INSTITUTO FEDERAL
SÃO PAULO
Câmpus Presidente Epitácio

ØMQ

João Pedro de França Lourenço
Thiago Bruchmann Carnaiba

joao.franca@aluno.ifsp.edu.br
thiago.bruchmann@aluno.ifsp.edu.br

Sumário

- Introdução
- Empresas que utilizam ØMQ
- ØMQ Architecture
- ØMQ Messaging Patterns vs ØMQ Devices
- ØMQ Queue Device
- ØMQ Forwarder Device
- Aplicação utilizando ØMQ
- High-Water-Mark

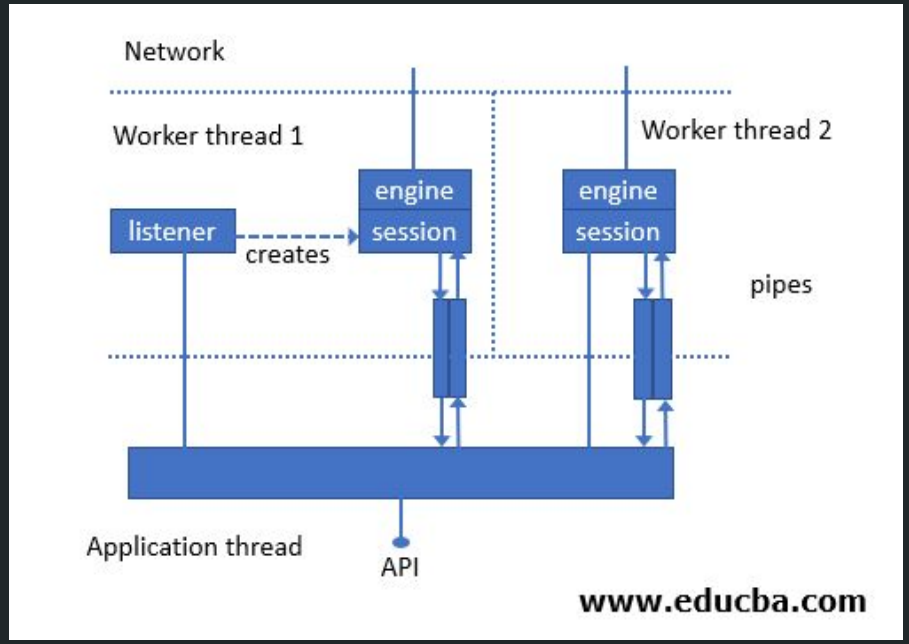
INSTITUTO FEDERAL
SÃO PAULO
Câmpus Presidente Epitácio

Introdução

- Biblioteca de mensagens assíncronas de alto desempenho
- Fornece uma fila de mensagens
- Pode ser executado sem um broker de mensagens dedicado

Empresas que utilizam

ØMQ Architecture



Estado Global

- A libzmq não possui variáveis globais.
- Em vez disso, o usuário da biblioteca é responsável por criar o estado global explicitamente.

Modelo de concorrência

- Passagem de mensagens
- Cada thread “vive” em sua própria thread
- Comunicação via chamadas enviadas aos objetos

Modelo de Threading

- Threads de aplicativos para utilizar a API
- Threads de I/O
- Cada Thread possui sua própria fila de leitura de “mensagens”

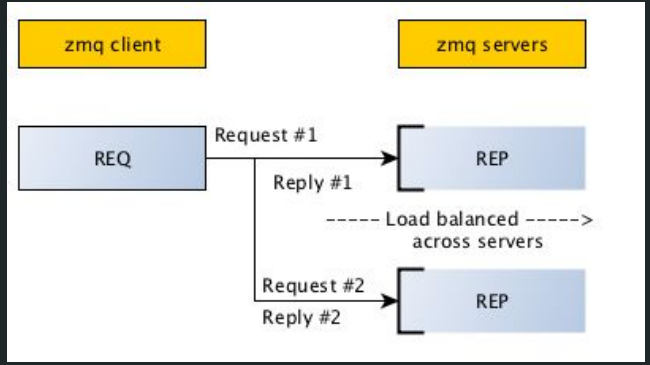
Encaminhamento de mensagens

- Dois tipos de canais:
  - Ativos
  - Passivos
- Geralmente o começo da lista é canais ativos e o resto é passivo

ØMQ Messaging Patterns vs ØMQ Devices

- O socket zmq.REQ vai bloquear o envio enquanto não receber uma resposta de sucesso
- O socket zmq.REP vai bloquear o recv enquanto não receber uma requisição.

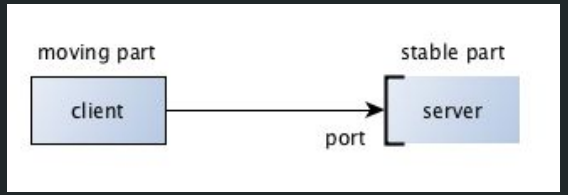
Cada solicitação/resposta é emparelhada e deve ser bem-sucedida.



ØMQ Messaging Patterns vs ØMQ Devices

Em teoria, a parte mais estável da rede (servidor) será conectada em uma porta específica e terá as partes mais dinâmicas (cliente) conectadas a ela.

Algumas vezes, ambas as extremidades podem ser dinâmicas e não é uma boa ideia fornecer portas conhecidas para qualquer uma das extremidades.

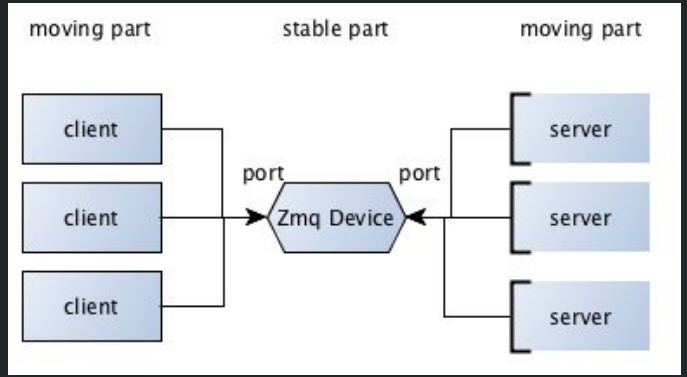


ØMQ Messaging Patterns vs ØMQ Devices

Nesses casos, você pode conectá-los usando o dispositivo de encaminhamento do zmq.

Esses dispositivos podem se conectar a 2 portas diferentes e encaminhar mensagens de uma extremidade à outra.

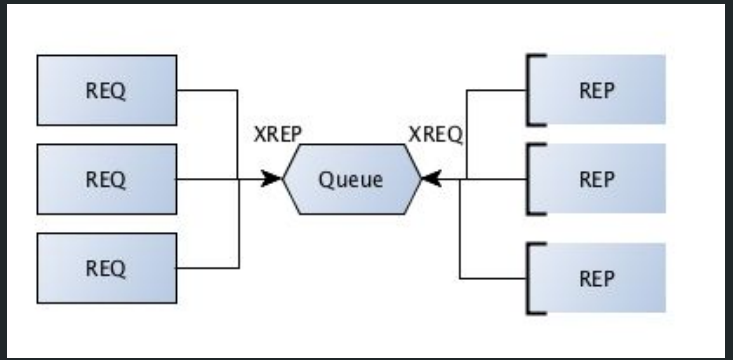
O dispositivo de encaminhamento pode se tornar o ponto estável em sua rede onde cada componente pode se conectar.



ØMQ Queue Device

Este é o dispositivo intermediário que fica entre clientes e servidores, encaminhando solicitações para servidores e retransmitindo respostas de volta ao cliente.

O dispositivo ZMQ recebe um tipo de dispositivo (ZMQ.QUEUE) e os dois soquetes são vinculados a portas conhecidas.



ØMQ Queue Device

```python
import zmq

def main():

    try:
        context = zmq.Context(1)

        frontend = context.socket(zmq.XREP)
        frontend.bind("tcp://*:5559")

        backend = context.socket(zmq.XREQ)
        backend.bind("tcp://*:5560")

        zmq.device(zmq.QUEUE, frontend, backend)
    except Exception as e:
        print(e)
        print("Desligando o dispositivo.")
    finally:
        pass
        frontend.close()
        backend.close()
        context.term()


if __name__ == "__main__":
    main()
```

```python
import zmq
import random

port = "5559"
context = zmq.Context()
print("Conectando ao server...")
socket = context.socket(zmq.REQ)
socket.connect("tcp://localhost:%s" % port)
client_id = random.randrange(1, 10005)

for request in range(1, 10):
    print("Enviando requisição ", request, "...")
    socket.send_string("Ola do %s" % client_id)

    message = socket.recv()
    print("Resposta recebida ", request, "[", message, "]")
```

INSTITUTO FEDERAL
SÃO PAULO
Câmpus Presidente Epitácio

ØMQ Queue Device

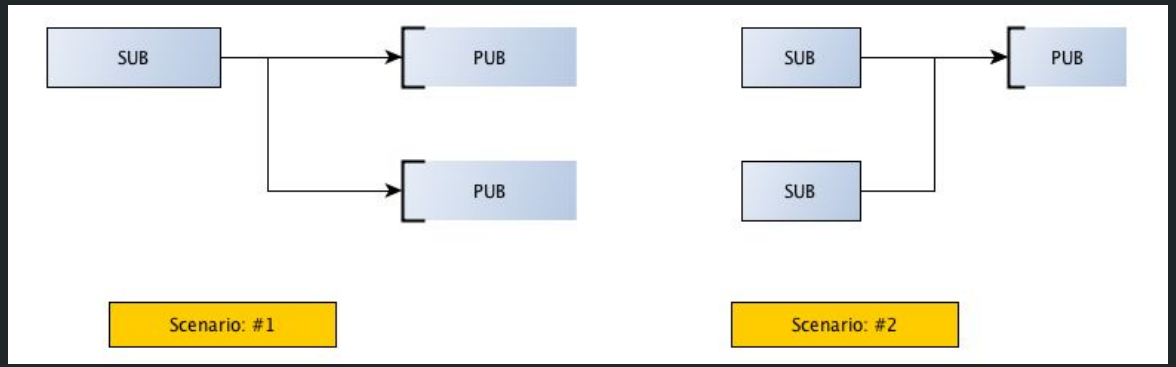
```python
import zmq
import time
import random

port = "5560"
context = zmq.Context()
socket = context.socket(zmq.REP)
socket.connect("tcp://localhost:%s" % port)
server_id = random.randrange(1, 10005)
while True:

    message = socket.recv()
    print("Requisição recebida: ", message)
    time.sleep(1)
    socket.send_string("Ola do servidor: %s" % server_id)
```

ØMQ Forwarder Device

Pub/Sub padrão

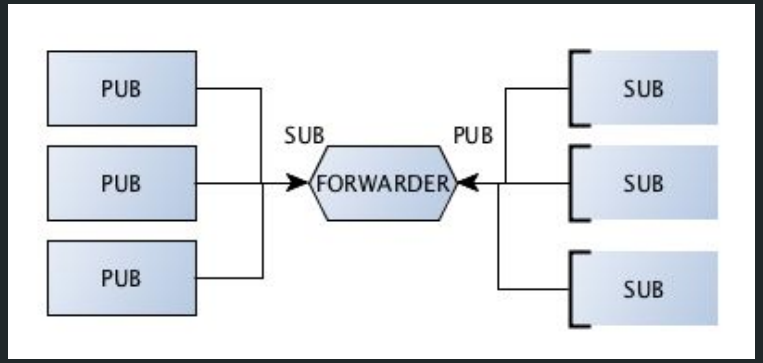


ØMQ Forwarder Device

Assim como no dispositivo de filas (queue), que é como o agente de solicitação-resposta, o dispositivo remetente (forwarder) é como o servidor proxy pub-sub.

Ele permite que editores e assinantes sejam partes móveis e se torne o hub estável para interconectá-los.

O forwarder coleta mensagens de um conjunto de editores e as encaminha para um conjunto de assinantes.



High-Water-Mark

É um limite no número máximo de mensagens pendentes que o ZeroMQ está enfileirando na memória para qualquer ponto único com o qual o soquete especificado esteja se comunicando.

Se esse limite for atingido, o soquete entra em um estado excepcional e, dependendo do tipo de soquete, o ZeroMQ tomará as medidas apropriadas, como bloquear ou descartar mensagens enviadas.

REQ socket: block.
PUB socket: drop.
ZMQ_HWM

Referências

Ashish Vidyarthi. **Learning ØMQ with pyzmq.** [S. l], 1 mar. 2012. Disponível em: https://learning-0mq-with-pyzmq.readthedocs.io/en/latest/index.html. Acesso em: 15 out. 2022.

The ZeroMQ Authors. **ØMQ Documentation.** [S. l], 2022. Disponível em: https://zeromq.org/get-started/. Acesso em: 15 out. 2022.

ØMQ/2.1.11 API Reference. Disponível em: http://api.zeromq.org/2-1:_start. Acesso em: 15 out. 2022.

ØMQ - The Guide \zguide\. Disponível em: https://zguide.zeromq.org/. Acesso em: 15 out. 2022.