import tkinter as tk

from tkinter import filedialog, messagebox, ttk

import threading

from sender import send\_file

from receiver import receive\_file

import os

class UDPFileTransferApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("UDP File Transfer Tool")

self.root.geometry("520x560")

self.root.resizable(False, False)

self.file\_queue = []

self.paused = False

self.cancelled = False

self.sending = False

tk.Label(root, text="UDP File Transfer", font=("Arial", 18, "bold")).pack(pady=10)

# --- Send Section ---

sender\_frame = tk.LabelFrame(root, text="Send Files", padx=10, pady=10)

sender\_frame.pack(fill="x", padx=20, pady=10)

self.file\_listbox = tk.Listbox(sender\_frame, height=5, width=50)

self.file\_listbox.grid(row=0, column=0, columnspan=3, padx=5, pady=5)

tk.Button(sender\_frame, text="Add Files", command=self.browse\_files).grid(row=1, column=0, pady=5)

tk.Button(sender\_frame, text="Clear List", command=self.clear\_file\_list).grid(row=1, column=1, pady=5)

tk.Label(sender\_frame, text="Receiver IP:").grid(row=2, column=0, sticky="e", pady=5)

self.ip\_entry = tk.Entry(sender\_frame)

self.ip\_entry.insert(0, "127.0.0.1")

self.ip\_entry.grid(row=2, column=1, padx=5)

tk.Label(sender\_frame, text="Port:").grid(row=3, column=0, sticky="e", pady=5)

self.port\_entry = tk.Entry(sender\_frame)

self.port\_entry.insert(0, "5005")

self.port\_entry.grid(row=3, column=1, padx=5)

button\_frame = tk.Frame(sender\_frame)

button\_frame.grid(row=4, column=0, columnspan=3, pady=10)

tk.Button(button\_frame, text="Start Sending", command=self.send\_thread, bg="lightblue").grid(row=0, column=0, padx=5)

tk.Button(button\_frame, text="Pause", command=self.pause\_transfer, bg="orange").grid(row=0, column=1, padx=5)

tk.Button(button\_frame, text="Resume", command=self.resume\_transfer, bg="lightgreen").grid(row=0, column=2, padx=5)

tk.Button(button\_frame, text="Cancel", command=self.cancel\_transfer, bg="red", fg="white").grid(row=0, column=3, padx=5)

# --- Receive Section ---

receiver\_frame = tk.LabelFrame(root, text="Receive File", padx=10, pady=10)

receiver\_frame.pack(fill="x", padx=20, pady=10)

tk.Label(receiver\_frame, text="Port:").grid(row=0, column=0, sticky="e")

self.recv\_port\_entry = tk.Entry(receiver\_frame)

self.recv\_port\_entry.insert(0, "5005")

self.recv\_port\_entry.grid(row=0, column=1, padx=5)

tk.Button(receiver\_frame, text="Choose Folder & Receive", command=self.receive\_thread, bg="lightgreen").grid(row=1, column=1, pady=10)

# --- Status and Progress ---

self.status\_label = tk.Label(root, text="", fg="green")

self.status\_label.pack(pady=5)

self.progress = ttk.Progressbar(root, orient="horizontal", length=400, mode="determinate")

self.progress.pack(pady=5)

def browse\_files(self):

files = filedialog.askopenfilenames()

if files:

self.file\_queue.extend(files)

for file in files:

self.file\_listbox.insert(tk.END, os.path.basename(file))

def clear\_file\_list(self):

self.file\_queue.clear()

self.file\_listbox.delete(0, tk.END)

def send\_thread(self):

if not self.file\_queue:

self.set\_status("No files selected.", error=True)

return

self.paused = False

self.cancelled = False

self.sending = True

threading.Thread(target=self.send\_files\_queue, daemon=True).start()

def send\_files\_queue(self):

ip = self.ip\_entry.get()

port = int(self.port\_entry.get())

for i, file in enumerate(self.file\_queue):

if self.cancelled:

break

try:

self.set\_status(f"Sending: {os.path.basename(file)} ({i+1}/{len(self.file\_queue)})")

self.progress["value"] = 0

def update\_progress(sent, total):

if self.cancelled:

raise Exception("Transfer cancelled by user.")

while self.paused:

self.set\_status("Paused...")

threading.Event().wait(0.1)

percent = (sent / total) \* 100

self.progress["value"] = percent

send\_file(file, ip, port, progress\_callback=update\_progress)

except Exception as e:

if "cancelled" in str(e).lower():

self.set\_status("Transfer cancelled.", error=True)

break

else:

self.set\_status(f"Error sending file: {str(e)}", error=True)

if not self.cancelled:

self.set\_status("All files sent!")

self.sending = False

self.progress["value"] = 0

def receive\_thread(self):

threading.Thread(target=self.receive\_file, daemon=True).start()

def receive\_file(self):

save\_dir = filedialog.askdirectory(title="Choose Folder to Save Received File")

if not save\_dir:

return

port = int(self.recv\_port\_entry.get())

try:

self.set\_status("Receiving file...")

self.progress["value"] = 0

def update\_progress(received, total):

percent = (received / total) \* 100

self.progress["value"] = percent

receive\_file(save\_dir, port, progress\_callback=update\_progress)

self.set\_status("File received successfully!")

except Exception as e:

self.set\_status(f"Error: {str(e)}", error=True)

def pause\_transfer(self):

if self.sending:

self.paused = True

self.set\_status("Paused...")

def resume\_transfer(self):

if self.paused:

self.paused = False

self.set\_status("Resuming...")

def cancel\_transfer(self):

if self.sending:

self.cancelled = True

self.paused = False

self.set\_status("Cancelling...")

def set\_status(self, message, error=False):

self.status\_label.config(text=message, fg="red" if error else "green")

# --- Start App ---

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = UDPFileTransferApp(root)

root.mainloop()