Proyecto 1

“Cliente Echo”

Alumno: Santiago Rubén Barboza

Lu:106007

Materia: Redes de Computadoras 2016

Consigna

Empleando el lenguaje de programación Python, deberá implementar un programa cliente que permita conectarse a servidores Echo. El mismo deberá ejecutarse desde consola y deberá brindar opciones para establecer la comunicación con el servidor mediante los protocolo TCP y UDP.

Formato

Para llamar al programa se debe respetar el siguiente Formato:

clienteEcho <server:puerto> <-p tcp|udp> [-h]

Donde el parámetro -h es opcional y los otros 2 obligatorios. El significado de las diversas opciones del programa es el siguiente:

- server:nombre del servidor al cual se debe conectar, especificando el puerto adecuado.
- -p tcp|udp: tipo de protocolo de transporte utilizado para la conexión, solo acepta las palabras “udp” y “tcp”
- -h:Modo ayuda en línea: Si se especifica el parámetro -h en algún lugar de la línea de comandos, el programa sólo deberá mostrar una pequeña ayuda por pantalla, es decir tiene precedencia sobre cualquier otra opción. La cual consiste en listar las opciones de invocación del programa (sintaxis) y su semántica (con una aclaración sintética en base a la que se brinda en este enunciado)

Resolución

Si lee el parámetro – h se mostrara en pantalla el mensaje de ayuda correspondiente generado con las funciones de la librería argparse y se terminara el programa. Sino se leerá del parámetro server:puerto tanto la dirección del servidor (dirección IP o dirección de nombre simbólico) como el puerto del servidor que es un numero.

Luego si el parámetro p es udp se creara un socket de datagramas (Socket UDP) , se leerá un mensaje ingresado por consola, y se lo enviara al puerto y servidor destino especificados en la llamada. Por ultimo se esperara la recepción de la respuesta y cuando esta llegue además de mostrarla se la comparara con el mensaje leído y se aclarara si son iguales o distintos.

En cambio si el parámetro p es tcp se creara un socket de Stream (Socket TCP) , se conectara al mismo al socket que determine la respuesta de solicitarle un socket TCP al socket de Bienvenida que escucha en el puerto del servidor ingresado por consola. A partir de entonces esos puertos estarán conectados punto a punto, entonces se leerá un mensaje ingresado por consola mientras no se corte la transmicion, y se lo enviara al puerto conectado a nuestro puerto TCP. Luego se esperara la recepción de la respuesta y cuando esta llegue además de mostrarla se la comparara con el mensaje leído y se aclarara si son iguales o distintos. Finalmente cuando se interrumpe la transmicion se cierra la conexión del socket

Procedimientos

Para realizar este proyecto definí 2 Procedimientos, el primero se encarga de realizar la lectura del mensaje udp a enviar y de realizar tanto el envío como la recepción y el procesamiento del mensaje (determinar el buen funcionamiento del echo). Para ello necesita 2 parámetros el primero es un string con el nombre del server y el segundo es un numero que representa el numero de puerto destino.

La segunda tiene un funcionamiento análogo pero para la operatoria con TCP, al igual que la anterior requiere 2 parámetros el primero es un string con el nombre del server y el segundo es un numero que representa el numero de puerto de bienvenida del server destino.

En el programa principal se realiza el procesamiento de los argumentos y se decido dependiendo del tipo de la conexión a cual procedimiento invocar.

Implementación

El programa esta implementado en Python 2.7 que es la versión de Python instalada en las maquinas virtuales del laboratorio. Para poder aclarar la legibilidad del proyecto y no llenarlo de códigos de control utilice 2 librerías: Argparse y re. Estas librerías son librerías estándar en Python 2.7. La primera me permite definir de una manera muy simple los argumentos con los que se puede invocar al programa (creando de una manera transparente la opción de -h para la ayuda) y la segunda me permite validar el formato del parámetro server:port sin mucha dificultad mediante el uso de expresiones regulares.

El algoritmo general es el siguiente:

Si el parámetro -h esta presente se muestra la ayuda del programa.

Sino

    Se procesa el Server:port y se lo separa en un valor de server y uno de port

    Si la conexión es udp

        Se realiza un envio_udp con el servidor y el puerto destino

    Sino (TCP)

        Se realiza un envio_tcp con el servidor y el puerto destino

envio_udp(Server:String, Port:Int)

    Se crea un Socket de tipo UDP (de datagrama)

    Se lee un mensaje por consola

    Se lo envía al puerto del servido destino

    Se espera la recepción de la respuesta del servidor y se la muestra

    Si el mensaje enviado es igual al recibido

        El funcionamiento del server echo udp es correcto

    Sino

        El funcionamiento del server echo udp es incorrecto

envio_tcp(Server:String, Port:Int)

Se crea un Socket de tipo TCP (de Stream)

Se envía una solicitud al server de bienvenida que nos asigna una conexión con un socket del servidor que usamos para enviar mensajes

Mientras no interrumpan

Se lee un mensaje por consola y se lo envía

Se espera la recepción de la respuesta del servidor y se la muestra

Si el mensaje enviado es igual al recibido

El funcionamiento del server echo tcp es correcto

Sino

El funcionamiento del server echo tcp es incorrecto

Se cierra la conexión con el puerto TCP

Código del programa

```
#.....Librerias Importadas.....
import argparse                 #Para la lectura de Argumentos
import re                       #Para la busqueda del patron en server:port
import sys
from socket import *

#.....Definicion de Procedimientos.....

#Envio de Echo UDP (-String Server,-Int Port)
def envio_udp(Server,Port):
    clientSocket = socket(AF_INET, SOCK_DGRAM)
    #Creo un Socket UDP
    print "Conexion UDP al Servidor: %s al Puerto: %s" %(Server,Port)
    mensaje= raw_input("(UDP) Ingrese el mensaje a enviar: ")                    #Leo mensaje a enviar
    clientSocket.sendto(mensaje,(Server,Port))
```

```
        #Envio el mensaje al (server, puerto)
        mensaje2,servaddr = clientSocket.recvfrom(2048)
        #Recibo el mensaje
        print "(UDP) Se recibio el mensaje: "+mensaje2
        #Imprimo el mensaje recibido
        if(mensaje == mensaje2):
                print "(UDP) El mensaje recibido es igual al enviado"
        else:
                print "(UDP) El mensaje recibido es distinto al enviado"


#Envio de echo TCP  (-String Server,-Int Port)
def envio_tcp(Server,Port):
        clientSocket = socket(AF_INET, SOCK_STREAM)
                        #Creo un Socket TCP
        clientSocket.connect((Server,Port))
                #Conecto el Socket con el socket TCP que asigne
        try:
                                        # el pedido de conexion al socket de Bienvenida
                while True:

                        print "Conexion TCP al Servidor: %s al Puerto: %s" %(Server,Port)
                        mensaje= raw_input("(TCP) Ingrese el mensaje a enviar: ")
        #Leo mensaje a enviar
                        clientSocket.send(mensaje)
                                #Envio el mensaje
                        mensaje2= clientSocket.recv(2048)
                        #Leo el mensaje que me envia el server
                        print "(TCP) Se recibio: "+mensaje2
                        if(mensaje == mensaje2):
                                print "(TCP) El mensaje recibido es igual al enviado"
                        else:
                                print "(TCP) El mensaje recibido es distinto al enviado"


        except KeyboardInterrupt:
```

```
        print "Finalizo la conexion TCP"
        clientSocket.close()
            #Cierro la conexion del Socket  TCP

#.....................................Codigo del Programa Ppal.....................................

#Generacion de ayuda y control de parametros obligatorios
parser = argparse.ArgumentParser()
parser.add_argument('server',metavar="server:port",type=str, nargs=1, help=' nombre del servidor al cual se debe conectar, especificando el puerto adecuado')
parser.add_argument("-p", metavar="<tcp|udp>", choices=['udp', 'tcp'],help="tipo de protocolo de transporte utilizado para la conexión",required=True)
args = parser.parse_args()

#Reconocer server:port
patron = re.compile('([0-9a-zA-Z.]+):([0-9]+)')       # En el nombre del server pueden ir minusculas,mayusculas y puntos
matcher = patron.search(args.server[0])                    #  en cambio en el puerto solo numeros
server=matcher.group(1)              #Primera Parte de la Expresion
port=matcher.group(2)                #Segunda Parte de la Expresion

#Reconocer tipo de conexion
if (args.p == "udp") :
        envio_udp(server,int(port))
else:
        envio_tcp(server,int(port))
```

Diagrama de Flujo del Programa Gral