Name: Sayan Acharya

Roll: 002010501009

Group: A1

Assignment: 8

Deadline: 7th - 11th November 2022

Submission date: 8th November

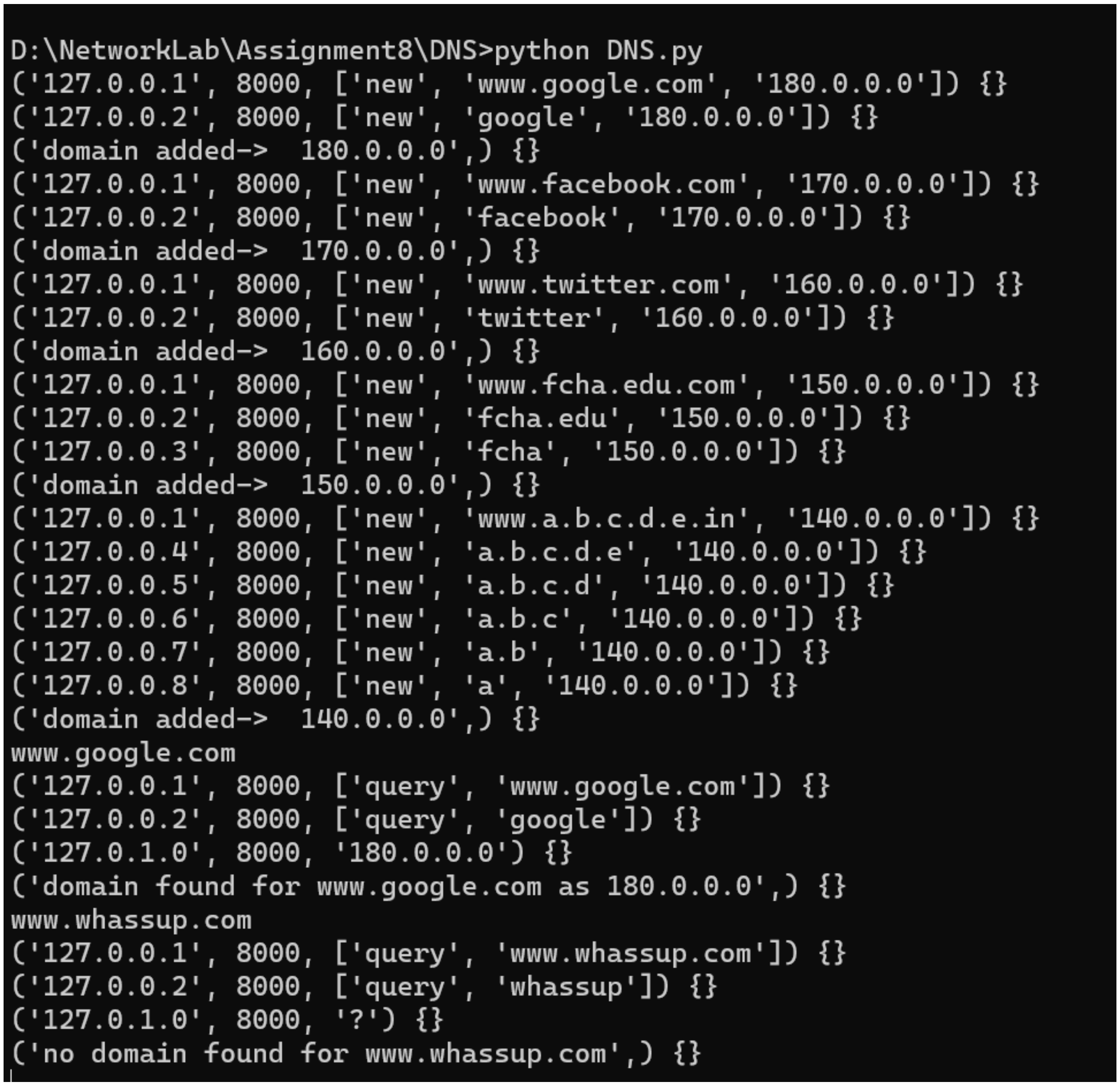
Assignment:
Implement any two protocols using TCP/UDP Socket as suitable.
1. FTP
2. DNS
3. Telnet

Code:
```
import socket,threading,json,time

MAX_BUF_SIZE=4096

print_lock=threading.Lock()

def _print(*args,**kwargs):
  print_lock.acquire()
  print(args,kwargs)
  print_lock.release()

def get_new_ip():
  index=1
  while True:
    yield f"127.0.0.{index}"
    index+=1

get_new_ip=iter(get_new_ip()).__next__

class dns_packet:
  def __init__(self,ip,port,pid,data):
    self.ip=ip
    self.port=port
    self.pid=pid
    self.data=data

  def encode(self,format='utf-8'):
    return f"{self.ip} {self.port} {self.pid} {self.data}".encode(format)

  @classmethod
  def decode(cls,string,format='utf-8'):
    listy=str(string,encoding=format).split()

    return dns_packet(listy[0],int(listy[1]),int(listy[2]),' '.join(listy[3::]))
```

```
class Server:
  def __init__(self,ip,port,parent_server,is_root_server=False):
    self.ip=ip
    self.port=port
    self.parent_server=parent_server

    # _print(ip)
    self.sock=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
    self.sock.bind((ip,port))

    self.is_root_server=is_root_server
    self.child_server=dict()

    self.domain_map=dict()

    self.threads=[]

    self.start_server()

  def get_addr(self):
    return (self.ip,self.port)

  def start_server(self):
    t=threading.Thread(target=self.listen,args=[],daemon=True)
    self.threads.append(t)
    t.start()

  # def add_device(self,addr):
  #   self.conneceted_devices.append(addr)

  def listen(self):
    while True:
      packet,addr=self.sock.recvfrom(MAX_BUF_SIZE)
      packet=dns_packet.decode(packet)
      t=threading.Thread(target=self.handle_packet,args=[packet,addr],daemon=True)
      self.threads.append(t)
      t.start()

  def peel_one_domain(self,string:str):#peeled part,rest
    listy=string.split('.')
    return listy[-1],'.'.join(listy[0:-1:]),len(listy)

  def remove_www(self,string:str):
    listy=string.split('.')
    index=0
    if listy[0]=="www":
      index+=1
    return '.'.join(listy[index::])

  def handle_packet(self,packet:dns_packet,prev_addr):
    data=packet.data.split()
    _print(self.ip,self.port,data)
    if data[0]=="query":
      if self.is_root_server:
        data[1]=self.remove_www(data[1])

      cur_dom,rest_dom,rest_len=self.peel_one_domain(data[1])
      if rest_len==1:#search ended
        ip='?'
        if cur_dom in self.domain_map:
          ip=self.domain_map[cur_dom]
        _print(packet.ip,packet.port,ip)
        self.send_packet(dns_packet(packet.ip,packet.port,packet.pid,f"reply
{ip}"),(packet.ip,packet.port))
      else:
        if cur_dom not in self.child_server:
          self.send_packet(dns_packet(packet.ip,packet.port,packet.pid,f"reply
?"),(packet.ip,packet.port))
        else:
          self.send_packet(dns_packet(packet.ip,packet.port,packet.pid,f"query
{rest_dom}"),self.child_server[cur_dom].get_addr())

    elif data[0]=="reply":
      if self.parent_server:
        self.send_packet(packet,self.parent_server)
      else:
        # _print(packet.ip,packet.port)
        self.send_packet(packet,(packet.ip,packet.port))

    elif data[0]=="new":
      if self.is_root_server:
        data[1]=self.remove_www(data[1])

      cur_dom,rest_dom,rest_len=self.peel_one_domain(data[1])
      if rest_len==1:#search ended
        self.domain_map[cur_dom]=data[2]
        _print(f"domain added-> {rest_dom} {data[2]}")
      else:
        if cur_dom not in self.child_server:
          self.child_server[cur_dom]=Server(get_new_ip(),8000,self.get_addr())

        self.send_packet(dns_packet(packet.ip,packet.port,packet.pid,f"new {rest_dom}
{data[2]}"),self.child_server[cur_dom].get_addr())



  def send_packet(self,packet:dns_packet,addr):
    self.sock.sendto(packet.encode(),addr)


class DNS:
  def __init__(self,ip_root,port_root):
    self.ip_root=ip_root
    self.port_root=port_root
    self.root_server=Server(ip_root,port_root,None,True)

  # def add_new_domain(self,dom_name,ip):
  #   self.root_server.send_packet(dns_packet('0',0,0,f"new {dom_name} {ip}"),(self.))

  # def query_domain(self,dom_name,query_machine_addr,pid):
  #   self.root_server.send_packet(dns_packet(*query_machine_addr,pid,f"query {dom_name}"))

class Client:
  def __init__(self,ip,port,dns_server:DNS):
    self.ip=ip
    self.port=port
    self.sock=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
    self.sock.bind((ip,port))
    self.dns_server=dns_server
    self.pid=0

    self.pid_table=dict()
    self.threads=[]

    self.start_client()

  def start_client(self):
    t=threading.Thread(target=self.listen,args=[],daemon=True)
    self.threads.append(t)
    t.start()

  def listen(self):
    while True:
      packet,addr=self.sock.recvfrom(MAX_BUF_SIZE)
      packet=dns_packet.decode(packet)

      data=packet.data.split()

      if data[0]=="reply":
        if data[1]=='?':
          _print(f"no domain found for {self.pid_table[packet.pid]}")
        else:
          _print(f"domain found for {self.pid_table[packet.pid]} as {data[1]}")

  def query(self,dom_name):
    self.pid_table[self.pid]=dom_name
    self.send_packet(dns_packet(self.ip,self.port,self.pid,f"query
{dom_name}"),self.dns_server.root_server.get_addr())
    self.pid+=1

  def add_new_domain(self,dom_name,ip):
    self.send_packet(dns_packet(self.ip,self.port,self.pid,f"new {dom_name}
{ip}"),self.dns_server.root_server.get_addr())

  def send_packet(self,packet,addr):
    self.sock.sendto(packet.encode(),addr)


if __name__=="__main__":
  dns=DNS(get_new_ip(),8000)
  client=Client('127.0.1.0',8000,dns)
  mappa=json.load(open('dom.json'))
  for i,j in mappa.items():
    client.add_new_domain(i,j)
    time.sleep(0.5)

  while True:
    inp=input()
    if inp.upper()=="EXIT":
      break
    client.query(inp)

  # mappa=dict()
  # while True:
  #   inp=input()
  #   if inp.upper()=="EXIT":
```

```
#     break

#   inp=inp.split()
#   mappa[inp[0]]=inp[1]
# json.dump(mappa,open('dom.json','w'))
```

Output:



```
D:\NetworkLab\Assignment8\DNS>python DNS.py
('127.0.0.1', 8000, ['new', 'www.google.com', '180.0.0.0']) {}
('127.0.0.2', 8000, ['new', 'google', '180.0.0.0']) {}
('domain added->  180.0.0.0',) {}
('127.0.0.1', 8000, ['new', 'www.facebook.com', '170.0.0.0']) {}
('127.0.0.2', 8000, ['new', 'facebook', '170.0.0.0']) {}
('domain added->  170.0.0.0',) {}
('127.0.0.1', 8000, ['new', 'www.twitter.com', '160.0.0.0']) {}
('127.0.0.2', 8000, ['new', 'twitter', '160.0.0.0']) {}
('domain added->  160.0.0.0',) {}
('127.0.0.1', 8000, ['new', 'www.fcha.edu.com', '150.0.0.0']) {}
('127.0.0.2', 8000, ['new', 'fcha.edu', '150.0.0.0']) {}
('127.0.0.3', 8000, ['new', 'fcha', '150.0.0.0']) {}
('domain added->  150.0.0.0',) {}
('127.0.0.1', 8000, ['new', 'www.a.b.c.d.e.in', '140.0.0.0']) {}
('127.0.0.4', 8000, ['new', 'a.b.c.d.e', '140.0.0.0']) {}
('127.0.0.5', 8000, ['new', 'a.b.c.d', '140.0.0.0']) {}
('127.0.0.6', 8000, ['new', 'a.b.c', '140.0.0.0']) {}
('127.0.0.7', 8000, ['new', 'a.b', '140.0.0.0']) {}
('127.0.0.8', 8000, ['new', 'a', '140.0.0.0']) {}
('domain added->  140.0.0.0',) {}
www.google.com
('127.0.0.1', 8000, ['query', 'www.google.com']) {}
('127.0.0.2', 8000, ['query', 'google']) {}
('127.0.1.0', 8000, '180.0.0.0') {}
('domain found for www.google.com as 180.0.0.0',) {}
www.whassup.com
('127.0.0.1', 8000, ['query', 'www.whassup.com']) {}
('127.0.0.2', 8000, ['query', 'whassup']) {}
('127.0.1.0', 8000, '?') {}
('no domain found for www.whassup.com',) {}
```

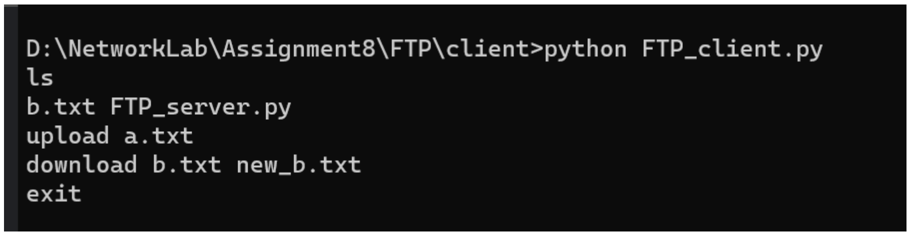
Code:(FTP)

```
import socket,threading,os
```

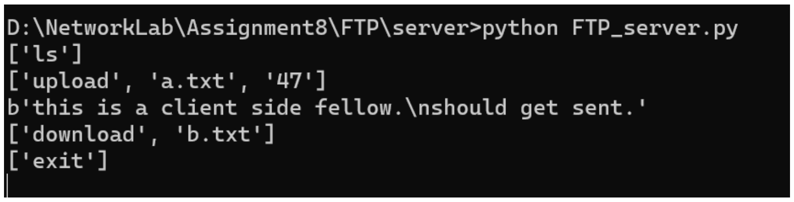
```
MAX_BUF_SIZE=4096

class ftp_server:
  def __init__(self,ip,port):
    self.ip=ip
    self.control_port=port
    self.data_port=port+1

    self.control_sock=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    self.data_sock=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    self.control_sock.bind((ip,self.control_port))
    self.data_sock.bind((ip,self.data_port))

    self.control_sock.listen()
    self.data_sock.listen()

    self.threads=[]
    self.start_server()

  def start_server(self):
    t=threading.Thread(target=self.establish_connection,args=[],daemon=True)
    self.threads.append(t)
    t.start()

  def establish_connection(self):
    while True:
      control_client,control_addr=self.control_sock.accept()
      data_client,data_addr=self.data_sock.accept()


t=threading.Thread(target=self.handle_connection,args=[control_client,control_addr,data_client,
data_addr],daemon=True)
      self.threads.append(t)
      t.start()

  def get_file(self,file_name):
    with open(file_name,'r') as f:
      lines=[i.strip('\r\n').strip('\n') for i in f.readlines()]
      return '\n'.join(lines).encode('utf-8')

  def make_file(self,file_name,_data):
    with open(file_name,'w') as f:
      _data=str(_data,encoding='utf-8')
      f.write(_data)
  def
handle_connection(self,control_client:socket.socket,control_addr,data_client:socket.socket,data
_addr):
    #take commands from control and then act accordingly
    while True:
      command=control_client.recv(MAX_BUF_SIZE)
      command=str(command,encoding='utf-8')
      command=command.split()
      print(command)
      if command[0]=="ls":
        if len(command)==1:
          command.append('.')
        all_files=' '.join(os.listdir(command[1]))
        data_client.send(all_files.encode('utf-8'))

      elif command[0]=="download":
        file_name=command[1]
        if not os.path.exists(file_name):
          control_client.send("-1".encode('utf-8'))
        else:
          control_client.send(str(os.path.getsize(command[1])).encode('utf-8'))
          _data=self.get_file(file_name)
          data_client.send(_data)

      elif command[0]=="upload":
        file_name=command[1]
        _size=int(command[2])
        _data=data_client.recv(_size)
        print(_data)
        self.make_file(file_name,_data)

      elif command[0]=="exit":
        data_client.close()
        control_client.close()
        break


if __name__=="__main__":
  server=ftp_server('127.0.0.1',8000)
  server.establish_connection()


import os,socket,threading,time
```

```
MAX_BUF_SIZE=4096

class ftp_client:
  def __init__(self,ip,port):
    self.ip=ip
    self.port=port
    self.control_sock=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    self.data_sock=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    self.control_sock.bind((ip,port))
    self.data_sock.bind((ip,port+1))

  def connect_with_server(self,server_ip,server_port):
    self.control_sock.connect((server_ip,server_port))
    self.data_sock.connect((server_ip,server_port+1))

  def make_file(self,file_name,_data):
    with open(file_name,'w') as f:
      _data=str(_data,encoding='utf-8')
      f.write(_data)

  def get_file(self,file_name):
    with open(file_name,'r') as f:
      lines=[i.strip('\r\n').strip('\n') for i in f.readlines()]
      return '\n'.join(lines).encode('utf-8')

  def command(self,command):
    command=command.split()
    if command[0]=='ls':
      self.control_sock.send(' '.join(command).encode('utf-8'))
      _data=self.data_sock.recv(MAX_BUF_SIZE)
      _data=str(_data,encoding='utf-8')
      print(_data)

    elif command[0]=='download':
      new_file_name=command[2]
      self.control_sock.send(' '.join(command[0:2:]).encode('utf-8'))
      _size=self.control_sock.recv(MAX_BUF_SIZE)
      _size=int(_size)
      _data=self.data_sock.recv(_size)
      self.make_file(new_file_name,_data)

    elif command[0]=='upload':
      _size=os.path.getsize(command[1])
      command.append(str(_size))
      self.control_sock.send(' '.join(command).encode('utf-8'))

      _data=self.get_file(command[1])
      self.data_sock.send(_data)

    elif command[0]=='exit':
      self.control_sock.send(' '.join(command).encode('utf-8'))
      time.sleep(0.5)
      self.control_sock.close()
      self.data_sock.close()
    else:
      print("wrong command given. no action taken")

if __name__=="__main__":
  client=ftp_client('127.0.0.2',8000)
  client.connect_with_server('127.0.0.1',8000)
  while True:
    inp=input()
    inp=inp.lower()
    client.command(inp)
    if inp=='exit':
      break

# download file_name
# upload file_name new_file_name
# ls directory(optional)
# exit
```

Output:



```
D:\NetworkLab\Assignment8\FTP\client>python FTP_client.py
ls
b.txt FTP_server.py
upload a.txt
download b.txt new_b.txt
exit
```



```
D:\NetworkLab\Assignment8\FTP\server>python FTP_server.py
['ls']
['upload', 'a.txt', '47']
b'this is a client side fellow.\nshould get sent.'
['download', 'b.txt']
['exit']
```

Discussion:

For dns, there are multiple servers starting from a root and they propagate in a trie like manner. Whenever a url is searched, the sections are taken one by one and searched. If there is a match, the suitable ip is returned

For ftp, two TCP connections are established and one of them is used as a control socket and the other one is data socket. the commands are given through the control socket and the actual data is passed through the data socket.

Comment:

While working on this, I learned a lot about the various protocols and their uses.