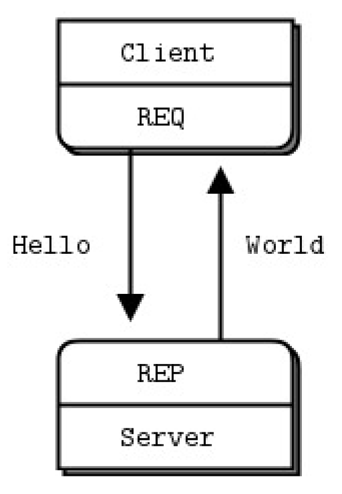
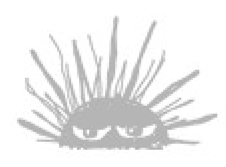
ZeroMQ - 三种模型的python实现

ZeroMQ是一个消息队列网络库，实现网络常用技术封装。在C/S中实现了三种模式，这段时间用python简单实现了一下，感觉python虽然灵活。但是数据处理不如C++自由灵活。

1.Request-Reply模式：

客户端在请求后，服务端必须回响应





server:

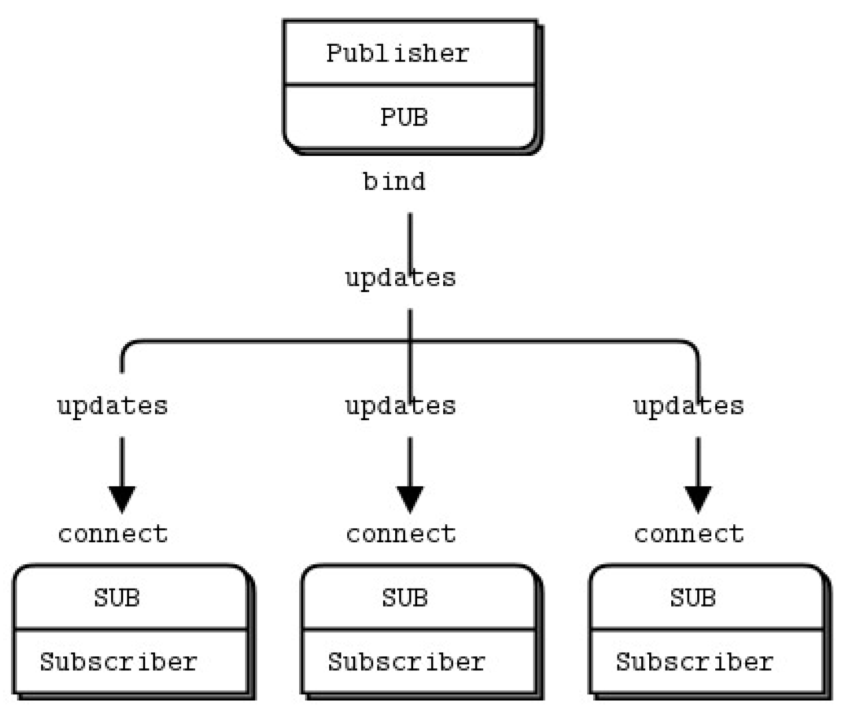
```python
#!/usr/bin/python
#-*-coding:utf-8-*-
import time
import zmq

context = zmq.Context()
socket = context.socket(zmq.REP)
socket.bind("tcp://*:5555")

while True:
    message = socket.recv()
    print message
    #time.sleep(1)
    socket.send("server response!")
```

client:

```python
#!/usr/bin/python
#-*-coding:utf-8-*-

import zmq
import sys

context = zmq.Context()
socket = context.socket(zmq.REQ)
socket.connect("tcp://localhost:5555")

while(True):
    data = raw_input("input your data:")
    if data == 'q':
        sys.exit()

    socket.send(data)

    response = socket.recv();
    print response
```

2.Publish-Subscribe模式：

广播所有client，没有队列缓存，断开连接数据将永远丢失。client可以进行数据过滤。



server:

```python
#!/usr/bin/python
#-*-coding:utf-8-*-

import zmq
context = zmq.Context()
socket = context.socket(zmq.PUB)
socket.bind("tcp://127.0.0.1:5000")
while True:
    msg = raw_input('input your data:')
    socket.send(msg)
```

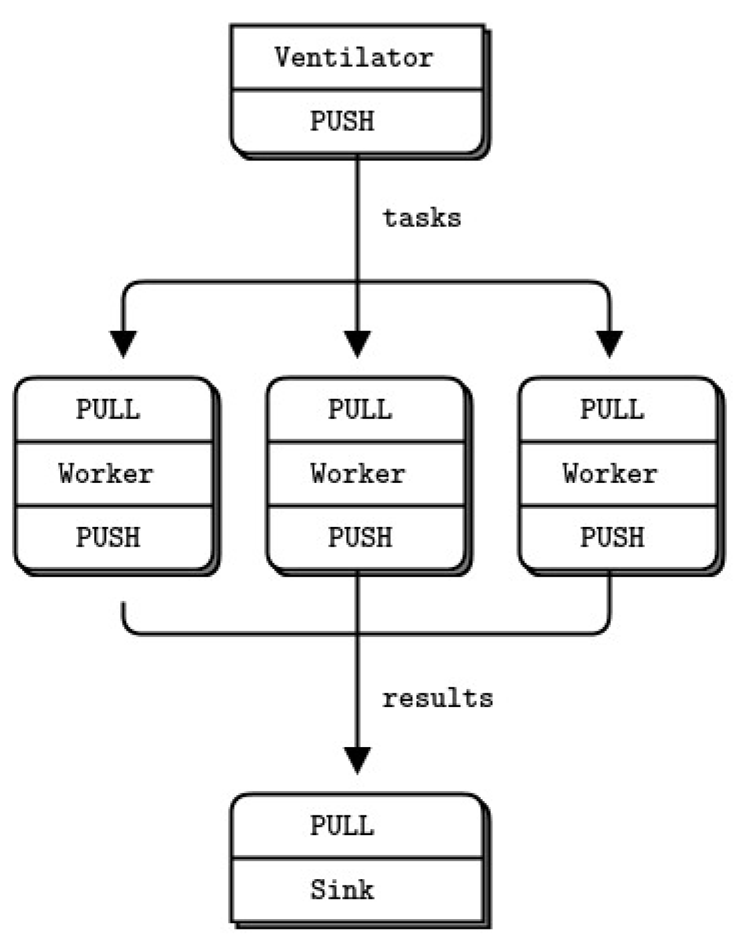
client:

```python
#!/usr/bin/python
#-*-coding:utf-8-*-

import time
import zmq
context = zmq.Context()
socket = context.socket(zmq.SUB)
socket.connect("tcp://127.0.0.1:5000")
socket.setsockopt(zmq.SUBSCRIBE,'')
while True:
    print  socket.recv()
```

3.Parallel Pipeline模式：

由三部分组成，push进行数据推送，work进行数据缓存，pull进行数据竞争获取处理。区别于Publish-Subscribe存在一个数据缓存和处理负载。

当连接被断开，数据不会丢失，重连后数据继续发送到对端。



server:

```python
#!/usr/bin/python
#-*-coding:utf-8-*-

import zmq

context = zmq.Context()

socket = context.socket(zmq.PULL)
socket.bind('tcp://*:5558')

while True:
    data = socket.recv()
    print data
```

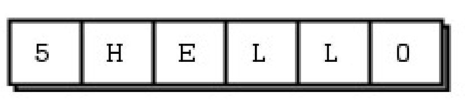
work:

```python
#!/usr/bin/python
#-*-coding:utf-8-*-

import zmq

context = zmq.Context()

recive = context.socket(zmq.PULL)
recive.connect('tcp://127.0.0.1:5557')

sender = context.socket(zmq.PUSH)
sender.connect('tcp://127.0.0.1:5558')

while True:
    data = recive.recv()
    sender.send(data)
```

client:

```python
#!/usr/bin/python
#-*-coding:utf-8-*-

import zmq
import time

context = zmq.Context()
socket = context.socket(zmq.PUSH)

socket.bind('tcp://*:5557')

while True:
    data = raw_input('input your data:')
    socket.send(data)
```

消息结构：
在每个消息buff前均会自带一个buff长度




```
18:20:54.112550 IP (tos 0x0, ttl 64, id 59554, offset 0, flags [DF], proto TCP (6), length 62)
    localhost.46374 > localhost.5555: Flags [P.], cksum 0xfe32 (incorrect -> 0x62d8), seq 114:124, ack 112, win 342, options [nop,nop,TS val 16607630 ecr 16601426], length 10
        0x0000:  4500 003e e8a2 4000 4006 5415 7f00 0001  E..>..@.@.T.....
        0x0010:  7f00 0001 b526 15b3 d553 a2ce bdc8 bc38  .....&...S.....8
        0x0020:  8018 0156 fe32 0000 0101 080a 00fd 698e  ...V.2........i.
        0x0030:  00fd 5152 0100 0006 3132 3334 3536       ..QR....123456
18:20:54.112677 IP (tos 0x0, ttl 64, id 61718, offset 0, flags [DF], proto TCP (6), length 72)
    localhost.5555 > localhost.46374: Flags [P.], cksum 0xfe3c (incorrect -> 0xcee2), seq 112:132, ack 124, win 342, options [nop,nop,TS val 16607630 ecr 16607630], length 20
        0x0000:  4500 0048 f116 4000 4006 4b97 7f00 0001  E..H..@.@.K.....
        0x0010:  7f00 0001 15b3 b526 bdc8 bc38 d553 a2d8  .......&...8.S..
        0x0020:  8018 0156 fe3c 0000 0101 080a 00fd 698e  ...V.<........i.
        0x0030:  00fd 698e 0100 0010 7365 7276 6572 2072  ..i.....server.r
        0x0040:  6573 706f 6e73 6521                      esponse!
```