Práctica 7

Vilchis Domínguez Miguel Alonso

1. Contexto:

En la primer iteración del proyecto se uso xmlrpc, ya que la curva de aprendizaje para aprender xmlrpc es muy sencilla, sin embargo, nuestro interior científico sabe que xmlrpc utiliza muchos recursos y no es tan eficiente es por eso que estamos preocupados por la manera en la que implementamos el sistema, por lo que utilizaremos sockets de tcp en lugar de xmlrpc.

2. Objetivo

Que el alumno se familiarice con sockets (TCP), para cambiar la comunicación entre el servidor de ubicación y los clientes.

3. Especificaciones:

- Se dejará de utilizar la biblioteca xmlrpc para utilizar sockets en su lugar.
- La estructura del proyecto sigue siendo la misma:

```
|
|--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
```

  Los cambios que deben considerarse son:

  - En *ApiClient.py* se dejará de crear un proxy al servidor del contacto, seguiremos el concepto de procedimiento remoto, aunque ahora del lado del cliente se implementará cada método al que se hacer referencia, pasando los datos al formato adecuado para mandarlo al servidor. Se recomienda usar una convención de

$$paquete\_1 : \ nombre\_del\_metodo\$\#no.parametros\$\#parametro\_1...\$\#parametro\_n$$

Donde $# es una cadena especial para separar los métodos, de sus parametros. Este paquete será mandado al servidor, el cual se encargará de hacer el parseo y procesar la información de manera adecuada.

- En *ApiServer.py*, se dejará de crear un servidor xmlrpc, ahora MyApiServer será un thread y en el método run estará a la escucha de los métodos solicitados, hará el parseo de los datos y procesara la información.
- En *Directory.py* cambiará su estructura de manera análoga al apiServer.
- En *DirectoryChannel* se debe dar soporte para la consulta del metodo $access_wrapper$ con el metodo *access*. Ahora con sockets

4. Preguntas para el reporte:

- Con uso de wireshark o tcpdump realizar un comparativo del tráfico que genera la práctica anterior con esta nueva práctica. ¿Cuál versión genera menos tráfico ?
- De la pregunta anterior, de manera experimental, ¿cuál es la cantidad de tráfico generado por metadatos de xmlrpc ?
- De manera teórica, ¿cuál es la cantidad de tráfico generado por metadatos de xmlrpc ? (Revisar el api y el contenido de un paquete xmlrpc).

5. Entrega

Fecha de entrega: 10 de Octubre 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 Practica7, el readme debe contener el nombre de los 2 integrantes del equipo.

Deben mandar el link del depósito con el asunto *[Redes-17]Practica7* 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.