Trabajo Práctico

Programación de Sockets de Internet con Python

Arellano E. Nahuel
nahuel.arellano@gmail.com

1) Analizando las capturas del echo se puede contemplar las diferencias entre los protocolos. En TCP se puede visualizar el Three-way handshake (negociación en tres pasos) y las confirmaciones de las tramas enviadas, mientras que en UDP solo se envia las tramas con mensaje sin confirmación. En ambos casos el mensaje viaja en texto plano.

4) Se desarrollo el programa eje4.py para calcular el RTT para los protocolos UDP y TCP. Para su correcta ejecución deben cargarse los archivos IPsUDP.txt y IpsTCP.txt con la lista de ip separadas por “;” que se desea probar.

Opciones de ejecución:

- para udp: python ejer4.py --udp [--segundos]
- para tcp: python ejer4.py --tcp [--segundos]
- para ambos: python ejer4.py --udp --tcp [--segundos]

UDP

****
Estadisticas IP: 195.46.39.40
Paquetes: enviados=13, Recibido=12, Perdido=1
Promedio=239 ms
****
****
Estadisticas IP: 195.46.39.40
Paquetes: enviados=13, Recibido=12, Perdido=1
Promedio=239 ms
****
****
Estadisticas IP: 208.67.222.222
Paquetes: enviados=18, Recibido=17, Perdido=1
Promedio=169 ms
****
****
Estadisticas IP: 208.67.220.220
Paquetes: enviados=18, Recibido=18, Perdido=0
Promedio=169 ms
****

TCP

****
Estadisticas IP: www.google.com
Paquetes: enviados=50, Recibido=50, Perdido=0
Promedio=60 ms
****
****

Estadisticas IP: www.facebook.com
Paquetes: enviados=45, Recibido=45, Perdido=0
Promedio=66 ms
****
****
Estadisticas IP: www.unlu.edu.ar
Paquetes: enviados=59, Recibido=59, Perdido=0
Promedio=51 ms
****
****
Estadisticas IP: www.yahoo.com
Paquetes: enviados=11, Recibido=11, Perdido=0
Promedio=279 ms
****

Se puede observar que en promedio los tiempos de UDP son mayores que los de TCP, pero la mayor demora se produce contra el servidor web www.yahoo.com utilizando una conexión TCP, se puede deducir que el factor determinante en los retardos es la distancia.

5) Se utilizo el algoritmo de cifrado AES para resolver la problemática de encriptacion de la autentificación que plantea el ejercicio.
Con AES se implemento una encriptación simétrica, donde los programas cliente y servidor acuerdan una clave pública para encriptar sus mensajes, en nuestro caso solo para la etapa de autentificación del sistema, el resto de los mensajes viajan en texto plano, a diferencia del protocolo SSH donde toda la comunicación entre el cliente y servidor es segura.

9) Para el desarrollo del chat un requisito que se presento, fue la necesidad de mantener en el servidor una lista dińamica de socket, con la finalidad de distribuir los mensajes y poder aceptar nuevas conexiones de los clientes, la solución fue implementada utilizando la llamada de sistema *select* para recuperar las conexiones disponibles para luego realizar el tratamiento correspondiente.

14) Se realizo el desarrollo de los nodos master, mapper y reducer y se realizaron pruebas con el el texto del Quijote de Cervantes.

Modo de Uso en consola:
Setear variables de configuracion en el archivo 'config.ini'

Ejecutar los mappers
Toma variable de conexion (host, port) por parametro, deben coincidir con las seteadas en el archivo de configuracion para poder establecer comunicacion con los otros nodos.

#nodos mappers
python nodo_mapper.py [-h] host port

Ejecutar reducer
Toma la configuracion desde archivo 'config.ini'.
Sin variables por parametros

#nodo reducer
python nodo_reducer.py

Ejecutar master
Toma la configuracion desde archivo 'config.ini'.
Debe ingresarse por parametro el archivo entrada a procesar y el archivo salida donde devolvera el resultado.

#nodo master
nodo_master.py [-h] in_file out_file