

موضوع: پیاده سازی python در massaging

از مزایای این تکنیک میتوان به موارد زیر اشاره کرد:

- There are no direct connections between programs.
- Communication between programs can be independent of time.
- Work can be carried out by small, self-contained programs.
- Communication can be driven by events.
- Applications can assign a priority to a message.
- Security.
- Data integrity.
- Recovery support.

که از خصوصیات مهم این تکنیک افزایش بهره وری ، مقیاس پذیری و افزایش اعتماد از صحت پیام ها می باشد.

یکی از بهترین و ساده ترین library های موجود برای زبان پایتون ØMQ^۱ می باشد که در پیاده سازی این روش ما از این کتاب خانه استفاده می کنیم.

۴ مدل پیاده سازی برای این روش وجود دارد که عبارتند از:

- Pair
- Client/server
- Publish/subscriber
- Push/pull

در پیاده سازی روش client/server یک نرم افزار چت سازی پیاده سازی کردیم که با پایتون ۳ نوشته شده است و نحوه استفاده از آن به فرم زیر است:

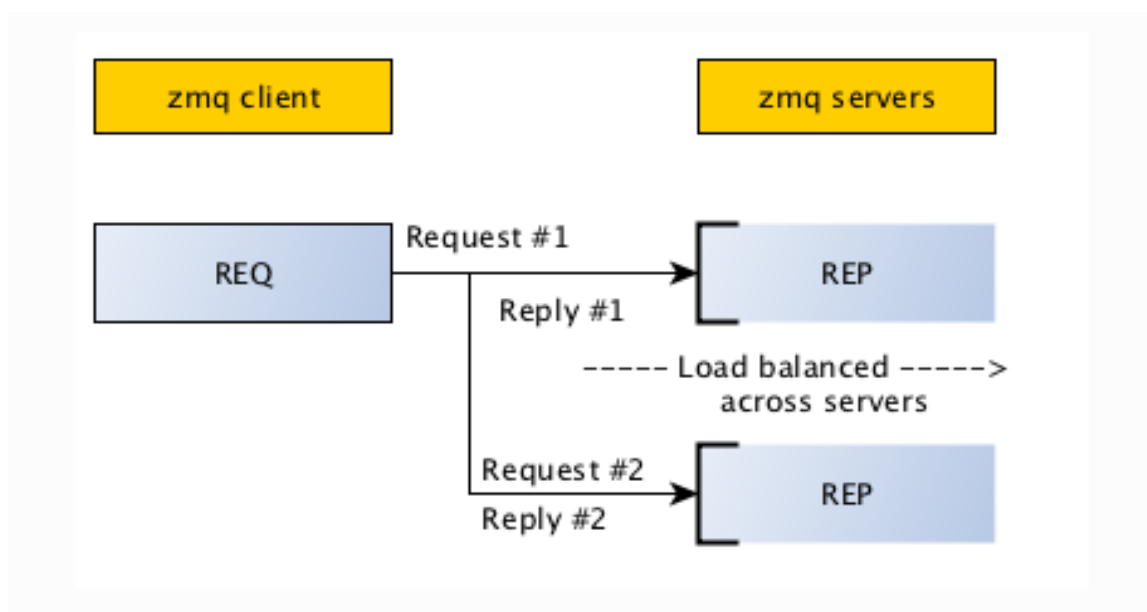
^۱ <https://learning-mq-with-pyzer.readthedocs.io/en/latest/index.html>

The screenshot shows a Linux desktop environment with two terminal windows open. The left terminal window shows the execution of a ZMQ client script, where the user enters messages 'salam' and 'merc', and the server responds with 'b'khubi?'. The right terminal window shows the execution of a ZMQ server script, which receives the messages and responds accordingly. The desktop background is a dark blue image with a person's face, and the taskbar at the bottom contains various application icons.

```
undercover@shadow: ~/Desktop/All_In_One/integration system/ZMQ
File Edit View Search Terminal Help
undercover@shadow:~/Desktop/All_In_One/integration system/ZMQ$ python3 client.py
Please enter your msg here: salam
sending salam
from server : b'khubi?'
Please enter your msg here: merc
sending merc
[ ]

undercover@shadow: ~/Desktop/All_In_One/integration system/ZMQ
File Edit View Search Terminal Help
undercover@shadow:~/Desktop/All_In_One/integration system/ZMQ$ python3 server.py
from client: b'salam'
enter your msg here: khubi?
#####
from client: b'merc'
enter your msg here: [ ]
```

نحوه عملکرد برنامه:



کد نوشته شده برای سرور:

```
"""
saeid ghasemshirazi
94403036
"""

import zmq

context = zmq.Context()
#zmq.rep for reply client requests
#ZMQ REQ sockets can connect to many servers.
#socket zmq.REQ will block on send unless it has successfully received a reply
back.

socket = context.socket(zmq.REP)
socket.bind("tcp://127.0.0.1:8877")

while True:
    #socket.recv for recieve msg form client
    msg = socket.recv()
    print("from client: ",msg)
    server_msg=input("enter your msg here: ")
    socket.send_string(server_msg)
    print("#####")
```

کد نوشته شده برای کلاینت:

```
import zmq

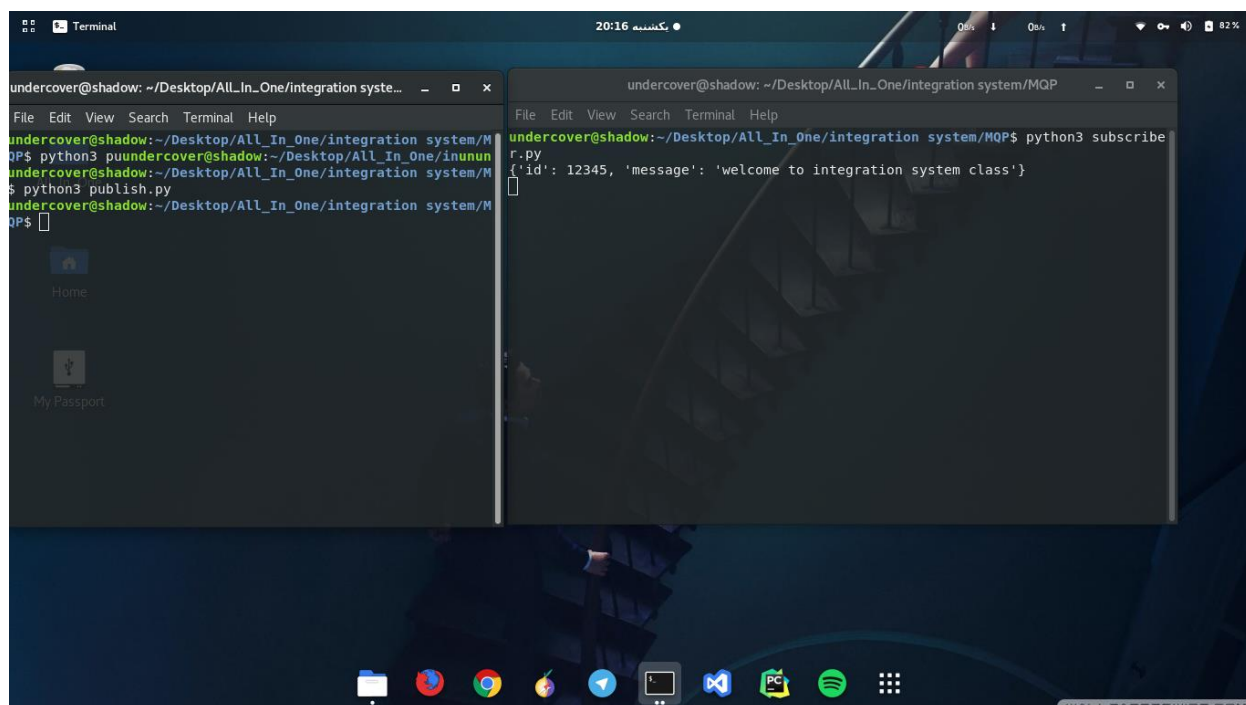
#Contexts are thread safe unlike sockets. An application can create and manage
multiple contexts.
context = zmq.Context()
#zmq.req for send request to server
#socket zmq.REP will block on recv unless it has received a request.
socket = context.socket(zmq.REQ)
socket.connect("tcp://127.0.0.1:8877")

while True:
    msg=input("Please enter your msg here: ")
    socket.send_string(msg)
    print( 'sending',msg)
    print("from server : ", socket.recv())
```

برای استفاده این روش روی ۲ سیستم مختلف به صورت زیر میتوان عمل کرد که در سرور ip را ۰.۰.۰.۰ قرار داده و در کد کلاینت ip را ip server قرار میدهیم.

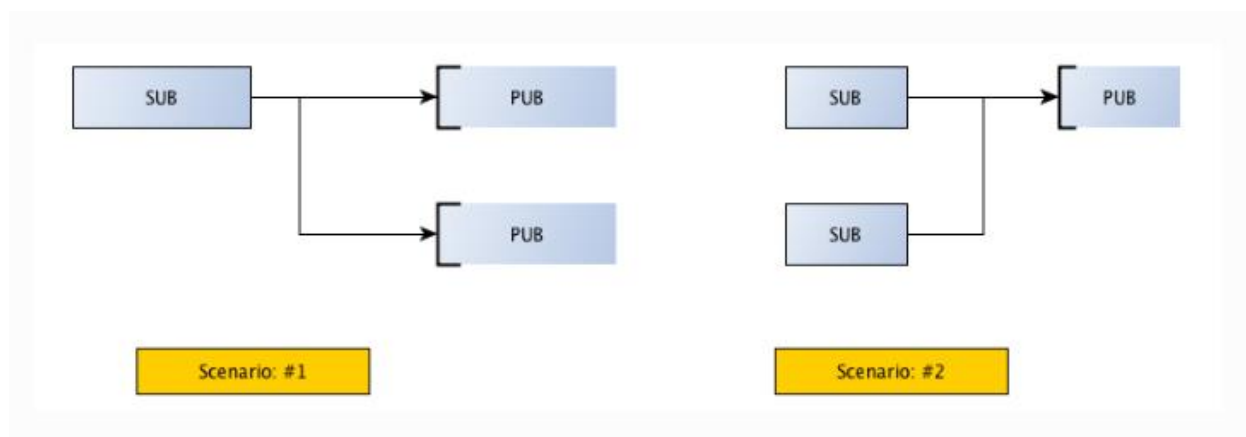
در روش بعدی که روش publish/subscriber هست از کتاب خانه Message Queue استفاده می کنیم.

نحوه عملکرد آن به صورت زیر می باشد:



```
undercover@shadow: ~/Desktop/All_In_One/integration syste...
File Edit View Search Terminal Help
undercover@shadow:~/Desktop/All_In_One/integration system/MQP$ python3 publish.py
undercover@shadow:~/Desktop/All_In_One/integration system/MQP$ python3 subscribe.py
{'id': 12345, 'message': 'welcome to integration system class'}
```

سناریو این روش به صورت زیر می باشد:



کد نوشته شده برای مایژولpublish

```
import message_queue
import pika

if __name__ == '__main__':

    adapter = message_queue.AMQPAdapter(host='127.0.0.1')
    # Configure queue
    adapter.configure_queue(queue='python.publish.test')
    # Instantiate publisher
    publisher = message_queue.Publisher(adapter)

    # Create a new message
    message = message_queue.Message(
        {
            'id': 12345,
            'message': 'welcome to integration system class'
        }
    )

    # Publish message
    publisher.publish(message)
```

کد نوشته شده برای subscriber

```
import json

import message_queue
import pika

def my_worker(channel, method, properties, body):
    print (json.loads(body))

if __name__ == '__main__':

    adapter = message_queue.AMQPAdapter(host='127.0.0.1')
    adapter.configure_queue(queue='python.publish.test')
    subscriber = message_queue.Subscriber(adapter)
    # Print message
    subscriber.consume(my_worker)
```