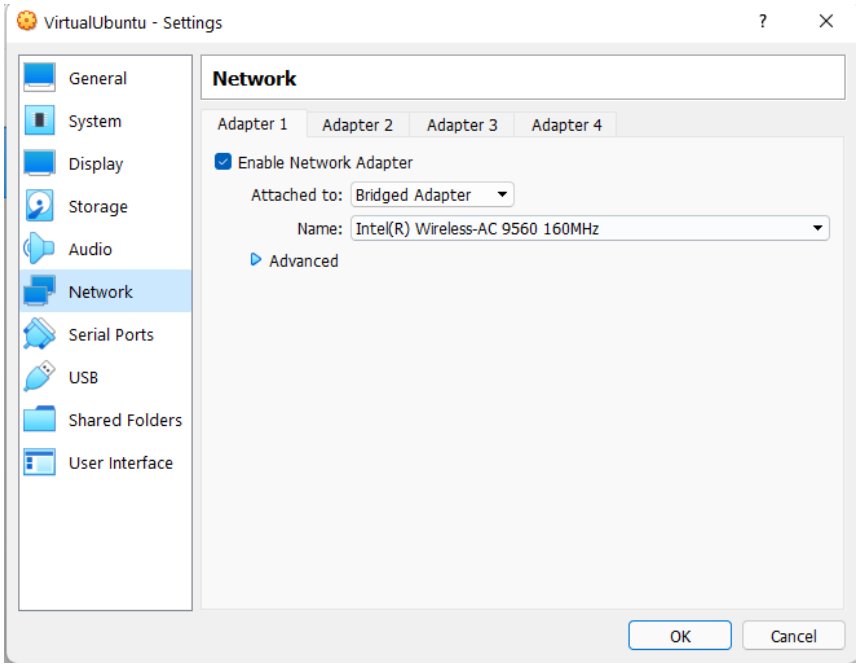
CS359 - Computer Network Lab

Lab 3

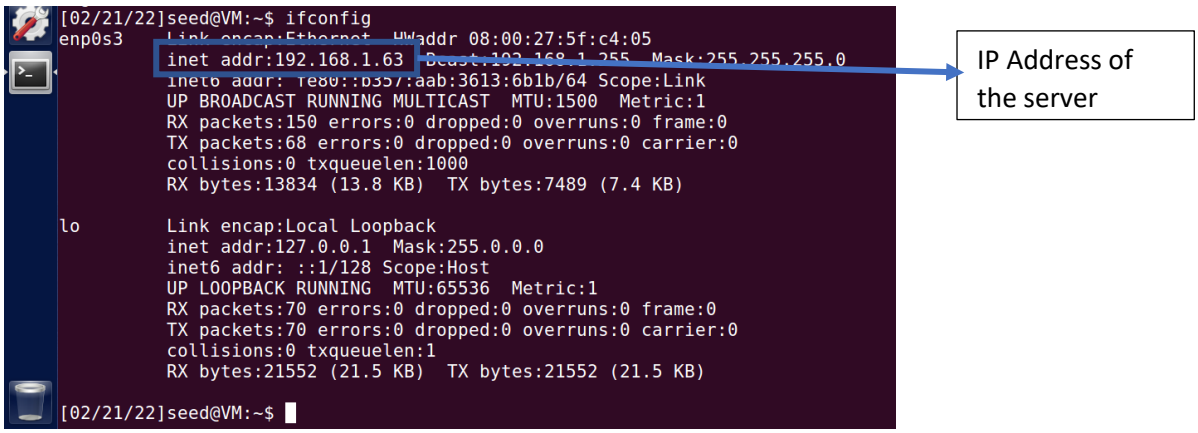
Socket Programming

Tarusi Mittal 1901CS65

For this lab, we need to have a server and a client side. So I have taken the virtual machine to be the serve and my computer to be the client. So for that we set the network settings of our virtual machine to bridged adapter.



Now we will run our virtual machine and follow the following steps:



QUESTION 1:

1. We create a python file with the name server.py and insert our code of making the server into it.

We have used the port 8888, for this purpose we can use any port greater than 1024.

```
import socket

# Function which handles request generated by clients
def handle_request(request):

    headers = request.split('\n')
    filename = headers[0].split()[1]
    if filename == '/':
        filename = 'index.html' #this is our home

    try:
        fin = open(filename[1:])
        content = fin.read()
        fin.close()
        response = 'HTTP/1.0 200 OK\n\n' + content
        #This sends an OK message
    except FileNotFoundError:
        response = 'HTTP/1.0 404 NOT FOUND\n\nFile Not Found, please enter
a valid Filename!!'
        #If file not found then send this

    return response
```

```
#This is teh IP address of teh virtual machine
SERVER_HOST = "192.168.1.63"
#This helkps ud to communicate between server and the client
SERVER_PORT = 8888

# These commands create a socket for us
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
server_socket.bind((SERVER_HOST, SERVER_PORT))
server_socket.listen(1)
print('Listening on port %s ...' % SERVER_PORT)

while True:
    client_connection, client_address = server_socket.accept()
    request = client_connection.recv(1024).decode()
    print("Client is connected with Ip:",client_address)
    response = handle_request(request) #Return the response
    client_connection.sendall(response.encode())
    print("Client has disconnected with Ip:",client_address)
    client_connection.close()

server_socket.close() #Closes the socket
```

45,0-1 Bot

2. Now we make our HelloWorld.html file on this server which we will later use on our client side to access.

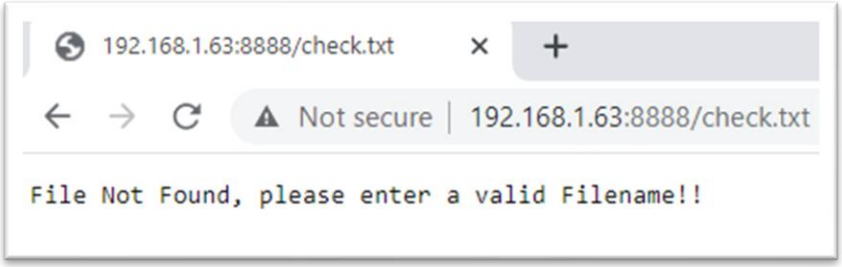
```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>
    <h1>Hello World !</h1>
    <h3> LAB-3 </h3>
    <h3> 1901CS65 </h3
  </body>
</html>
                                                    13,0-1        All
```

3. Now we will start and run our server.

```
[02/21/22]seed@VM:~$ python3 server.py
Listening on port 8888 ...
Client is connected with Ip: ('192.168.1.3', 62608)
Client has disconnected with Ip: ('192.168.1.3', 62608)
```

As we have entered the correct file name it opens in our browser.

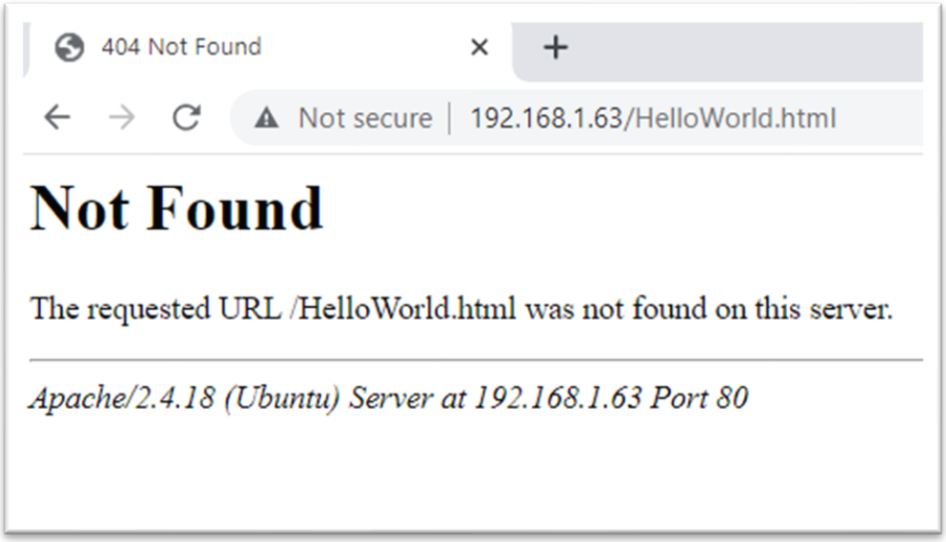
4. If we enter wrong name in our browser then:



We have got the error of file not found.

5. Now in the above examples we have specified our port 8888.

Let us see the case when our port is not defined and we simply write 192.168.1.63.



In this case we have got our error 404 of file not found. Because if we don't specify the port the default port taken by our system in port 80.

QUESTION 2:

In this question we want to establish multi connection with from the client side.

There we first make the following files.

1. make_config.py: This file will make the json configuration file for us by the name configuration.json. In this file we specify a default page which is our index.html, our blocked id's which can be changed anytime, and our maximum number of concurrent sessions allowed.

```
import json

# a Python object (dict):
x = {
  "max_concurrent_thread": 4,
  "blockIp": ['192.168.1.23'],
  "defaultFile": "/alpha.html"
}


print(x)
with open("configuration.json", "w") as outfile:
    json.dump(x, outfile)

~
~
~
~
~
~
~
~
~
"create_config.py" 14L, 226C
```

2. **alpha.html:** This is our default file which we had made where basically we land

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title></title>
  </head>
  <body>
    <h1>WELCOME!!</h1>
    <h3>This is the landing page :)</h3>
    <h3>Try opening multiple tabs by HelloWorld.html, beta.t
xt and gamma.txt</h3>
  </body>
</html>
```

3. HelloWorld.html: A file to open on the client side. Same as question 1.

4. beta.txt: A file to open on the client side

5. gamma.txt: A file to open on the client side

5. multipletabs.py: It sets up the server code which will allow us to open multiple tabs on our client side.

```
Terminal
import socket
import json
import threading
import time
import _thread

#IT will read from the json file that we have created
with open('configuration.json', 'r') as openfile:
    configure = json.load(openfile)


#it keeps a count of simulatenous connections
concurrent_connection = 0

# We define some values as default which
#are taken from our json configuration file
configure_filename = configure["defaultFile"]
max_connections = configure["max_concurrent_thread"]
ip_blocked = configure["blockIp"]
```

```
Terminal
#It handles the requests that are generated by clients
def handle_request(request,ip):
    global concurrent_connection
    headers = request.split('\n')
    filename = headers[0].split()[1]

    #It redirects to the default directory
    if filename == '/':
        filename = configure_filename

    try:
        fin = open(filename[1:])
        content = fin.read()
        fin.close()
        # Generating an OK response for client
        response = 'HTTP/1.0 200 OK\n\n' + content
    except FileNotFoundError:
        #generating a 404 error with a cutomized msg
        response = 'HTTP/1.0 404 NOT FOUND\n\n Sorry!, File Not Found!!'
    # Blocking the response, if ip is blocked
    if ip_blocked.count(ip)>0:
        response = 'HTTP/1.0 404 NOT FOUND\n\n Sorry!, This Ip is blocked!!'

    if concurrent_connection > max_connections:
        response = 'HTTP/1.0 404 NOT FOUND\n\n Sorry!!,Limit exceeded, Cant complete your request  !!'
    return response

#This is teh IP address of the socket
SERVER_HOST = "192.168.1.63"
#This helkps ud to communicate between server and the client
SERVER_PORT = 8088
```

```python
# Create socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
server_socket.bind((SERVER_HOST, SERVER_PORT))
server_socket.listen(1)
print(' Listening on port %s ...' % SERVER_PORT)

# Function to handle multiple requests from different clients
def solve(client_connection,client_address):
    global concurrent_connection
    #stores the ip adress oand port number of the client
    ip = client_address[0]
    port = client_address[1]

    #as we have opeoned a new page it increases the connection number by 1
    concurrent_connection += 1
    print("The new connection was made from IP:",client_address)
    while True:
        request = client_connection.recv(1024).decode()
        response = handle_request(request,ip)
        client_connection.sendall(response.encode())
        print("The client has disconnected, Ip:", client_address)
        break
    # Making the thread sleep so that we can see the effect of max allowed users
    time.sleep(10)
    client_connection.close()
    # If connection is closed that it reduces simulatenous users
    concurrent_connection -= 1
```

```python
while True:
    try:
        # Wait for client connections
        # it creates a new thread whenever a new client requests a server
        client_connection, client_address = server_socket.accept()
        _thread.start_new_thread(solve,(client_connection,client_address,))
    except KeyboardInterrupt as e:
        print("Server is now closed")
        break
    except Exception as e:
        print("Error ",e)
        break

server_socket.close() #we close the socket
```

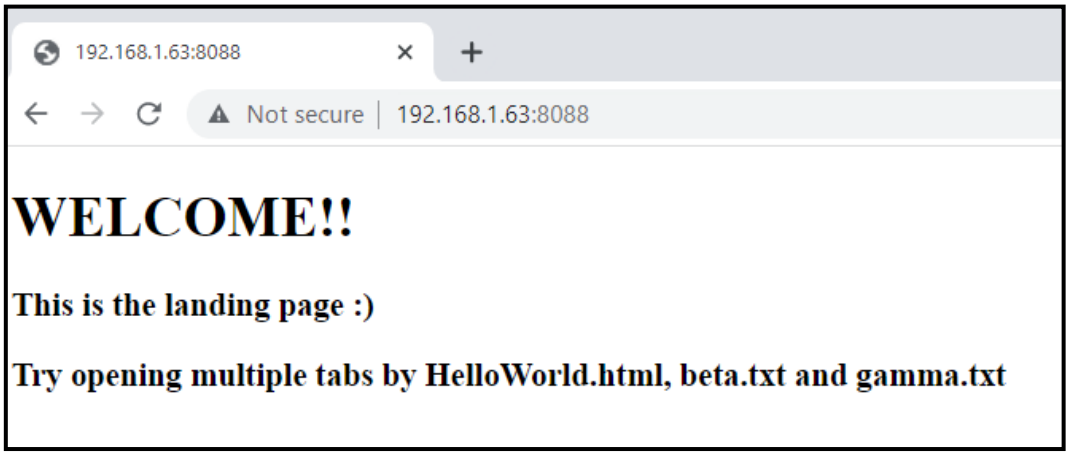
This completes our file making process and now we will run the server code and will try opening different tabs

```
[02/21/22]seed@VM:~$ vim create_config.py
[02/21/22]seed@VM:~$ python3 create_config.py
{'blockIp': ['192.168.1.23'], 'defaultFile': '/alpha.html',
'max_concurrent_thread': 4}
[02/21/22]seed@VM:~$ vim create_config.py
[02/21/22]seed@VM:~$ vim alpha.html
[02/21/22]seed@VM:~$ vim beta.txt
[02/21/22]seed@VM:~$ vim gamma.txt
[02/21/22]seed@VM:~$ vim multipletabs.py
[02/21/22]seed@VM:~$ python3 multipletabs.py
 Listening on port 8088 ...
```

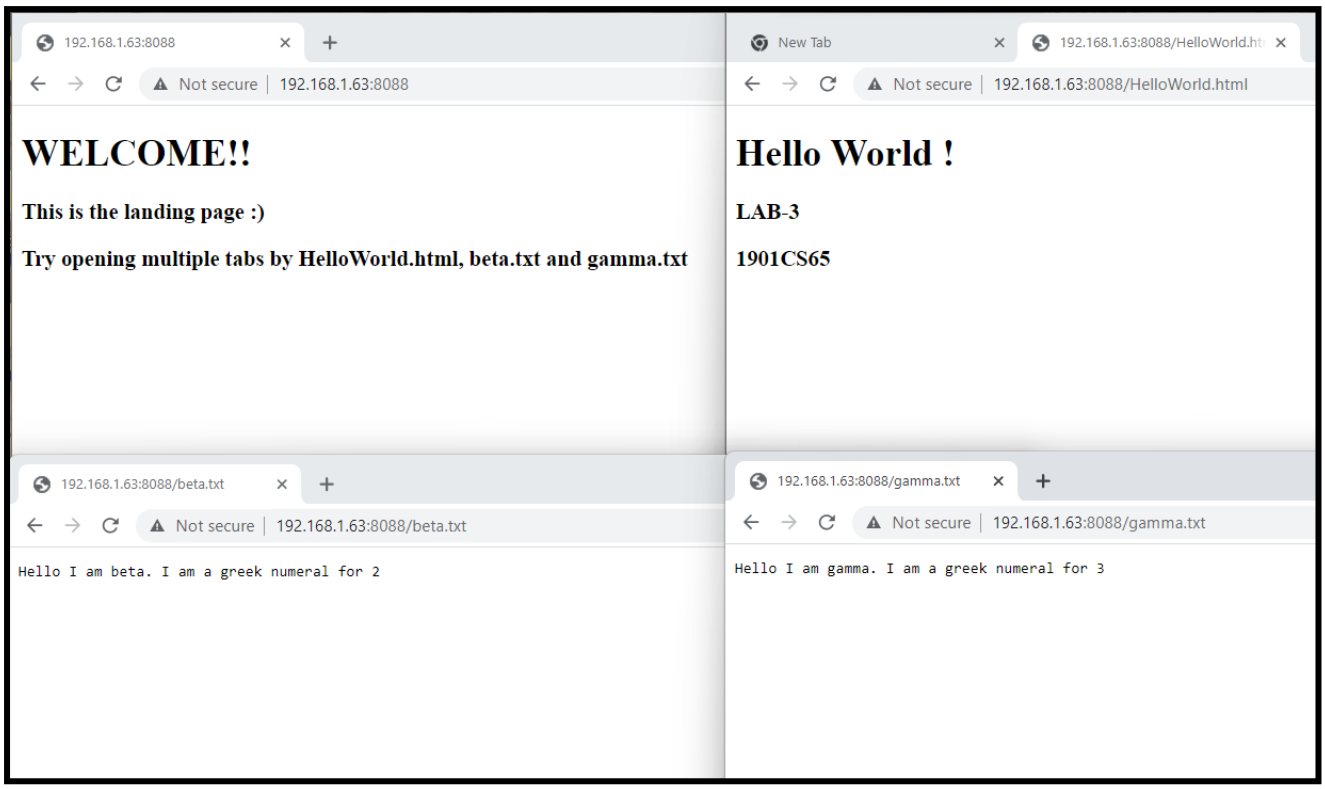
First we will open the link without specifying any file name to see if our default values are working which we have specified in our config file.

We have used the port 8088 for this.

As we can see without specifying any file name we landed at our default page.



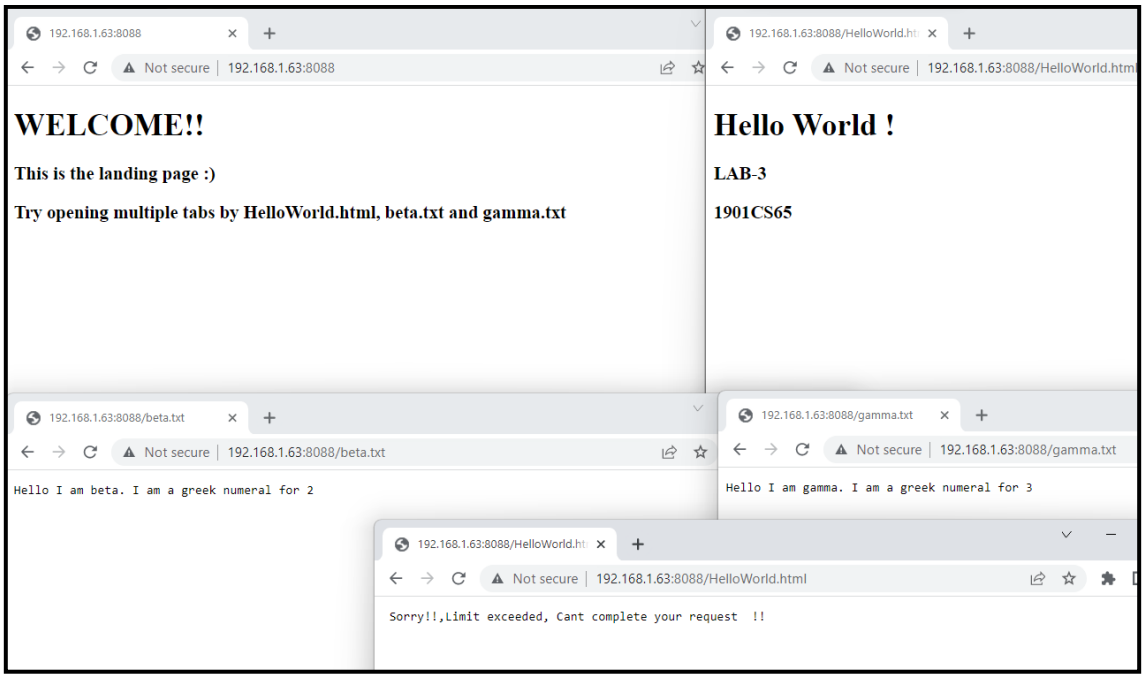
We try to open 4 tabs as is the maximum no as specified by our config.

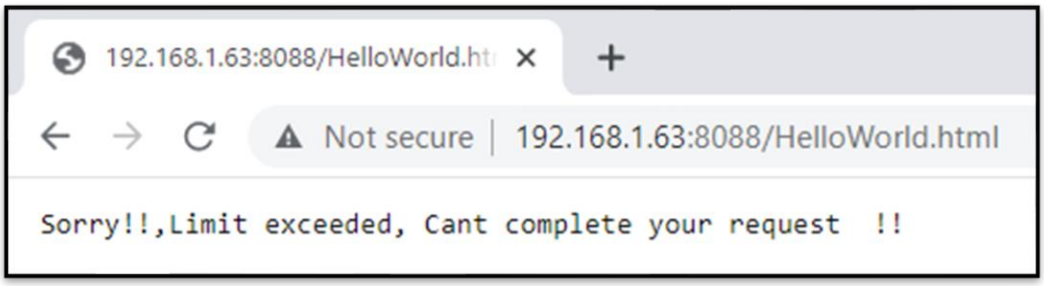


This is what our terminal looks like when we have made 4 requested

```
[02/21/22]seed@VM:~$ python3 multipletabs.py
 Listening on port 8088 ...
The new connection was made from IP: ('192.168.1.3', 62173)
The new connection was made from IP: ('192.168.1.3', 62172)
The client has disconnected, Ip: ('192.168.1.3', 62172)
The client has disconnected, Ip: ('192.168.1.3', 62173)
The new connection was made from IP: ('192.168.1.3', 62178)
The client has disconnected, Ip: ('192.168.1.3', 62178)
The new connection was made from IP: ('192.168.1.3', 62180)
The client has disconnected, Ip: ('192.168.1.3', 62180)
```

Now as our maximum no of tabs that can be opened simultaneously were 4. So we now open 5 tabs to check if our server is giving appropriate response:





Now we will see what if we add our windows Ip address to the blockIP Addresses list.

For that we first need to find the IP address of our device:

```
Windows PowerShell

PS C:\Users\Tarusi Mittal> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Media State . . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . : iitp.ac.in

Ethernet adapter VirtualBox Host-Only Network:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::92f:77ff:3dce:5949%22
   IPv4 Address. . . . . . . . . . . : 192.168.56.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Unknown adapter Local Area Connection:

   Media State . . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 2:

   Media State . . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::6db0:a1f2:9605:29ea%23
   IPv4 Address. . . . . . . . . . . : 192.168.1.3
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1

Ethernet adapter vEthernet (WSL):

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::5550:1c4a:8f51:97ae%68
   IPv4 Address. . . . . . . . . . . : 172.29.48.1
   Subnet Mask . . . . . . . . . . . : 255.255.240.0
   Default Gateway . . . . . . . . . :
```

Required IP Address for our purpose.

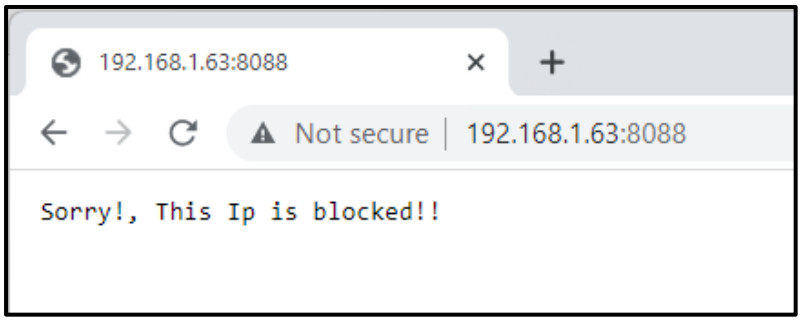
After finding the IP address, we change our config file to add the required IP address.

```
import json

# a Python object (dict):
x = {
  "max_concurrent_thread": 4,
  "blockIp": ['192.168.1.23','192.168.1.3'],
  "defaultFile": "/alpha.html"
}


print(x)
with open("configuration.json", "w") as outfile:
    json.dump(x, outfile)
```

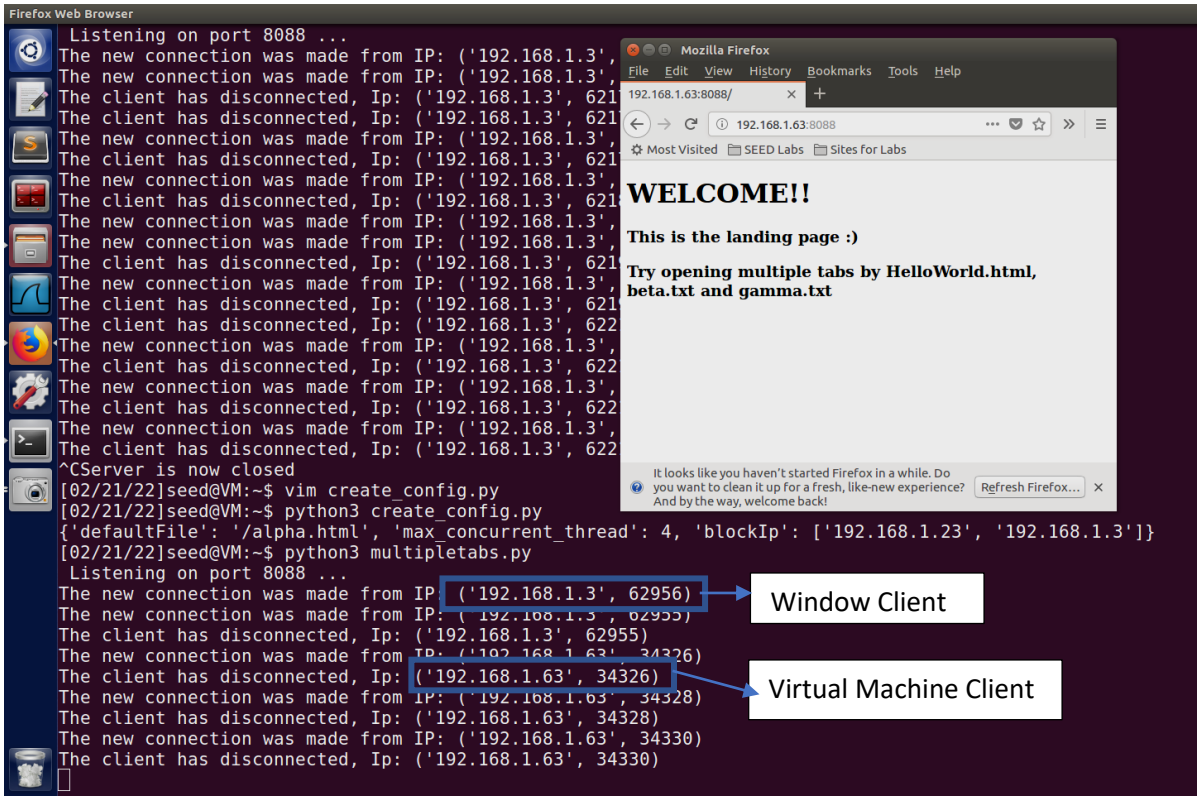
We deploy our config code and start the server.

```
[02/21/22]seed@VM:~$ vim create_config.py
[02/21/22]seed@VM:~$ python3 create_config.py
{'defaultFile': '/alpha.html', 'max_concurrent_thread': 4, 'blockIp': ['192.168.1.23', '192.168.1.3']}
[02/21/22]seed@VM:~$ python3 multipletabs.py
 Listening on port 8088 ...
The new connection was made from IP: ('192.168.1.3', 62956)
The new connection was made from IP: ('192.168.1.3', 62955)
The client has disconnected, Ip: ('192.168.1.3', 62955)
```

As we can see when we now try to access it from our windows client we get the error of IP Blocked



But if we try to run our page from the virtual machine itself we can see that we are able to run it, since its address was not in the block list.



Also on the terminal we can see the requests being generated from both the windows and the virtual machine.

END