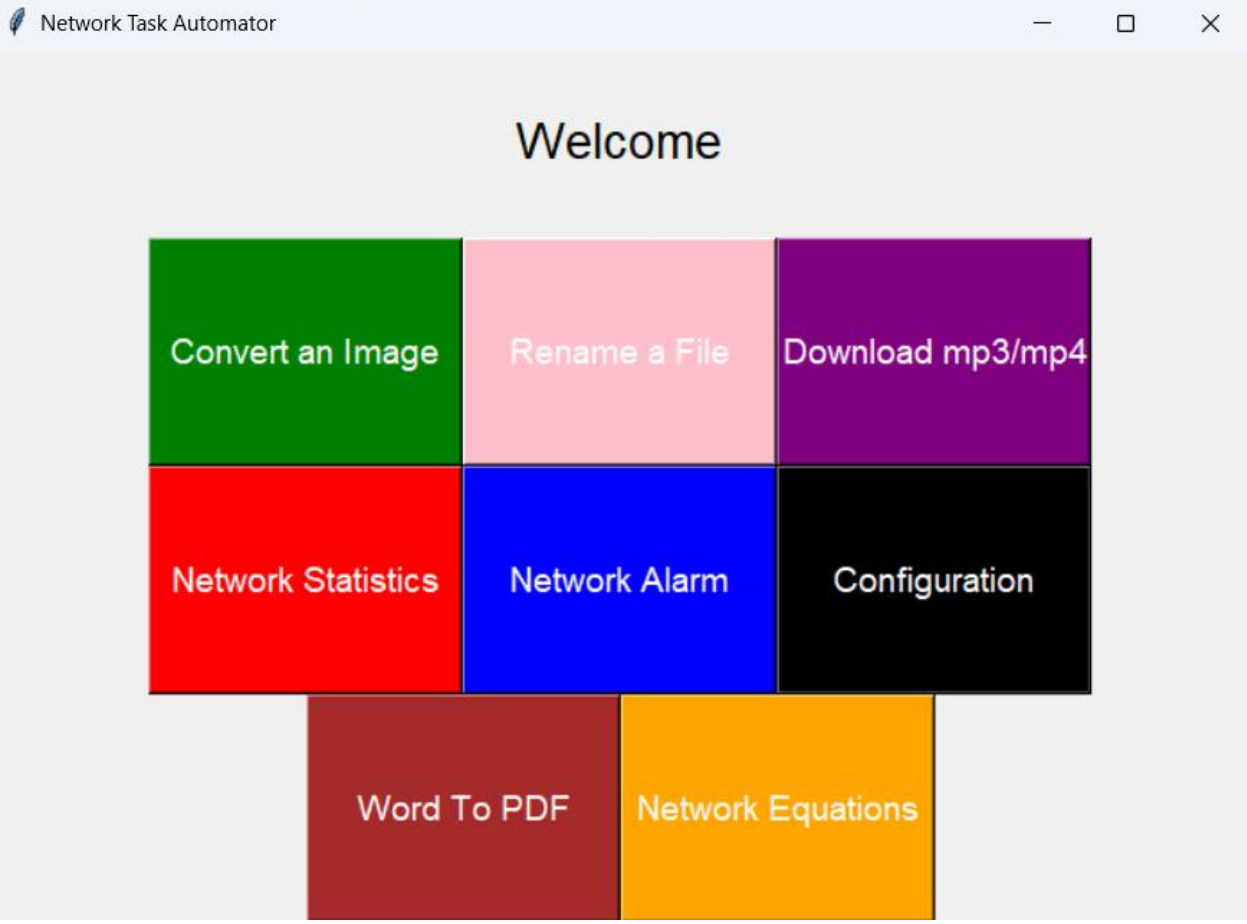
Network Task Automator



Our Application is Network task Automator simply in this project our server has to do some tasks which we will talk about them briefly and return it to our client which is:

1. Performing automated network tasks and network configuration
2. Network Alarm
3. Network Statistics
4. Converting images, renaming files
5. Performing network equations
6. Word to pdf conversion
7. Audio/Video file transmission

First before talking about the tasks our server is a multithreaded server which has many ports every port is binded to a special socket for every task we have special port for it

```
#PORTS
StatPort = 13000
AudioPort = 13001
RenamePort = 13002
EquationsPort = 5050
ConfigurationPort = 5060
AlarmPort = 12000
ImagePort = 12001
Word2PdfPort = 12345
```

Every socket is binded to its port number

```
RenameSocket = socket(AF_INET, SOCK_STREAM)
RenameSocket.bind((ServerIP, RenamePort))
RenameSocket.listen(5)

#Socket for AUDIO/VIDEO server
AudioSocket = socket(AF_INET, SOCK_STREAM)
AudioSocket.bind((ServerIP, AudioPort))
AudioSocket.listen(5)

#Socket for NETWORK STATISTICS server
StatSocket = socket(AF_INET, SOCK_STREAM)
StatSocket.bind((ServerIP, StatPort))
StatSocket.listen(5)
```

And then we create a thread for every task

```
#Statistics thread
Stat_Thread = Thread (target= Connection_handler, args = (StatSocket, Statistics_handler, "Statistics port 13000"))
Stat_Thread.start()

#Audio thread
Audio_Thread = Thread (target= Connection_handler, args = (AudioSocket, Audio_handler, "Audio port 13001"))
Audio_Thread.start()
```

1- Network Configuration :

We have implemented a function on the server which is “ Configuration_client ” that acts like a DHCP server configuration that allow the server to to assign the clients from [C1-Cn] in our code to be assigned ip addresses

from the server like the dynamic host configuration protocol this task can be implanted at port number = 5060

```
def ips(start, end):
    '''Return IPs in IPv4 range, inclusive.'''
    start_int = int(ip_address(start).packed.hex(), 16)
    end_int = int(ip_address(end).packed.hex(), 16)
    # return ip_address(i)
    return [ip_address(ip).exploded for ip in range(start_int, end_int)]
```

```
def Configuration_client(conn):
    global i
    assigned_ip = ip_addresses[i]
    i=i+1
    conn.sendall(assigned_ip.encode('utf-8'))
    conn.close
```

2. Network Alarm :

In this task we receive the port number from the client to check whether if it is closed or opened by using nmap port scanner at port number = 12000

```python
def Alarm_handler(connectionSocket):
    receivedPort = connectionSocket.recv(2048)
    #Receive port number from client to check its state
    PortNumber = receivedPort.decode()
    print(f"Received {PortNumber}")
    #Instantiate a PortScanner object
    scanner = nmap.PortScanner()
    res = scanner.scan(ServerIP, PortNumber)
    #Get the state -> open/closed
    res = res['scan'][ServerIP]['tcp'][int(PortNumber)]['state']
    #Declare a network alarm in case the port is closed
    if (res == 'closed'):
        print(f"NETWORK ALARM!!! PORT {PortNumber} is {res}")
    elif (res == "open"):
        print(f"PORT {PortNumber} is {res}")
    connectionSocket.close()
```

3. Network Statistics :

In this task the server returns to the clients the statistics of the ports in range whether if it is opened or close also using the nmapScanner and then we save the statistics in a file called stat_data at port number = 13000

```
def Statistics_handler(connectionSocket):
    command = connectionSocket.recv(1024).decode()
    #Name of the file
    statfile = "Network_Statistics.txt"
    #Range of ports to be scanned
    begin = 20
    end = 25
    #Instantiate a PortScanner object
    scanner = nmap.PortScanner()
    txtfile = open (statfile, "w")
    #Loop to scan the range of ports
    for i in range(begin,end+1):
        #Scan the target port
        res = scanner.scan(ServerIP,str(i))
        #The result is a dictionary containing several informati
        res = res['scan'][ServerIP]['tcp'][i]['state']
        result = f'port {i} is {res}'
        txtfile.write(result)
        txtfile.write("\n")
    txtfile.close()
    #File transmission
    with open (statfile,'rb') as file:
        stat_data = file.read()
    connectionSocket.sendall(stat_data)
    connectionSocket.close()
```

4. Converting images, renaming files

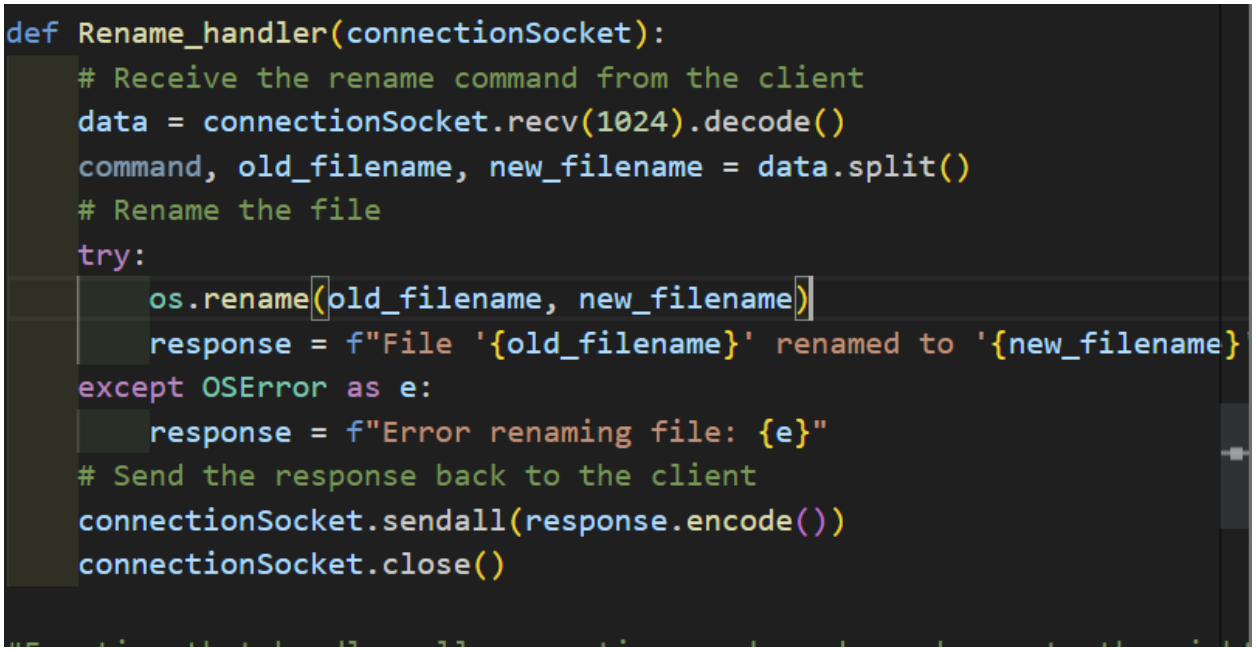
Firstly there is two different functions for handling every task

In the image conversion handler the server receives from the user Image name , Image path and Image Format to be converted then we convert it by decoding the user inputs and finally remove the old image at port number=12001

```
#Function that handles IMAGE CONVERSION
def ImageConversion_handler(connectionSocket):
    ImageName = connectionSocket.recv(2048) #read image name
    ImagePath = connectionSocket.recv(2048) #read image path
    ImageFormat = connectionSocket.recv(2048) #read image format
    print(f"Received{ImageName}")
    print(f"Received {ImagePath}")
    print(f"Received {ImageFormat}")

    # ImageFormat = ImageFormat.decode("utf-8").upper()
    SavedImage = ImageName.decode() + '.' + ImageFormat.decode()
    # SavedImage = 'Converted.' + ImageFormat.decode()

    im = Image.open(ImagePath) #open image
    im.save(SavedImage)
    im.close()

    # Delete the original image file
    os.remove(ImagePath.decode())
    # im.save(SavedImage, format=ImageFormat)
    print("Image successfully converted!")


    # capitalizedSentence = ImagePath.decode("utf-8").upper()
    # connectionSocket.send(bytes(capitalizedSentence,"utf-8"))
    connectionSocket.close()
```

Second Renaming files

In this function we receive from the user string containing the command to know to rename or not and the old file name and the new file name at port number=13002



```
def Rename_handler(connectionSocket):
    # Receive the rename command from the client
    data = connectionSocket.recv(1024).decode()
    command, old_filename, new_filename = data.split()
    # Rename the file
    try:
        os.rename(old_filename, new_filename)
        response = f"File '{old_filename}' renamed to '{new_filename}
    except OSError as e:
        response = f"Error renaming file: {e}"
    # Send the response back to the client
    connectionSocket.sendall(response.encode())
    connectionSocket.close()
```

5. Performing network equations:

In this task our client sends to the receiver the total information to perform Total delay which is the summation of propagation delay and conversion delay at port number 5050

```python
#Function that handles NETWORK EQUATIONS
def Equations_handler(conn): # this function will run in parallel d
    connected = True
    count = 4
    transm = 0
    prop =0
    global ii
    while connected:
        msg_Length = conn.recv(HEADER).decode(FORMAT)  # decode -->
        # bec every time we send a message we need to encode it to
        if msg_Length: #if the message is not null bec if null we w
            msg_Length = int(msg_Length) # "msg_lenght " convert it
            msg =conn.recv(msg_Length).decode(FORMAT)

            #disconnect handling
            if msg == DISCONNECT_MESSAGE:
                connected=False

            if count>0 :
                # if msg != DISCONNECT_MESSAGE :
                    msg =int(msg)
                    messages.append(msg)
                    count = count - 1
            if count == 0:
                print("Length of bits  = ",messages[ii])
                print("Rate of Transimission = ",messages[ii+1])
                transm = messages[ii]/messages[ii+1]
```

```
        print ("The Transmission delay = ",transm)
        ##########################
        print("Distance  = ",messages[ii+2])
        print("Speed = ",messages[ii+3])
        prop = messages[ii+2]/messages[ii+3]
        print ("The Propagation delay = ",prop)
        count = 4
        ii=ii+4
```

6. Word to pdf conversion:

   In this class we take from the user the file12345 name then

   we convert the word to pdf at portnumber=

```
#Function that handles WORD2PDF CONVERSION
def Word2PDF_handler(connectionSocket):
    filename = connectionSocket.recv(1024).decode()
    print(filename)
    # filenamex=open("C:\\Users\\Habiba\\OneDrive\\Desktop\\networ
    pythoncom.CoInitialize()
    convert(filename)
    #print("Error converting the file:", str(e))
    connectionSocket.close()
```

7- Audio/Video file transmission:

In this function we receive the audio file and we send to the server

The file data

```
#Function that handles AUDIO/VIDEO FILE TRANSMISSION
def Audio_handler(connectionSocket):
    audio_file = connectionSocket.recv(1024).decode()
    #Reads the audio file as bytes/binary
    with open (audio_file,'rb') as file:
        audio_data = file.read()
    connectionSocket.sendall(audio_data)
    connectionSocket.close()
```