Nombre: Luis Alberto Vargas González.

Fecha: 1/02/2025.

U3 A1: Chat P2P.

Clase: Sistemas Distribuidos.

Maestría en Ciberseguridad.

Descripción de programas.

En esta práctica se desarrolla un chat tipo P2P con una interfaz gráfica en donde podemos ver visualmente como se envían mensajes los nodos 1:1, cabe destacar que se reutilizó el código de funciones de iniciar servidor de TCP e iniciar cliente de TCP para que cada nodo que sea creado sean servidor y cliente al mismo tiempo. Así mismo, en esta práctica se usaron 2 programas para hacer funcionar el sistema P2P, en el primer programa se definen las funciones de TCP previamente programadas y en el segundo se construye la interfaz gráfica.

Código de servidor y cliente TCP.

```
import socket
import threading

# Lista de nodos (IP, Puerto) en la red P2P
NODOS = [
("127.0.0.1", 56432), # Nodo 1
("127.0.0.1", 56433), # Nodo 2
]

# Función de servidor TCP para recibir mensajes
def iniciar_servidor_tcp(mi_ip, mi_puerto, callback_recibir_mensaje):
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_socket:
server_socket.bind((mi_ip, mi_puerto))
server_socket.listen()
print(f"Servidor TCP escuchando en {mi_ip}:{mi_puerto}")
```

```
while True:
conn, addr = server_socket.accept()
print(f"Conexión establecida con {addr}")
with conn:
while True:
data = conn.recv(1024)
if not data:
print(f"Cliente {addr} desconectado.")
break
mensaje = data.decode('utf-8')
callback_recibir_mensaje(f"Mensaje recibido de {addr}: {mensaje}")

# Función de cliente TCP para enviar mensajes
def iniciar_cliente_tcp(mi_ip, mi_puerto, mensaje):
print(f"Enviando mensaje a los nodos: {mensaje}")
# Enviar mensaje a todos los nodos en la red
for ip, puerto in NODOS:
if (ip, puerto) != (mi_ip, mi_puerto): # No enviarse a sí mismo
try:
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client_socket:
client_socket.connect((ip, puerto))
client_socket.sendall(mensaje.encode('utf-8'))
print(f"Mensaje enviado a {ip}:{puerto}")
except ConnectionRefusedError:
print(f"No se pudo conectar a {ip}:{puerto}")
```

Nota: en este programa al inicio de este es donde se listan los nodos que serán creados posteriormente. Se definen las ip y puertos de dichos nodos en una matriz.

Código de interfaz gráfica.

```
import tkinter as tk
from tkinter import scrolledtext
import threading
import argparse
import chat_p2p # Importamos el archivo de lógica P2P

# Crear la interfaz gráfica
def crear_interfaz(mi_ip, mi_puerto):
ventana = tk.Tk()
ventana.title("Chat P2P")
```

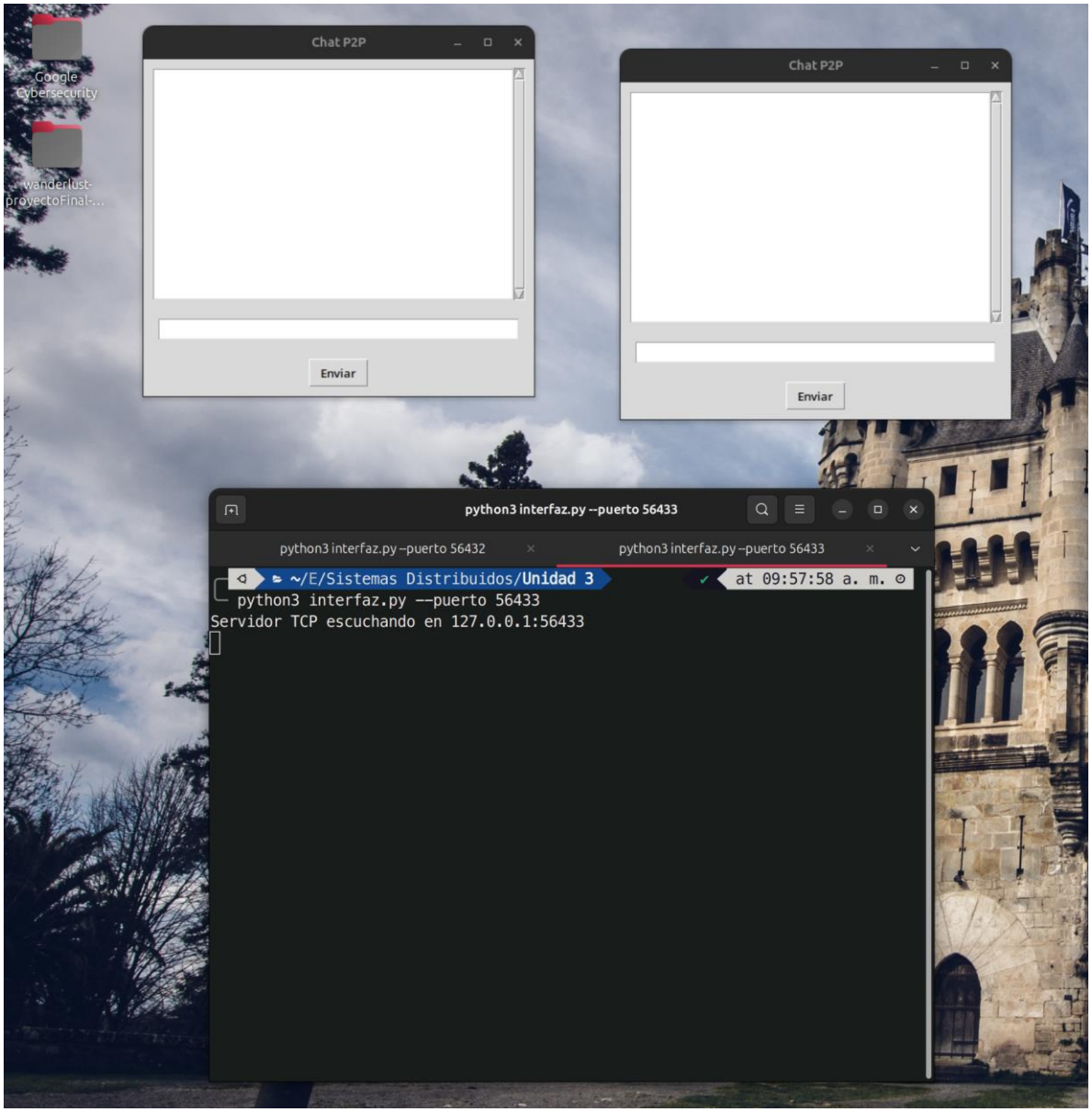
```
# Crear área de texto para mostrar mensajes
text_area = scrolledtext.ScrolledText(ventana, width=50, height=15,
wrap=tk.WORD)
text_area.grid(row=0, column=0, padx=10, pady=10)
text_area.config(state=tk.DISABLED) # Hacer que el área de texto sea de solo
lectura

# Crear campo de entrada para el mensaje
entry = tk.Entry(ventana, width=50)
entry.grid(row=1, column=0, padx=10, pady=10)

# Crear botón para enviar mensaje
def enviar_mensaje():
mensaje = entry.get() # Obtener el mensaje desde el campo de entrada
if mensaje:
text_area.insert(tk.END, f"Tú: {mensaje}\n")
text_area.yview(tk.END) # Desplazar hacia abajo automáticamente
entry.delete(0, tk.END) # Limpiar el campo de entrada
threading.Thread(target=chat_p2p.iniciar_cliente_tcp, args=(mi_ip, mi_puerto,
mensaje)).start()

boton_enviar = tk.Button(ventana, text="Enviar", command=enviar_mensaje)
boton_enviar.grid(row=2, column=0, padx=10, pady=10)

# Función para recibir mensajes del servidor y actualizarlos en la interfaz
def recibir_mensaje(mensaje):
text_area.config(state=tk.NORMAL)
text_area.insert(tk.END, f"{mensaje}\n")
text_area.yview(tk.END)
```

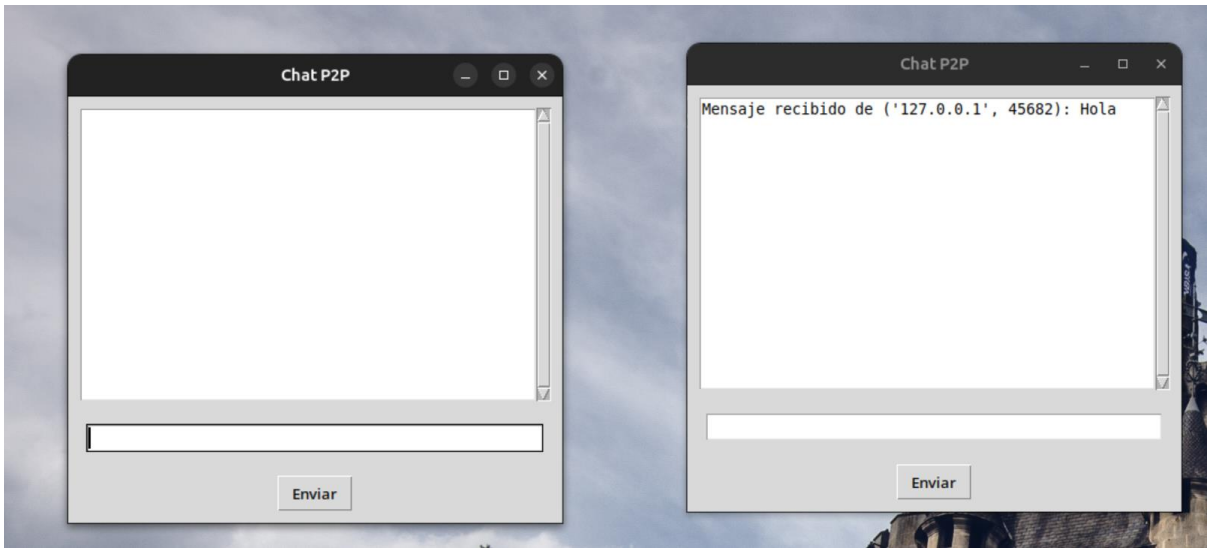
```
text_area.config(state=tk.DISABLED)

# Crear e iniciar el hilo del servidor
hilo_servidor = threading.Thread(target=chat_p2p.iniciar_servidor_tcp,
args=(mi_ip, mi_puerto, recibir_mensaje), daemon=True)
hilo_servidor.start()

# Ejecutar la interfaz gráfica
ventana.mainloop()

if __name__ == "__main__":
# Argumentos de línea de comandos para configurar IP y puerto
parser = argparse.ArgumentParser(description="Nodo P2P tipo chat.")
parser.add_argument("--puerto", type=int, required=True, help="Puerto en el
que este nodo escuchará conexiones.")
args = parser.parse_args()

MI_IP = "127.0.0.1" # Dirección IP fija para este ejemplo
MI_PUERTO = args.puerto

# Crear e iniciar la interfaz gráfica
crear_interfaz(MI_IP, MI_PUERTO)
```

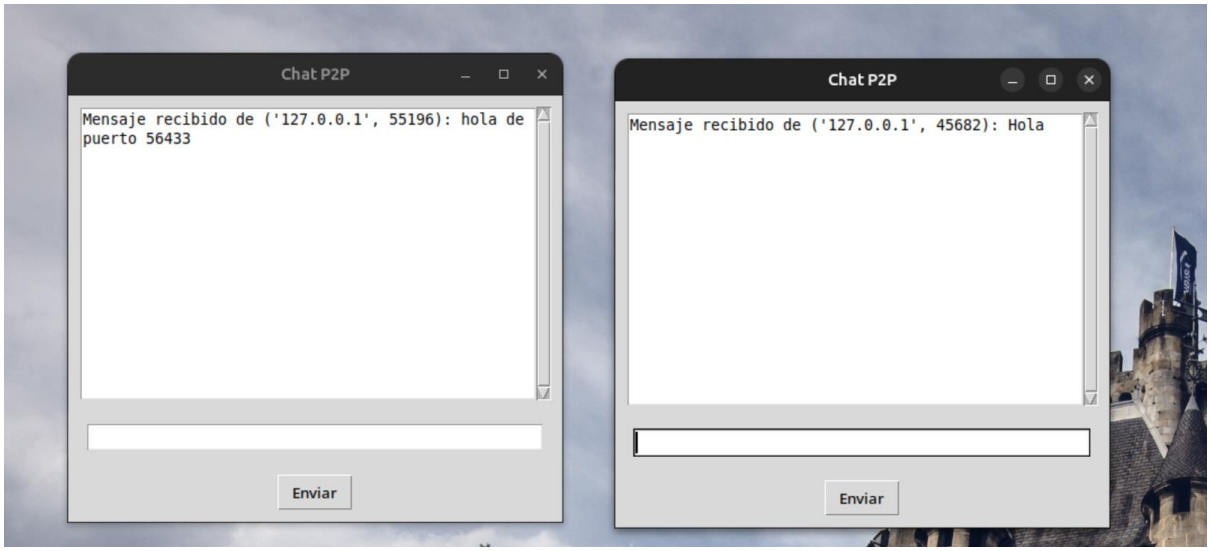
En este código se importa las funciones de TCP del otro programa y además se definen los hilos que nos permitirán iniciar el cliente y el servidor, así como la instanciación del método main que nos permitirá mandar llamar la función de interfaz gráfica la cual contiene 3 funciones y en la cual previamente se inicializaron dichas

funciones en las cuales se crean las ventanas de cada nodo chat p2p, se crean los botones de enviar y campos de texto en esas ventanas, la función para recibir mensajes del servidor y actualizar en la pantalla dichos nuevos mensajes

Evidencia de funcionamiento.







Como se puede notar, ambos nodos pueden enviar y recibir mensajes del uno al otro, para diferenciar el nodo 1, se envía “hola“ al 2 y en el 2 se envía al nodo 1 “hola de puerto 56433” el cual es el puerto del nodo 2.

Conclusiones

Al finalizar esta actividad nos pudimos percatar que la realización de esta conlleva un esfuerzo mediano pues el algoritmo en si es muy sencillo de implementar ya que ya tenemos implementado la mitad del mismo al reusar TCP por lo tanto es fácil de entender y de modificar en caso de ser requerido. La creación de la interfaz gráfica fue lo más complicado como tal ya que personalmente yo solo había creado dichas interfaces en Java y no en Python.