Taller Libre II TP - Sockets

1) **En la captura se observa que la información viaja en texto plano**.

Utilizando TCP se observa el establecimiento de la conexión (three-way handshake) y en UDP no se observa establecimiento de conexión. El comportamiento de ambos es el mismo a excepción de lo mencionado anteriormente.

Ejemplo de ejecuciones:

Opción 1:

- Python server_tcp.py
- Python cliente_tcp.py

Opción 2:

- Python server_tcp.py --ip xxx.xxx.xxx.xxx --puerto xxxx
- Python cliente_tcp.py --ip xxx.xxx.xxx.xxx --puerto xxxx

Opción 1:

- Python server_udp.py
- Python cliente_ udp.py

Opción 2:

- Python server_udp.py --ip xxx.xxx.xxx.xxx --puerto xxxx
- Python cliente_udp.py --ip xxx.xxx.xxx.xxx --puerto xxxx

2) **Este ejercicio se resolvió utilizando la librería pytz. Tamaño de PDU fijo de 60 caracteres que contiene un formato de fecha válido para strftime.**

Ejemplo de ejecuciones:

Opción 1:

- Python server.py US/Eastern
- Python cliente.py "%d-%m-%y %H:%M:%S %Z"

Opción 2:

- Python server.py US/Eastern –ip xxx.xxx.xxx.xxx –port xxxx
- Python cliente.py "%d-%m-%y %H:%M:%S %Z" --ip xxx.xxx.xxx.xxx –port xxxx

3) Ejemplo de ejecuciones:

- Python clientehttp.py http://www.unlu.edu.ar UNLU.html
- Python clientehttp.py http://www.labredes.unlu.edu.ar/sites/www.labredes.unlu.edu.ar/files/site/data/bdm/clase-seleccion-de-modelos.pdf ARCHIVO.pdf

5) Para este ejercicio se utilizó criptografía simétrica y fue implementado por el módulo pycripto, cifrando los datos a través de AES. La clave privada que se utiliza para encriptar y desencriptar el mensaje (usuario y contraseña) que viaja por la red, fue seteado por defecto en la implementación.

Usuario y contraseña para logearse:

Usuario: Santiago (S mayúscula)

Contraseña: 12345

Setear parámetros en:

/config/**config_servidor.cfg** (para el servidor)

/config/**config_cliente.cfg** (para el cliente)

Para deslogear escribir: **exit**

La contraseña del usuario se almacena usando el algoritmo de hash "MD5".

Cuando el usuario y contraseña son recibidos por el servidor, este aplica Hash.MD5 y compara con la contraseña "hasheada" en el archivo: "usuarios.txt".

Se cuenta con un archivo de log para contener los datos de sesión.

Ejecución del telnet:

- Python Servidor.py
- Python Cliente.py

7) Ejecución del servidor proxy:

Opción 1:

- Python Servidor_Proxy.py

Opción 2:

- Python Servidor_Proxy.py --ip xxx.xxx.xxx.xxx --port xxxx

Configurar un navegador de acuerdo a la IP y PORT seteados y cargar alguna página web HTTP.

9) Ejecución del Chat punto a punto:

Opción 1:

- Python chat-server.py
- Python chat-client.py

Opción 2:

- Python chat-server.py --ip xxx.xxx.xxx.xxx –port xxxx
- Python chat-client.py --ip xxx.xxx.xxx.xxx --port xxxx

15) En este ejercicio todos los nodos tienen la misma responsabilidad (SNodo.py). Estos nodos actúan tanto como cliente y como servidor. Cada nodo realiza "broadcast" para enterarse que nodo ingreso a la red y que recursos tiene ese nodo. Definí una función "mantener activos" que calcula para un nodo X, la diferencia entre el instante tiempo actual y el último contacto con ese nodo X y determina si todavía se encuentra activo en la red o no. El resultado de la diferencia se compara con "tiempo_nodo_activo" (valor máximo, por el cual se puede considerar activo a un nodo X), que es "seteable" en el archivo "config.cfg.

El intervalo de tiempo en que cada nodo realiza el broadcast es "seteable" desde el archivo "config.cfg"

Para emitir un listado que muestre para cada documento en que nodo está éste disponible, definí Solicitar_Listado.py. Este puede solicitar a cualquier nodo activo de la red, el listado mencionado anteriormente.

Este ejercicio cuenta con:

"/**config**/" donde se encuentra el archivo de configuración (config.cfg)

"/**distributed_index**/" almacena un archivo por cada nodo, con el respectivo índice de documentos que le corresponde a dicho nodo.

"/**files/**" contiene los documentos que comparte en la red.

"/nodos_activos/" almacena un archivo donde lista los nodos activos de la red junto con una marca de tiempo.

Ejecución:

Por cada nodo se debe contar con: "**/config/config.cfg**", "**/distributed_index/**","**/files/**" (con archivos dentro para compartir), "**/nodos_activos/**"

Setear parámetros en el archivo "**config.cfg**" ubicado en "**/config/**"

- Python SNodo.py

Luego de que existan 2 nodos o más en la red, se procede a ejecutar:

- Solicitar_Listado.py –ip xxx.xxx.xxx.xxx –port xxxx

Importante: En algunos casos de prueba a fallado la lectura de la IP del nodo seteado en el archivo de configuración. Si esto sucede, el valor que se le estaría pasando es vacío (" "), y esto en sockets significa que el servicio toma todas las interfaces de red que existan (incluyendo '127.0.0.1'). Esto se traduce durante la ejecución como duplicación de índice.

Solución: Si esto ocurre, la solución sería "setear" manualmente el parámetro "self.ip" de SNodo.py en la clase SNodo.