T.C.

MANİSA CELAL BAYAR UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

COMPUTER NETWORK

(Project 2)

Group 15

ENİS B. BULUT 190315040

Ergül FERİK 190315041

Doğukan DÜĞÜN 210315006

Assoc. Prof. Birim BALCI

MANISA, 2023

Contents

QUESTION 1

"Create an ICMP ping scanner and test the network to find the IP address of the given machine."

Used Technologies

1. Python Programming Language: This IP Address Scanning Tool was developed using the Python programming language.
2. Pyhcarm Community Edition 2022.3.3: IDE for Python
3. Threading Module: The threading module is used to perform parallel operations using multiple threads.
4. ping3 Library: The ping3 library is used to send ICMP ping requests and receive responses.

IP Address Scan Method:

- This IP Address Scan Tool uses ICMP ping requests to scan for IP addresses in a specific IP address range. It includes the following steps:
- The ping_range function takes the starting and ending IP addresses and returns a list of all IP addresses in that range.
- Starting and ending IP addresses are obtained from the user.
- The IP address range is determined using the ping_range function.
- The check_ip function checks if IP addresses are reachable by sending an ICMP ping request.
- Scanning is performed in parallel using multiple threads.
- Accessible IP addresses are added to the correctIPs list and saved in a file named "Correct_IP.txt".

Code Structure and Operation:

- The ping_range function takes the starting and ending IP addresses and creates a list of all the IP addresses between them.
- The check_ip function sends an ICMP ping request to the given IP address, and if it receives a response, it adds the correct IP addresses to the correctIPs list.
- Threads are used to scan IP addresses. A separate thread is created for each IP address and scanning is performed in parallel.
- Accessible IP addresses are written to a file named "Correct_IP.txt".
- "Scanning done." message is printed on the screen.

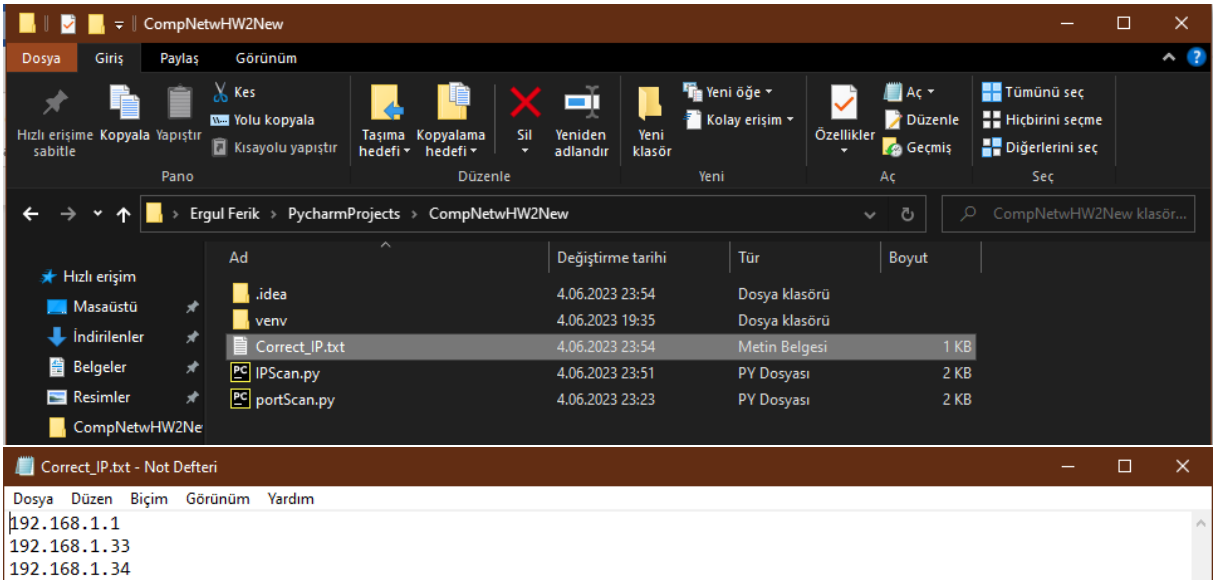
Source Code:

```python
import threading
from ping3 import ping


def ping_range(start_ip, end_ip):
    start = list(map(int, start_ip.split('.')))
    end = list(map(int, end_ip.split('.')))

    ip_range = []
    ip_range.append(start_ip)

    while start != end:
        start[3] += 1
        for i in range(3, 0, -1):
            if start[i] == 256:
                start[i] = 0
                start[i - 1] += 1
        ip_range.append('.'.join(map(str, start)))

    return ip_range


start_ip = input("Start IP:")
end_ip = input("End IP:")

ip_range = ping_range(start_ip, end_ip)
correctIPs = []


def check_ip(ip):
    try:
        if ping(ip):
            correctIPs.append(ip)
    except:
        pass


threads = []
for ip in ip_range:
    t = threading.Thread(target=check_ip, args=(ip,))
    threads.append(t)
    t.start()

for thread in threads:
    thread.join()

with open("Correct_IP.txt", "w") as f:
    for ip in correctIPs:
        f.write(ip + "\n")

print("Scanning done.")
```

Inputs and Output

```
with open("Correct_IP.txt", "w"... › for ip in correctIPs
Run:  IPScan
C:\Users\erg_m\PycharmProjects\CompNetwHW2New\venv\Scripts\python.exe C:\Users\erg_m\PycharmProjects\CompNetwHW2New\IPSc
Start IP:192.168.1.1
End IP:192.168.1.255
Scanning done.

Process finished with exit code 0
```

“Correct_IP.txt”



QUESTION 2

"Develop and/or design a tool / solution so that it performs port scan to find open ports on the target machine and discuss their advantages and disadvantages over other solutions."

Used Technologies

1. Python Programming Language: This Port Scanning Tool was developed using the Python programming language.
2. Pyhcarm Community Edition 2022.3.3: IDE for Python
3. socket Module: The socket module is used to perform port scanning using TCP/IP sockets.
4. Threading Module: The threading module is used to perform parallel operations using multiple threads.
5. queue Module: The queue module is used to share data between threads.

Port Scan Method:

- The destination IP address is passed as a parameter to the scan function.
- Using the socket module, the IP address is resolved and assigned to the t_IP variable.
- A certain timeout value (socket.setdefaulttimeout(0.25)) is set in the socket module for scanning.
- The portscan function tries to establish a TCP connection to the specified IP address and port combination.
- If the connection is successful, a message is printed stating that port is accessible, and this information is added to the ip_and_port list.
- Scanning is performed in parallel using multiple threads.
- Threads used in scanning are managed by the queue module.

Code Structure and Operation:

- A list named correctIPs is created and added to this list by reading the correct IP addresses from the "Correct_IP.txt" file.
- The scan function is run for each correct IP address and the accessible ports are added to the ip_and_port list.
- Accessible ports are associated with the correct IP addresses.
- Finally, the scan results are printed to the screen.

Source Code:

```python
import socket
import threading

from queue import Queue

ip_and_port = []


def scan(targetIP):
    global ip_and_port

    t_IP = socket.gethostbyname(targetIP)
    socket.setdefaulttimeout(0.25)
    print_lock = threading.Lock()

    def portscan(port):

        global ip_and_port

        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        try:
            con = s.connect((t_IP, port))
            with print_lock:
                ip_and_port.append(str(t_IP) + ":" + str(port))
            con.close()
        except:
            pass

    def threader():
        while True:
            workers = q.get()
            portscan(workers)
            q.task_done()

    q = Queue()

    for x in range(100):
        t = threading.Thread(target=threader)
        t.daemon = True
        t.start()

    for worker in range(1, 500):
        q.put(worker)

    q.join()


correctIPs = []

with open("Correct_IP.txt") as f:
    for i in f.readlines():
        correctIPs.append([i.strip()])

for i in correctIPs:
    scan(i[0])

for ip in correctIPs:
    for i in ip_and_port:
        arr = str(i).split(":")
        if ip[0] == arr[0]:
            ip.append(arr[1])

for i in correctIPs:
    if len(i) > 1:
        print(i[0], " is reachable and ports are : ")
        for j in range(1, len(i)):
            print(i[j])
    else:
        print(i[0], " is reachable but have not port")
```

Output

```
scan()
Run:  portScan
C:\Users\erg_m\PycharmProjects\CompNetwHW2New\venv\Scripts\python.exe C:\Users\erg_m\PycharmProjects\CompNetwHW2New\port
192.168.1.1  is reachable and ports are :
53
80
443
192.168.1.33  is reachable and ports are :
139
135
445
192.168.1.34  is reachable but have not port
192.168.1.37  is reachable but have not port

Process finished with exit code 0
```

Advantages and Disadvantages:

Advantages:

- Port scanning using parallel processes significantly reduces scanning time.
- Since scanning is performed with multiple threads, multiple ports can be checked at the same time and faster results can be obtained.
- Python programming language is easy to use and flexible. Therefore, the development and customization processes are easier.

Disadvantages:

- Sufficient resources and timing capabilities are needed to scan all ports in a given IP range. This may cause some restrictions when browsing large IP address ranges.
- Some target systems can detect and take action against network scanning activities such as port scanning. This may hinder the scanning process or affect the results.