LAB 273 Laboratorio de Telemática

Mini proyecto 2

Programación de Sockets

Nombres:

Helen Carolina Castillo M.

John Castillo Valencia

Introducción

- Un **socket** (enchufe), es un método para la comunicación entre un programa del **cliente** y un programa del **servidor** en una red. Un socket se define como el punto final en una conexión. Los sockets se crean y se utilizan con un sistema de peticiones o de llamadas de función a veces llamados interfaz de programación de aplicación de sockets (API, application programming interface).

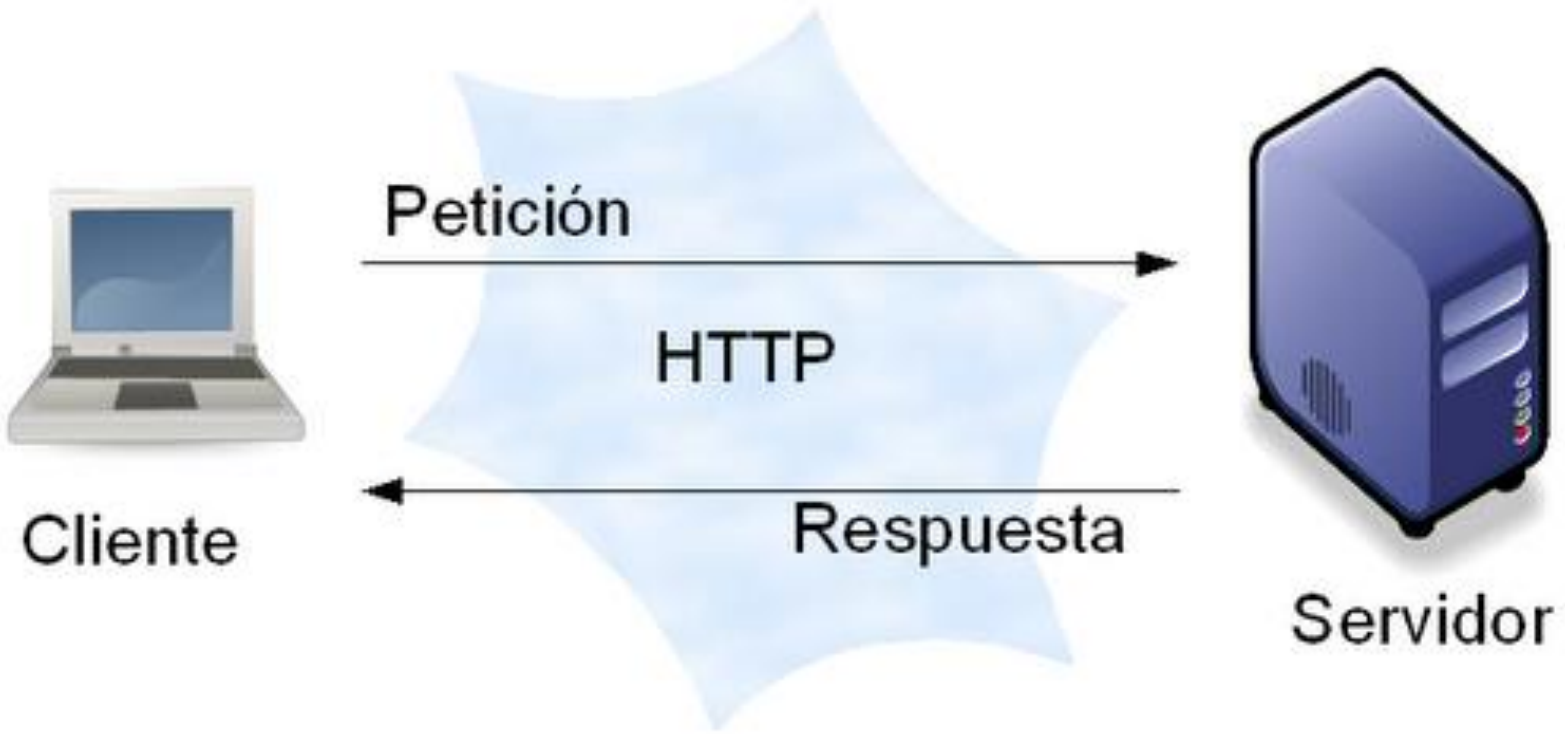
Introducción

- Un **Socket** es también una dirección de Internet, combinando una dirección IP (la dirección numérica única de cuatro partes que identifica a un ordenador particular en Internet) y un número de puerto (el número que identifica una aplicación de Internet particular, como SMTP, FTP, o HTTP).
- El **Protocolo de Control de Transmisión** (Transmission Control Protocol en inglés o TCP) es el método de comunicación de datos por defecto entre distintos dispositivos, a través de una red. Este establece y mantiene una conexión entre el emisor y el receptor durante el proceso de transferencia.

Enunciado

- Desarrollar una aplicación **Cliente/Servidor usando TCP** para jugar el **Juego del ahorcado**. El servidor debe seleccionar una palabra aleatoriamente de las que tenga registradas. El cliente tratará de adivinar la palabra. El servidor deberá proporcionar al cliente: las letras que adivinó correctamente, el número y la posición de las letras que faltan para completar la palabra y las letras incorrectas enviadas. Después que el usuario venció todos los intentos se le debe dar a conocer cuál era la palabra que no pudo adivinar.

Objetivos propuestos



Objetivos propuestos

- El **primer objetivo** es desarrollar un **programa servidor** que permita conexiones TCP para interactuar con el “juego de ahorcado” cuya lógica es parte del programa servidor.
- El **segundo objetivo** es desarrollar un **programa cliente** que realice una conexión con el programa servidor cuya lógica es simplemente establecer una conexión, enviar y recibir mensajes.

Software y bibliotecas requeridos

Cliente

Cliente

```python
import socket

host = '192.168.0.1'
port = 1233

ClientSocket = socket.socket()
print('Conectando...')
try:
    ClientSocket.connect((host, port))

    Response = ClientSocket.recv(2048)
    print(Response.decode('utf-8'))

    while True:
        Input = input('Prompt: ')
        if Input:
            ClientSocket.send(str.encode(Input))
            Response = ClientSocket.recv(2048)
            if not Response: break
            print(Response.decode('utf-8'))

except socket.error as e:
    print(str(e))

ClientSocket.close()
```

Cliente

Símbolo del sistema - python mul-client.py

```
D:\helen\jsockets\python>python mul-client.py
Conectando...
Bienvenido al SERVIDOR AHORCADO de 273...

Ingresa tu nombre:
Prompt: helen
Listo para empezar? S/N:
Prompt: s


    +---++
    |   ||
        ||
        ||
        ||
        ||
       ///////////

 _   _   _   _   _   _   _


----------------------------------------
Ingresa una letra:
Prompt:
```

Software y bibliotecas requeridos


```
66
67      +---++
68      |   ||
69      O   ||
70     /|\  ||
71      |   ||
72     / \  ||
73          ///////////''', '''
74  ''']
75
```

Servidor

```
76  WORDS = [
77      'lavadora',
78      'secadora',
79      'sofa',
80      'gobierno',
81      'diputado',
82      'democracia',
83      'computadora',
84      'teclado',
85      'pesadilla',
86      'telematica',
87      'umsa',
88      'informatica',
89      'bolivia'
90  ]
91
92  def random_word():
93      idx = random.randint(0, len(WORDS) - 1)
94      return WORDS[idx]
```

Servidor

```
96  def client_handler(connection):
97      connection.sendall(str.encode('Bienvenido al SERVIDOR AHORCADO de 273...\n\nIngre
98      data = connection.recv(2048)
99      nombre = data.decode('utf-8')
100
101     print('Ha iniciado un juego:', nombre)
102
103     connection.sendall(str.encode('¿Listo para empezar? S/N:'))
104     data = connection.recv(2048)
105     resp = data.decode('utf-8')
106     if resp.upper() != 'S': connection.close()
107
108     word = random_word()
109     hidden_word = [" _ "] * len(word)
110     tries = 0
111     sents = []
112
113     img = display_board_str(sents, hidden_word, tries)
114     reply = f'{img}' + '\nIngresa una letra:'
115     connection.sendall(str.encode(reply))
116
117     while True:
118         data = connection.recv(2048)
119         message = data.decode('utf-8')
120         if message.upper() == 'BYE': break
121
```

Servidor

```
117     while True:
118         data = connection.recv(2048)
119         message = data.decode('utf-8')
120         if message.upper() == 'BYE': break
121 
122         current_letter = str(message)
123         if len(current_letter) == 1:
124 
125             # Logica del juego
126 
127         else:
128             connection.sendall(str.encode('Ingresa solo una letra:'))
129 
130     connection.close()
```

DEMO

Conclusiones

Hemos experimentado el funcionamiento del protocolo **TCP** para la comunicación y transferencia de datos entre dos programas uno como servidor y otro como cliente.

El lenguaje de programación **Python** utilizado para el desarrollo de este proyecto tiene la capacidad y la característica de ser multiplataforma, es decir, puede ejecutarse tanto en Sistemas **Windows** o **GNU/Linux** y cuenta además con las librerías necesarias para realizar aplicaciones distribuidas para la interacción a través de la red con la API de SocKets tanto para TCP como UDP.