriendo).
  Parece que hay otra clase de hilo que sí tiene el metodo “terminate”.

  Buscar como esconder la pantalla del chat y mostrar la pantalla de llamada con el boton de “colgar”, deberia ser facil, pero los intentos hasta ahora no han ayudado.

- **Si un usuario presiona múltiples veces el botón de llamar, ¿qué sucede?**

  Hilos innecesarios extra para realizar la llamada son creados, lo cual produce varias instancias de grabado y reproduccion de audio.
  Sin embargo esto se puede solucionar si la funcion llamar no realiza nada cuando ya se encuentra una llamada activa y limitando la interfaz grafica.

- **Problemas que no fueron solucionados**

  Detencion de hilos de audio sin efectos secundarios.