Práctica 5

Vilchis Domínguez Miguel Alonso

1. Contexto:

Tu chat se ha hecho famoso dentro de la empresa en la que están trabajando Alice y Bob y varios de sus compañeros están interesados en obtener una copia del software, sin embargo, el diseño actual solo permite conexión entre 2 equipos, además de que los dos equipos tienen la interfaz de conexión abierta en el momento de hablar. Es por ello que le haremos modificaciones para mejorar la funcionalidad del programa, teniendo en cuenta que habrá más usuarios conectados a nuestro chat.

2. Objetivo de la práctica

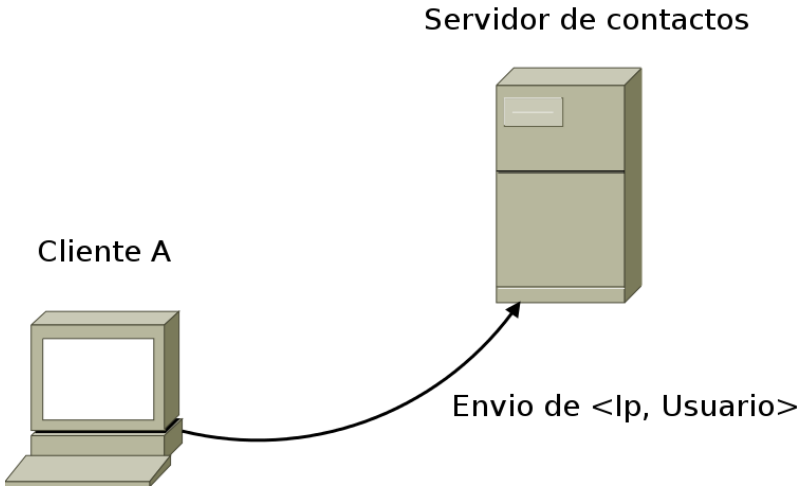
Que el alumno implemente un servicio de direcciones para permitir la conexión entre varios equipos.

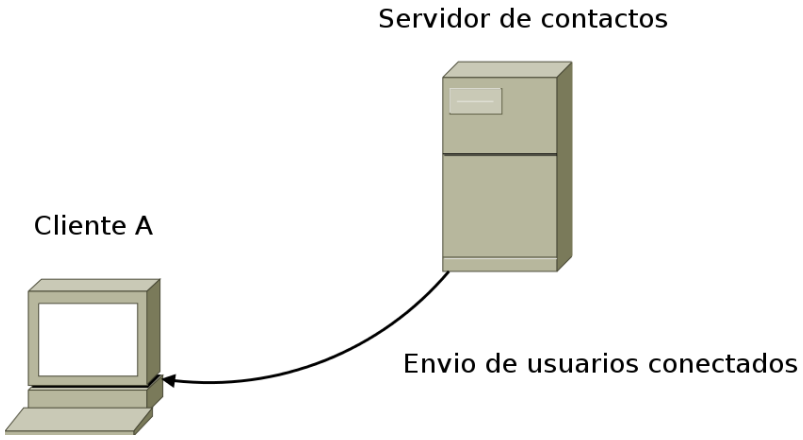
3. Especificaciones:

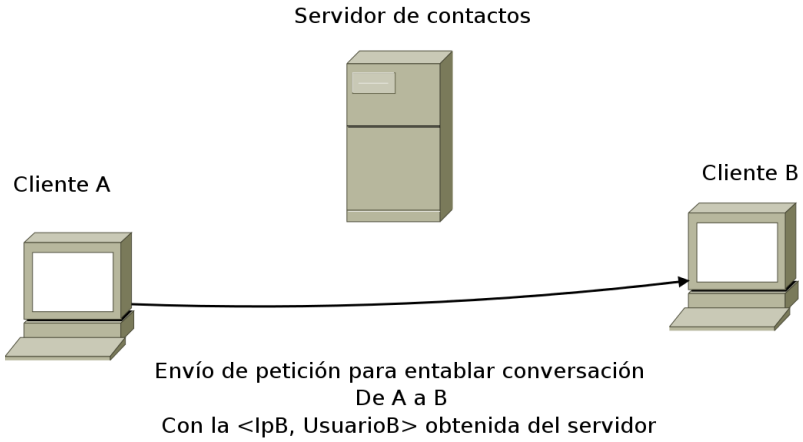
- Se agregará un segundo elemento al ambiente del chat, el cual servirá como servidor de contactos o servicio de direcciones.
- El servidor de contactos seguirá trabajando con xmlrpc.
- La primer ventana de la interfaz preguntará por la dirección Ip del servidor de contactos (o el puerto si se está trabajando de manera local) y nombre del usuario.
- Cuando un cliente inicie sesión, le mandará sus datos al servidor de contactos, es decir, la Ip ,puerto y nombre de usuario. El servidor de contactos le responderá con un diccionario cuyas llaves estarán determinadas por el nombre del usuario; el valor de cada entrada estará compuesto de otro diccionario que contendrá el puerto, la ip y el nombre de usuario.
  Esta estructura representa los contactos conectados hasta el momento(Sin incluir al contacto actual).
- Por el momento delegaremos la resposabilidad de mantener actualizada la lista de contactos a los clientes, por lo que se requiere agregar un botón para actualizar dicha lista.
- El usuario A podrá seleccionar entre los contactos que se encuentran contectados para entablar una conversación, en ese momento se creará una interfaz del lado del que inició la conversación (usuario A), como del lado del usuario seleccionado por el primero (usuario B). Razón por la que deben de agregar la funcionalidad al programa para que se encuentre escuchando por conexiones entrantes y en el momento que se cree una conexión de A a B, se genere una interfaz de chat en ambos equipos. La conexión se llevará acabo con los datos que fueron proporcionados por el servidor de contactos, es decir, la Ip del usuario B y el nombre de dicho usuario.
- La actualización de clientes que ya no están disponibles es responsabilidad del usuario, por lo que, deberás agregar un botón para cerrar sesión y de esa manera se actualice la lista del servidor de contactos.

4. Esquema del programa

Inicio de conexion Respuesta del servidor Conexión entre equipos







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

```
|
|--Constants
        |--Constants.py
        |--AuxiliarFunctions.py
|--GUI
        |--LoginWindow.py
        |--ListContactWindow.py
        |--ChatWindow.py
        |--CallWindow.py
|--Channel
        |--ApiServer.py
        |--ApiClient.py
        |--Channel.py
        |--RecordAudio.py
        |--RecordVideo.py
        |--DirectoryChannel.py
|--Directory
        |--Directory.py
|--GraphicalUserInterface.py
```

  - Las funcionalidades de las carpetas: *Carpeta Constants* y *Carpeta GUI* siguen teniendo la misma funcionalidad que en las prácticas pasadas.
  - *Carpeta Directory:* En esta carpeta se definirá lo referente al directorio de ubicación:
    - *Directory* en este archivo se definirá la clase GeneralDirectory. Esta clase representa un servidor de xmlrpc, que contiene los siguientes métodos:
      - $get\_contacts\_wrapper(self,\ username)$: Método que regresa los contactos activos en el directorio excluyendo el contacto actual.
      - $connect\_wrapper(self,\ ip\_string,\ port\_string,\ username)$: Método que agrega al cliente al diccionario de usuarios activos.
      - $connect\_wrapper(self, ip\_string, port\_string, username)$: Método que elimina al cliente de los usuarios activos.
  - *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*:En el archivo Channel se trabajará con dos clases, RequestChannel que contendrá los métodos necesarios para hacer uso de los métodos del api de un contacto, (un ApiClient para cada usuario). Y la clase BidirectionalChannel la cual representa un canal de comunicación bidireccional (contiene un ApiServer y un ApiClient).
      Note que un RequestChannel será creado cuando un cliente externo (sea B) quiera hablar con el cliente actual (sea A), para poder entablar una conversación de A a B.
    - *DirectoryChannel*: Este archivo contiene la clase DirectoryChannel que servirá como comunicación entre un cliente y el directorio de ubicación. Internamente trabajará con un canal bidireccional que será creado una vez que se finalice la etapa de login y el apiClient estará dirigido hacia el servidor de direcciones.

    - *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.
      El intercambio de datos (mensaje y video) deben ir precedidos por un identificador de conversación, este se puede crear con el método *get_message_header* que se encuentra en las funciones

auxiliares. Dicho identificador estará compuesto por el username e ip, con el objetivo de que el ApiServer despliegue en la ventana adecuada los datos que está recibiendo.

- *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 ejecutará $*pyhtonGraphicalUserInterface.py* y se debe de desplegar su ventana de login, con la instancia que soporta la comunicación entre dos computadoras, solicitando la ip del contacto. Si el usuario ejecuta $*pyhtonGraphicalUserInterface.py* $-l$ entonces se debe 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 adjunta la nueva estructura del proyecto.

5. Preguntas para el reporte:

- ¿Cuál es el principal problema de la arquitectura propuesta?
- ¿Cómo solucionarías ese problema ?
- ¿Existe alguna manera de saber de donde proviene el mensaje sin el encabezado que contiene el identificador?

6. Entrega

Fecha de entrega: 26 de Septiembre 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 Practica5, el readme debe contener el nombre de los 2 integrantes del equipo.

Deben mandar el link del depósito con el asunto *[Redes-17]Practica5* 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.