# COMPUTER NETWORKS

# INDUSTRIAL PROBLEM – 1

# TITLE : TCP PORT SCANNING


## TEAM :

SANMAT SANJAYAKUMAR PAYAGOUDAR

SRN : PES1UG20CS385

SATHWICK P

SRN : PES1UG20CS388


## DESCRIPTION :

In this project we are finding on which particular port number a particular website is running, and the name of the service on that particular port number. As we can see in result that without using threading the scanning takes a lot of time and using threading it takes very less time compared to without threading. And we are using a local host to verify, with a particular port number assigned to it.

## TCP PORT SCANNING WITHOUT USING THREADING :

### CODE:

```
import socket
from datetime import datetime

## Enter Host to scan
host = input("Enter a remote host to scan: ")
ip = socket.gethostbyname(host) # Translate a host name to IPv4 address format

#This is just a nice touch that prints out information on which host we are
about to scan
print("-" * 80)
print("              Please wait, scanning the host --------> ", ip)
print("-" * 80)

## Check what time the scan started
t1 = datetime.now()

## Using the range function to specify ports (here it will scans all ports
between 1 and 1024)
## We also put in some error handling for catching errors
```

```
try:
    for port in range(1,80):
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #it use for
Creates a stream socket
        result = sock.connect_ex((ip, port))
        if result == 0:
            ## if a socket is listening it will print out the port number
            print("\n Port %d Is Open!!!!!!!!!!!!!" %(port))
            sock.close()
        else:
            print("\n Port %d Is Close :( " %(port))

except:
    pass

## Checking the time again
t2 = datetime.now()
## Calculates the difference of time, to see how long it took to run the
script
total = t2 - t1
## Printing the information to screen
print('Scanning Completed in: ', total)
```

**OUTPUT :**

Time taken for scanning 80 ports : 27min, 43 sec

```
PS C:\Users\Sanam\Desktop\Study\CN\Project\2\TCP-Port-Scanner-master>  & 'C:\Python310\python.exe' 'c:\Users\
Sanam\.vscode\extensions\ms-python.python-2022.4.1\pythonFiles\lib\python\debugpy\launcher' '58303' '--' 'c:\
Users\Sanam\Desktop\Study\CN\Project\2\TCP-Port-Scanner-master\tcp_port_scanner_1.py'
Enter a remote host to scan: youtube.com
---------------------------------------------------------------------------
            Please wait, scanning the host --------->  142.250.205.238
---------------------------------------------------------------------------

 Port 80 Is Open!!!!!!!!!!!!
Scanning Completed in:  0:27:43.377989
PS C:\Users\Sanam\Desktop\Study\CN\Project\2\TCP-Port-Scanner-master>
```

## TCP PORT SCANNING USING THREADING :

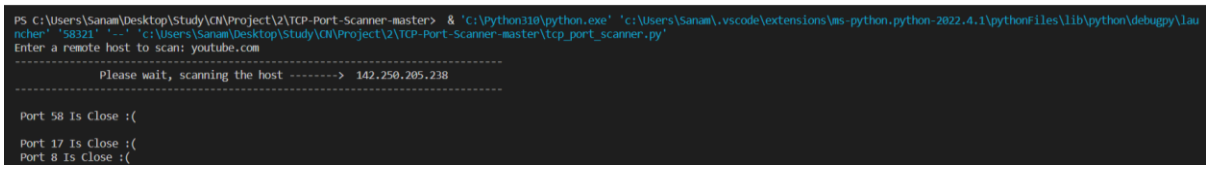
**CODE :**

```
import socket
from datetime import datetime
import threading
from queue import Queue


# a print_lock is what is used to prevent "double" modification of shared
variables.
print_lock = threading.Lock()

## Enter Host to scan
host = input("Enter a remote host to scan: ")
ip = socket.gethostbyname(host) # Translate a host name to IPv4 address format

#This is just a nice touch that prints out information on which host we are
about to scan
print("-" * 80)
print("               Please wait, scanning the host --------> ", ip)
print("-" * 80)

## Check what time the scan started
t1 = datetime.now()

## Using the range function to specify ports (here it will scans all ports
between 1 and 1024)
## We also put in some error handling for catching errors
def scan(port):
    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM) #it use for
Creates a stream socket
        result = sock.connect_ex((ip, port))
        if result == 0:
            ## if a socket is listening it will print out the port number
            serviceName=socket.getservbyport(port)
            print("\n Port %d Is Open!!!!!!!!!!!!!" %(port))
            print("Name of the service running at port number %d :
%s"%(port,serviceName))
            sock.close()
        else:
           print("\n Port %d Is Close :( " %(port))

    except:
```

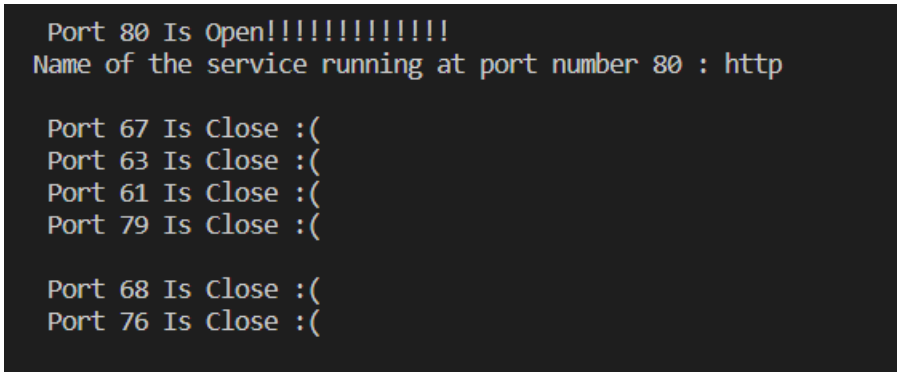
```
        pass

# The threader thread pulls an worker from the queue and processes it
def threader():
    while True:
        # gets an worker from the queue
        worker = q.get()

        # Run the example job with the avail worker in queue (thread)
        scan(worker)

        # completed with the job
        q.task_done()



# Create the queue and threader
q = Queue()

# how many threads are we going to allow for
for x in range(60):
     t = threading.Thread(target=threader)

     # classifying as a daemon, so they will die when the main dies
     t.daemon = True

     # begins, must come after daemon definition
     t.start()

for worker in range(1,21):
    q.put(worker)

# wait until the thread terminates.
q.join()

## Checking the time again
t2 = datetime.now()
## Calculates the difference of time, to see how long it took to run the
script
total = t2 - t1
## Printing the information to screen
print('Scanning Completed in: ', total)
```

**OUTPUT :**

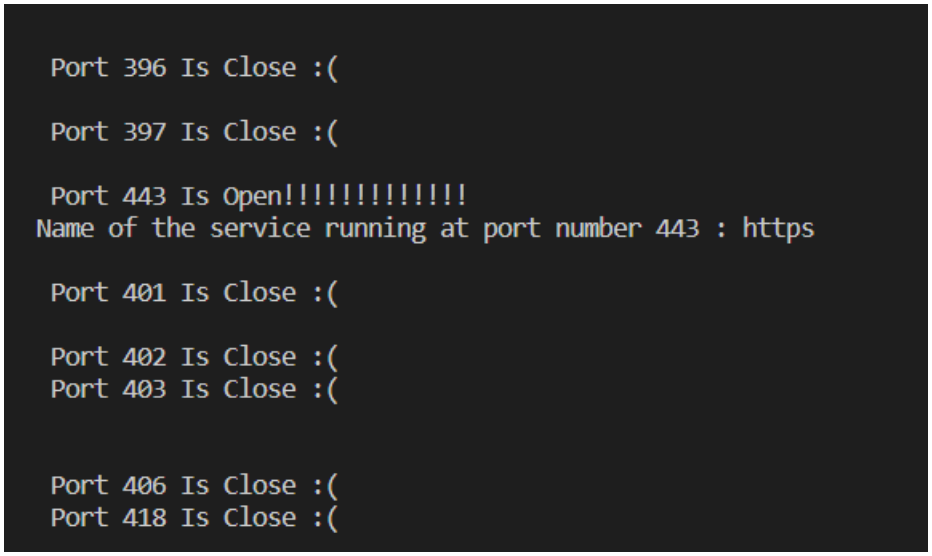
Input website : youtube.com



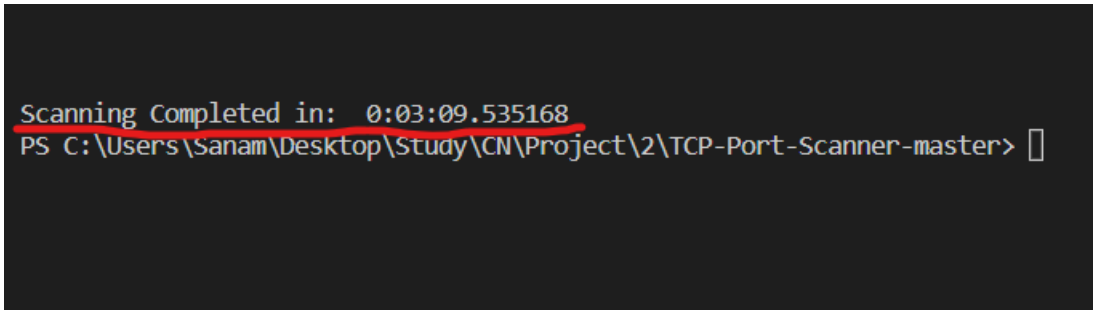

Port number 80 is open.




Port number 443 is open.




Time taken for scanning 500 ports : 3 min, 9 sec



## VERIFING WITH A LOCAL HOST :

**CODE :**

```
from http.server import BaseHTTPRequestHandler, HTTPServer
import time

hostName = "localhost"
serverPort = 20

class MyServer(BaseHTTPRequestHandler):
    def do_GET(self):
        self.send_response(30)
        self.send_header("Content-type","text/html")
        self.end_headers()

if __name__ == "__main__":
    webServer = HTTPServer((hostName, serverPort), MyServer)
    print("Server started http://%s:%s" % (hostName, serverPort))
    try :
        webServer.serve_forever()
    except KeyboardInterrupt:
        pass
        webServer.server_close()
        print("Server stopped.")
```

**OUTPUT :**

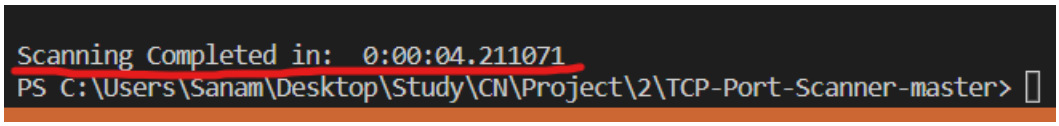
Local host running.

```
PS C:\Users\Sanam\Desktop\Study\CN\Project\2\TCP-Port-Scanner-master>  & 'C:\Python310\py
thon.exe' 'c:\Users\Sanam\.vscode\extensions\ms-python.python-2022.4.1\pythonFiles\lib\py
thon\debugpy\launcher' '59236' '--' 'c:\Users\Sanam\Desktop\Study\CN\Project\2\TCP-Port-S
canner-master\localhost.py'
Server started http://localhost:20
```

Port number 20 is open as we have assigned in code.

```
PS C:\Users\Sanam\Desktop\Study\CN\Project\2\TCP-Port-Scanner-master>  & 'C:\Python31
0\python.exe' 'c:\Users\Sanam\.vscode\extensions\ms-python.python-2022.4.1\pythonFile
s\lib\python\debugpy\launcher' '59303' '--' 'c:\Users\Sanam\Desktop\Study\CN\Project\
2\TCP-Port-Scanner-master\tcp_port_scanner.py'
Enter a remote host to scan: localhost
-------------------------------------------------------------------------------
             Please wait, scanning the host --------->  127.0.0.1
-------------------------------------------------------------------------------

 Port 20 Is Open!!!!!!!!!!!!!
Name of the service running at port number 20 : ftp-data

 Port 35 Is Close :(
```

Time taken for scanning 80 ports : 4 sec




## RESULT :

Time taken for scanning 80 ports without using Threading : 27min, 43 sec

Time taken for scanning 80 ports using Threading : 4 sec

As we can see in result that without using threading the scanning takes a lot of time and using threading it takes very less time compared to without threading.