# علی فدائی منش – ۹۸۳۱۱۳۰ – پروژه شبکه

نکات مهم پیاده سازی:

۱. از قفل mutex برای محافظت لیست publish ها و client ها استفاده شده چون که نخ ها هنگام نوشتن و خواندن دچار race condition نشوند.
۲. از رویکرد oop برای مدیریت هرچه بهتر کد استفاده شده است

روند کارکرد برنامه به شکل زیر است:

```
PS C:\Users\afada\Desktop\server project> python server.py
Connected by ('127.0.0.1', 63571)
```

در ابتدا سرور آغاز می شود و منتظر connection جدید می ماند.

```
PS C:\Users\afada\Desktop\server project> python client.py
msg: publish nature mosquito
your message published suucessfully
PS C:\Users\afada\Desktop\server project>
```

وقتی یک client درخواست میدهد، مثلا برای publish سرور با پیام ack پاسخ می دهد و تایید می کند و پیام را برای سایر client هایی که topic مورد نظر را در خواست داده بودند ارسال می کند.

```
PS C:\Users\afada\Desktop\server project> python server.py
Connected by ('127.0.0.1', 63571)
address: ('127.0.0.1', 63571) sent: publish nature mosquito
clients:  [<__main__.Client object at 0x0000026A591D7430>]
publishes:  [<__main__.Publish object at 0x0000026A59AF4B20>]
```

```
PS C:\Users\afada\Desktop\server project> python client.py
msg: publish nature monkey
your message published suucessfully
PS C:\Users\afada\Desktop\server project>
```

سپس یک client اگر درخواست subscribe روی یک موضوع خاص را بدهد، برایش ارسال می شود

```
PS C:\Users\afada\Desktop\server project> python client.py
msg: subscribe nature
subscribing on nature
mosquitomonkey
```

کد server:

```python
from pydoc import cli
import socket
import threading
HOST = '127.0.0.1'  # Standard loopback interface address (localhost)
PORT = 1373  # Port to listen on (non-privileged ports are > 1023)

clients = []
publishes = []
lock = threading.Lock()


class Client():
    def __init__(self, conn, addr, clients, publishes, lock) -> None:
        self.conn = conn
        self.addr = addr
        self.subscribes = []
        self.thread = threading.Thread(
            target=handler, args=(self, clients, publishes, lock))

    def sendMessage(self, msg):
        self.conn.sendall(bytes(msg, 'utf-8'))

    def close(self):
        self.conn.close()
```

```
    def start(self):
        self.thread.start()


class Publish():
    def __init__(self, msg, topic, publisher) -> None:
        self.publisher = publisher
        self.msg = msg
        self.topic = topic


def handler(client, clients, publishes, lock):
    with client.conn:
        print('Connected by', client.addr)
        data = client.conn.recv(1024).decode('utf-8')
        print(f"address: {client.addr} sent: {data}")
        data = data.split(' ')
        print
        if data[0] == 'publish':  # publish
            topic = data[1]
            message = data[2]
            client.sendMessage('puback')
            publish = Publish(message, topic, client)
            lock.acquire()
            publishes.append(publish)
            for clienti in clients:
                if topic in clienti.subscribes:
                    print(clienti.addr)
                    clienti.sendMessage(message)
            lock.release()

        elif data[0] == 'subscribe':  # subscribe
            client.sendMessage('suback')
            topics = data[1:]
            lock.acquire()
            client.subscribes.extend(topics)
            for publish in publishes:
                if publish.topic in client.subscribes:
                    client.sendMessage(publish.msg)
            lock.release()

        elif data[0] == 'ping':
```

```
            client.sendMessage('pong')

    print('clients: ', clients)
    print('publishes: ', publishes)


with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    while True:
        # print('start listening!')
        conn, addr = s.accept()
        client = Client(conn, addr, clients, publishes, lock)
        clients.append(client)
        client.start()
        # print("accepted")
```

کد client:

```
# echo-client.py

from msilib.schema import Binary
from pydoc_data.topics import topics
import socket

HOST = "127.0.0.1"  # The server's hostname or IP address
PORT = 1373  # The port used by the server

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    msg = input('msg: ')
    s.sendall(bytes(msg, 'utf-8'))
    if msg.split(' ')[0] == 'publish':  # publish
        data = s.recv(1024).decode('utf-8')
        if data == 'puback':
            print("your message published suucessfully")
        else:
            print("your message publishing failed")

    elif msg.split(' ')[0] == 'subscribe':  # subscribe
        data = s.recv(1024).decode('utf-8')
```

```
        if data == 'suback':
            print(f'subscribing on {" ".join(msg.split(" ")[1:])}')
        else:
            print('subscribing failed')
        while True:
            data = s.recv(1024).decode('utf-8')
            if data:
                print(data)

    elif msg.split(' ')[0] == 'ping':
        data = s.recv(1024).decode('utf-8')
        print(data)
```