Lab 03: using ZeroMQ to organize a distributed system

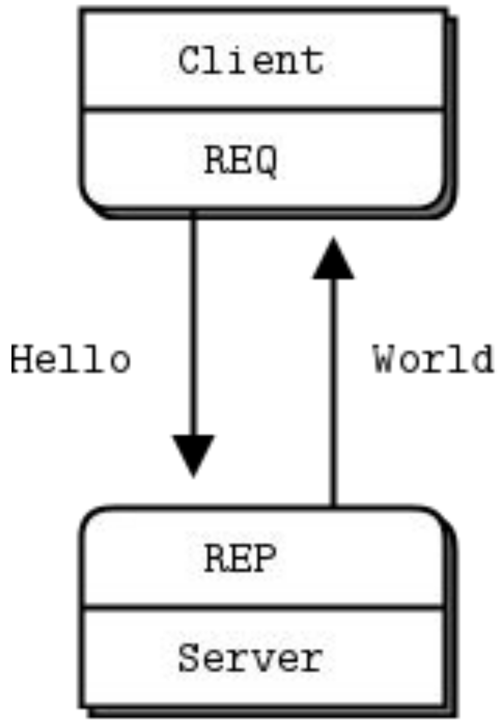
Distributed & network programming

Plan

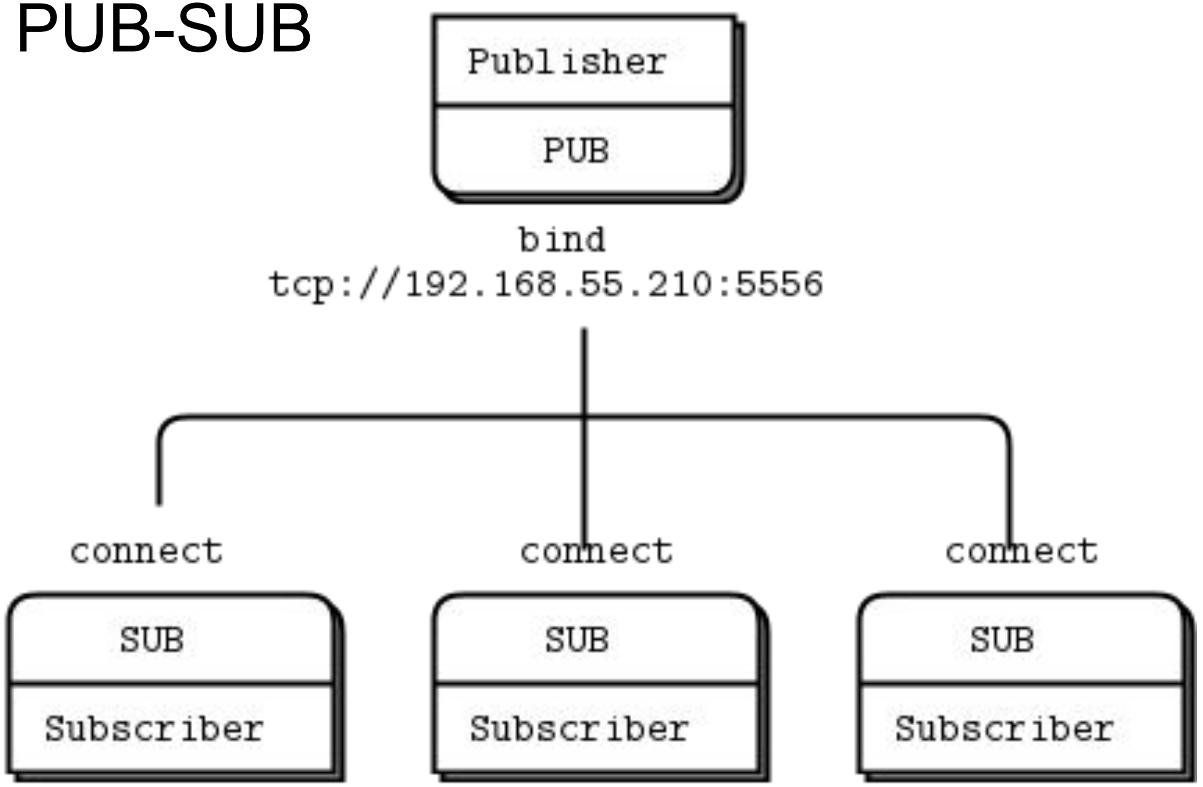
- Why people use message queues
- ZeroMQ composition patterns
- Lab assignment overview
- Useful pieces of code

Why use message queues?
What are they for?

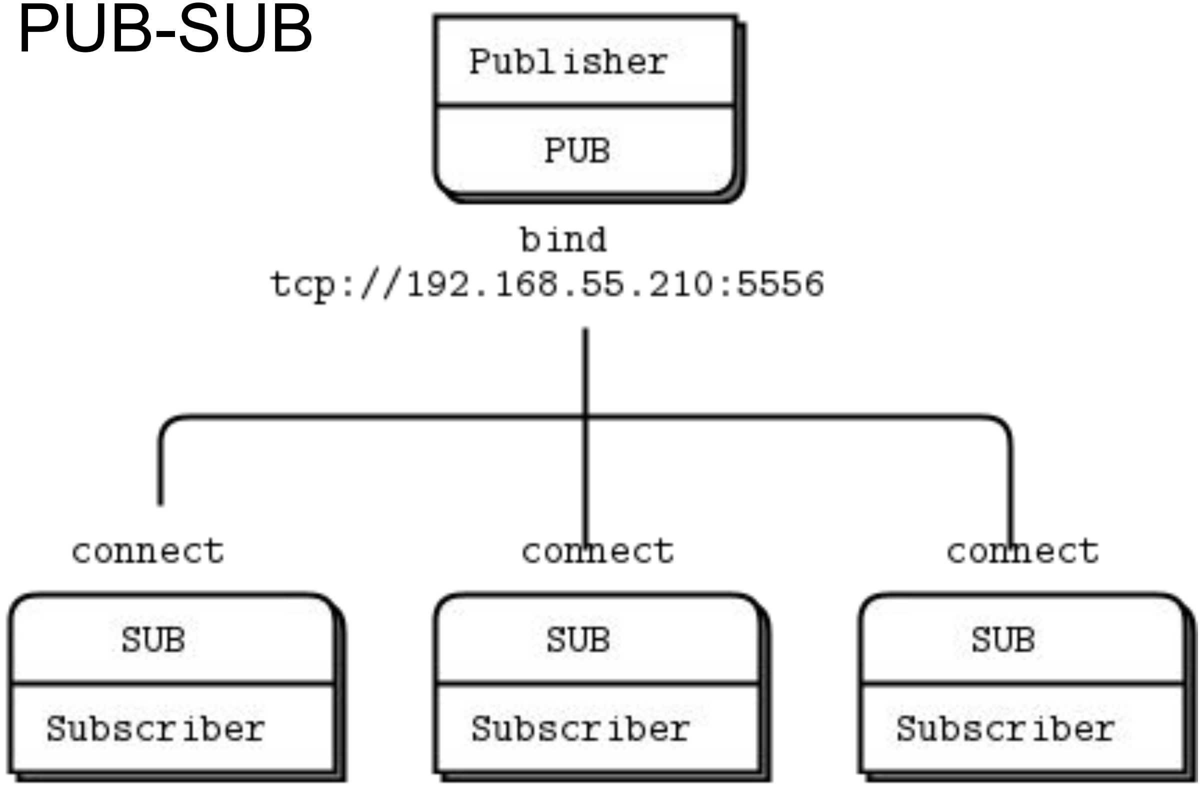
REQ-REP



PUB-SUB


Publisher
PUB
bind
tcp://192.168.55.210:5556
connect
connect
connect
SUB
Subscriber
SUB
Subscriber
SUB
Subscriber

PUB-SUB

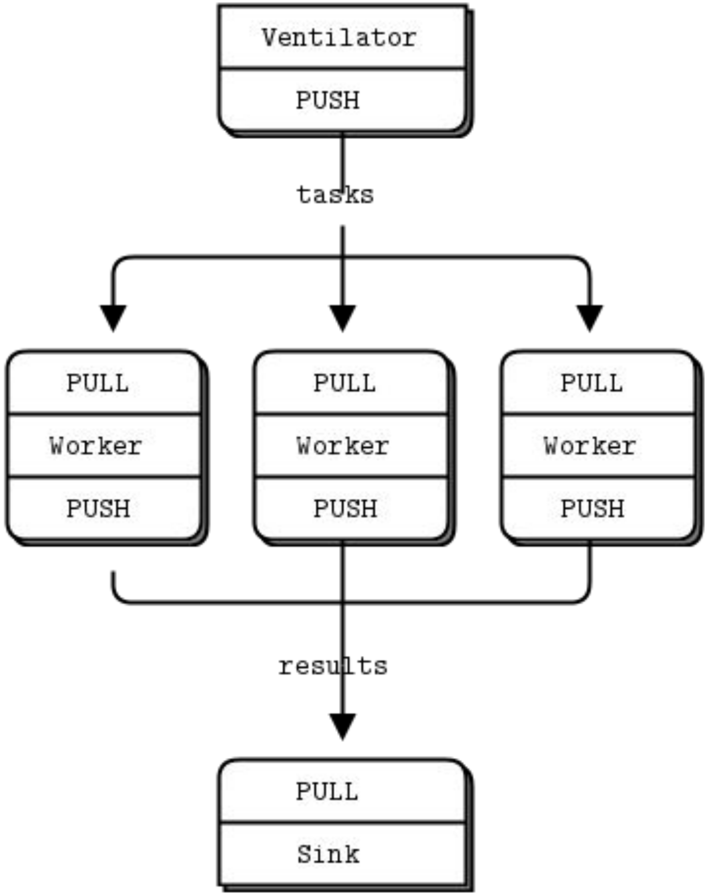


many inputs
one output

one input
many outputs

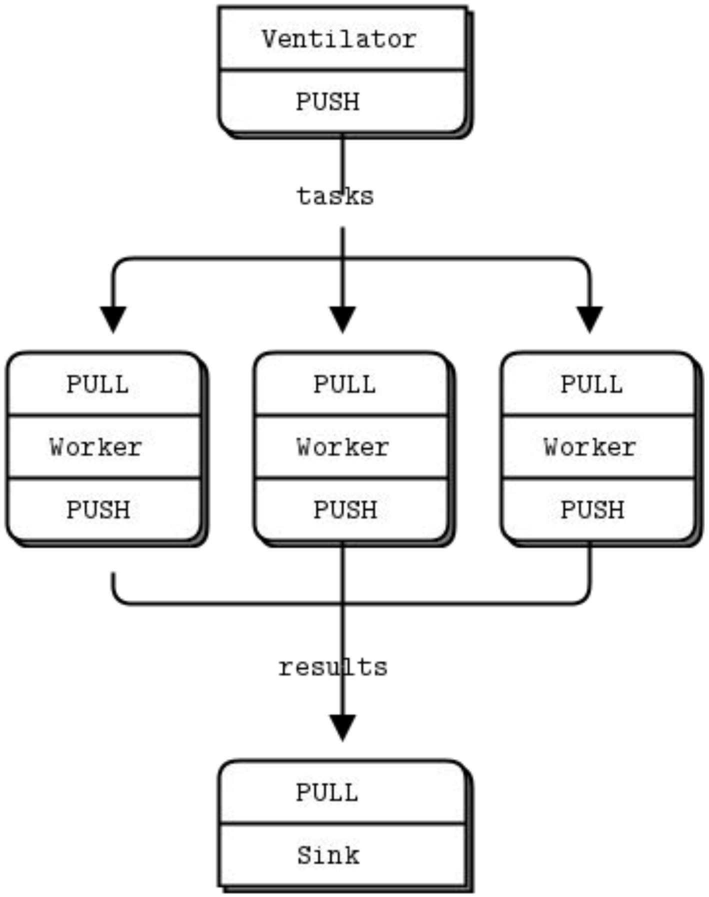
~~many inputs~~
~~many outputs~~

PULL-PUSH


Ventilator
PUSH
tasks
PULL
Worker
PUSH
PULL
Worker
PUSH
PULL
Worker
PUSH
results
PULL
Sink

PULL-PUSH

Messages are not duplicated



How it supposed to run

```
python3 server.py 5555 5556 5557 5558

python3 gcd.py 5557 5558

python3 primer.py 5557 5558

python3 client.py 5555 5556

python3 client.py 5555 5556
```

client.py

1) Connect to server ZeroMQ sockets: client_inputs, client_outputs

2) Read a line from the terminal

3) Send line to ZeroMQ

4) Receive a message from client_outputs and print it

server.py

1) Binds ZeroMQ sockets: client_inputs, client_outputs, worker_inputs, worker_outputs

2) Receive message from the client_inputs, send the message to worker_inputs

3) Receive message from worker_outputs, send message to client_outputs

primer.py

1) connects to ZeroMQ sockets: worker_inputs, worker_outputs

2) Receive message from the worker_inputs

3) If message has following format “isprime N” then test number N for primeness

4) Send result to worker_outputs: “N is prime” or “N is not prime”

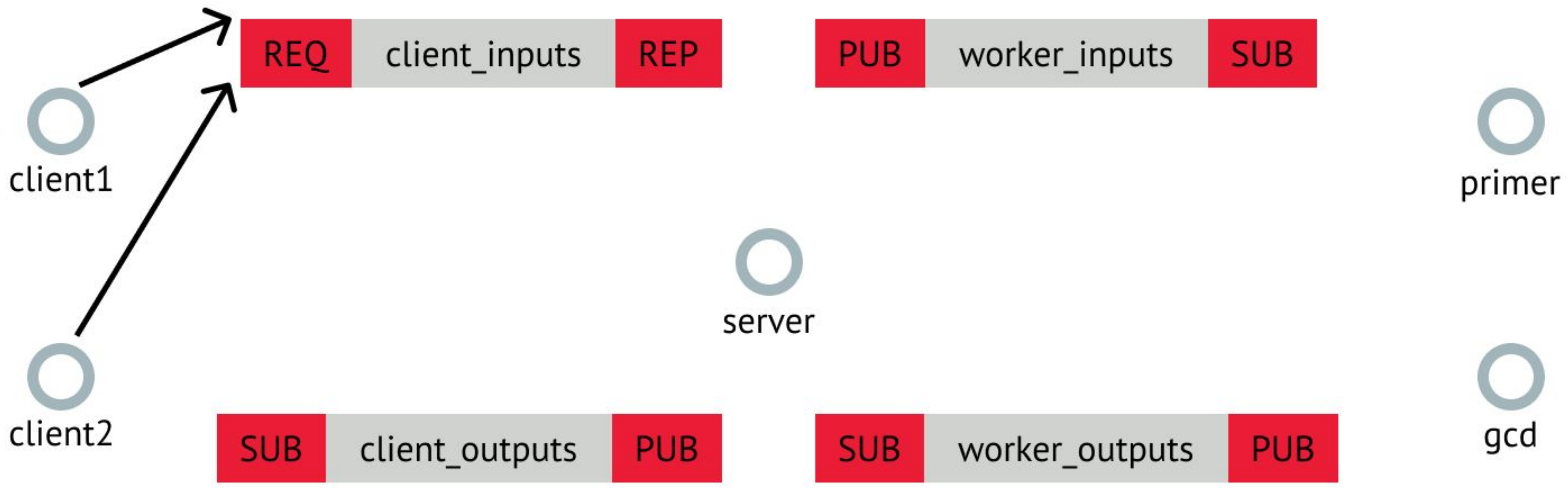
gcd.py

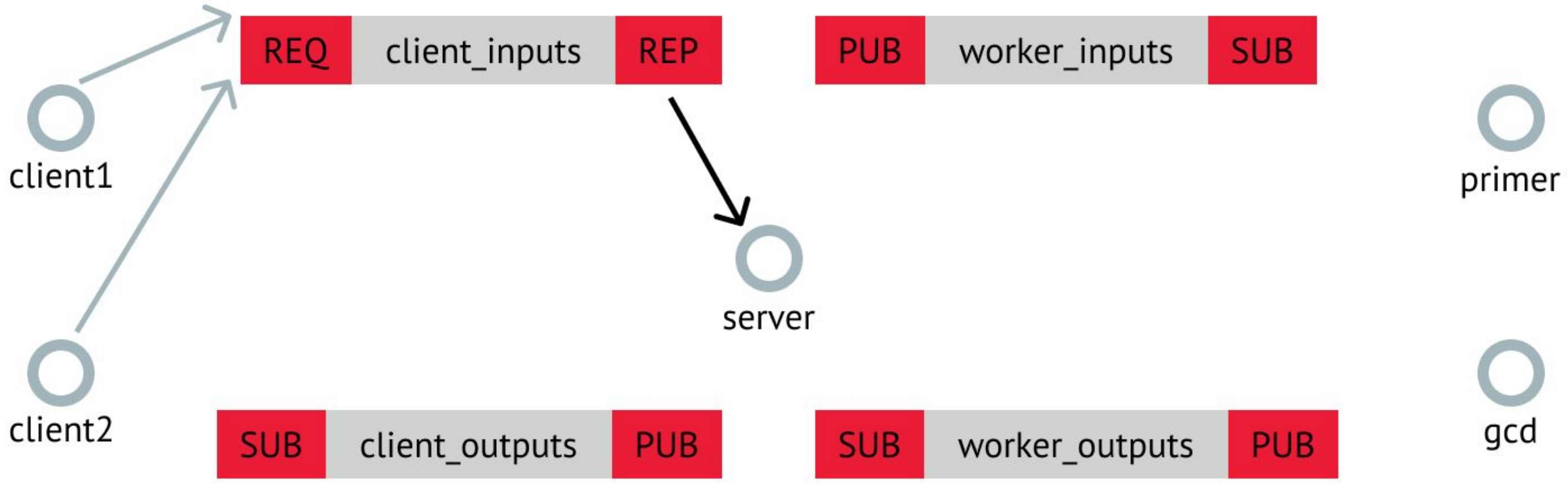
1) connects to ZeroMQ sockets: worker_inputs, worker_outputs

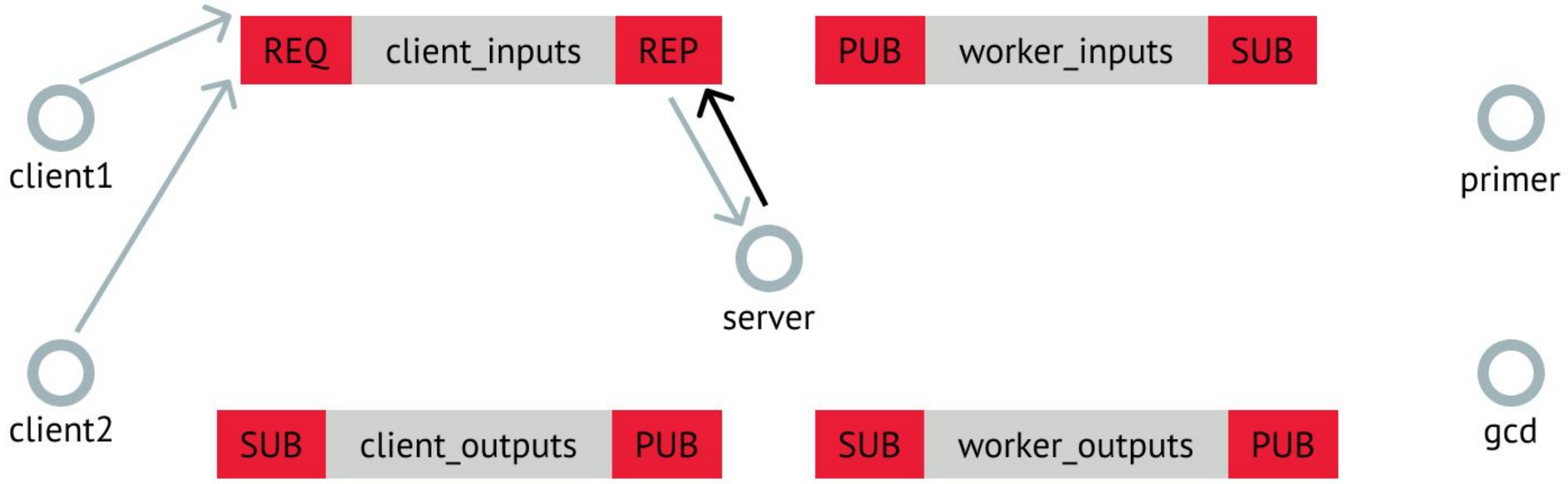
2) Receive message from the worker_inputs

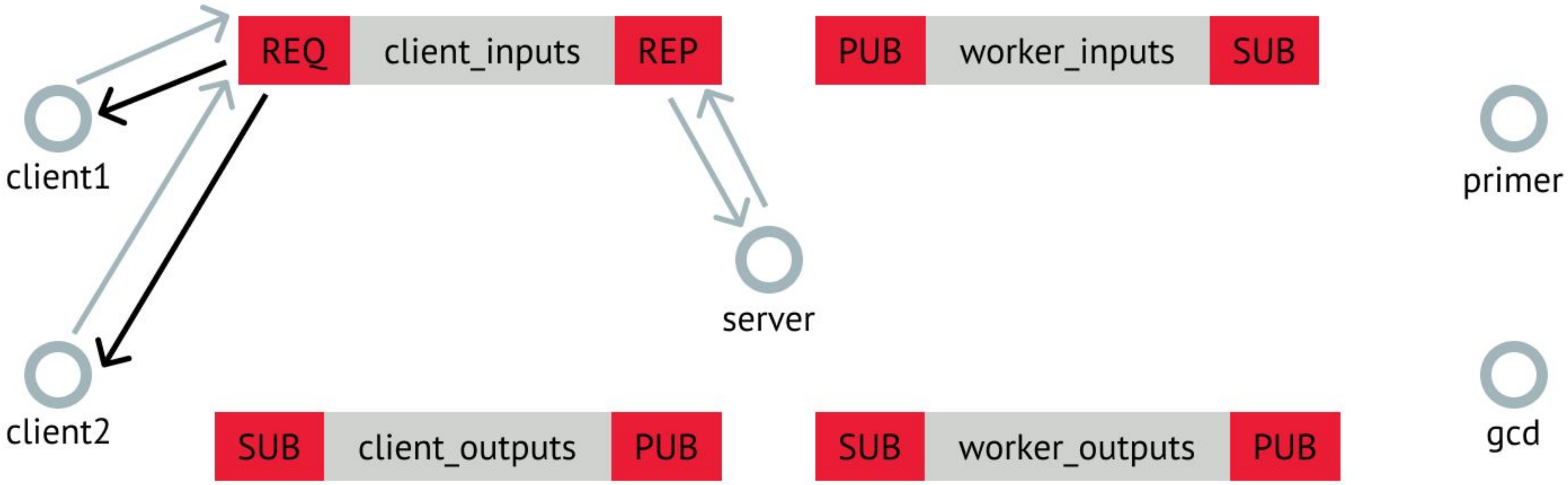
3) If message has following format “gcd A B” then computes Greatest Common Divisor for given two integers

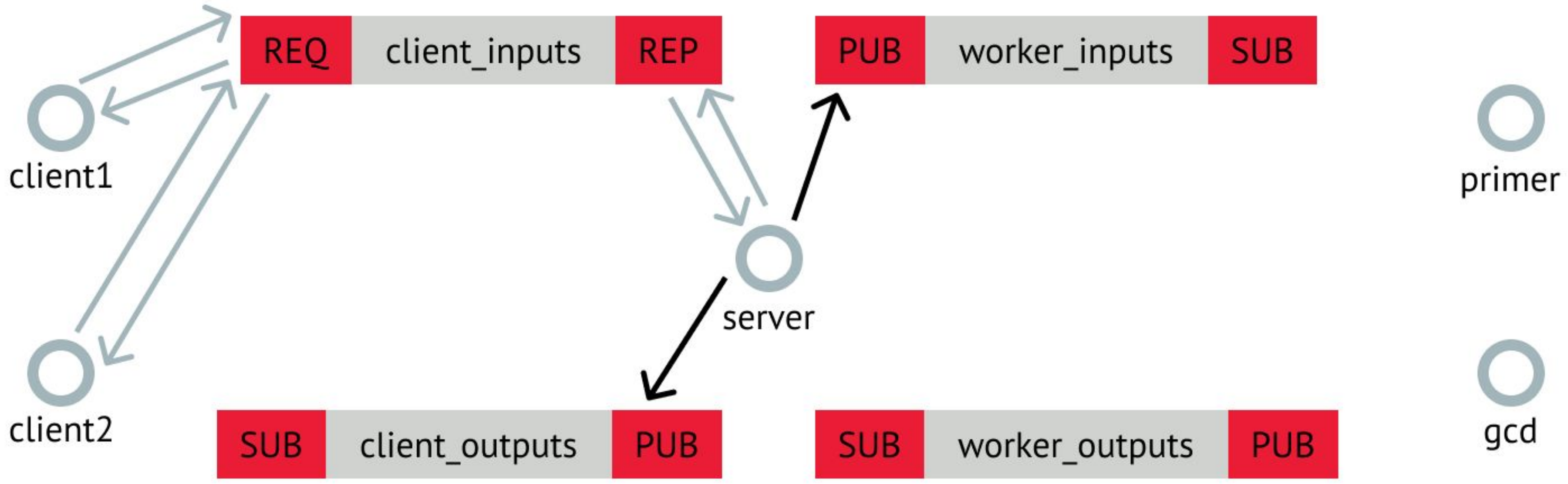
4) Send result to worker_outputs: “gcd for A B is C”

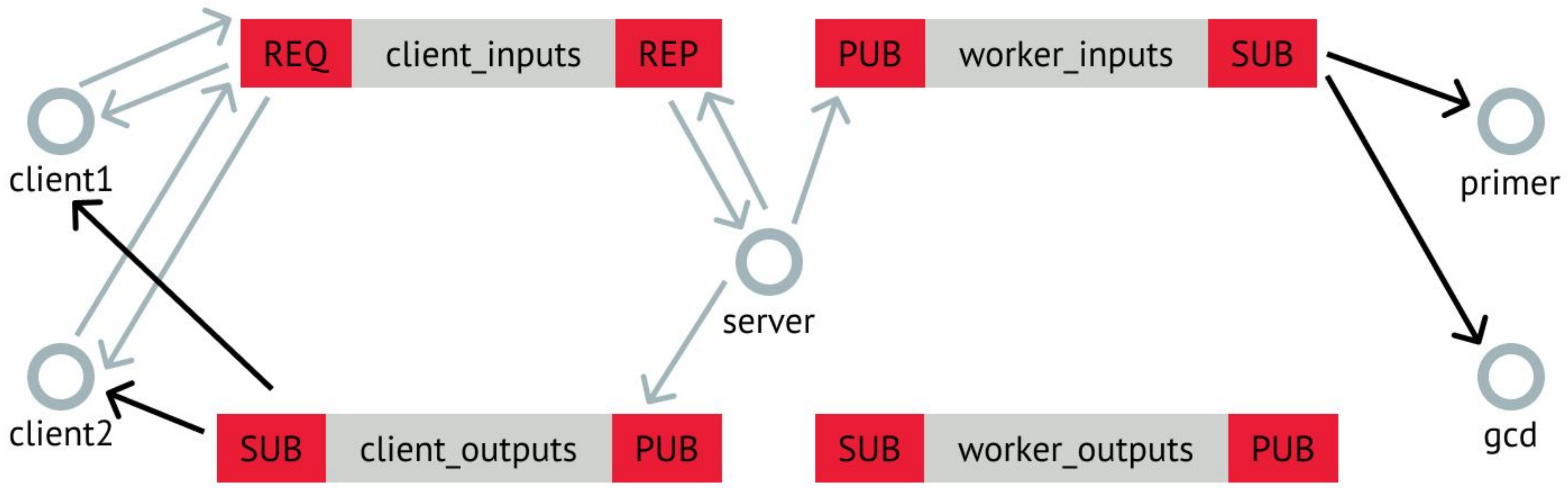

REQ
client_inputs
REP
PUB
worker_inputs
SUB
client1
primer
server
client2
gcd
SUB
client_outputs
PUB
SUB
worker_outputs
PUB

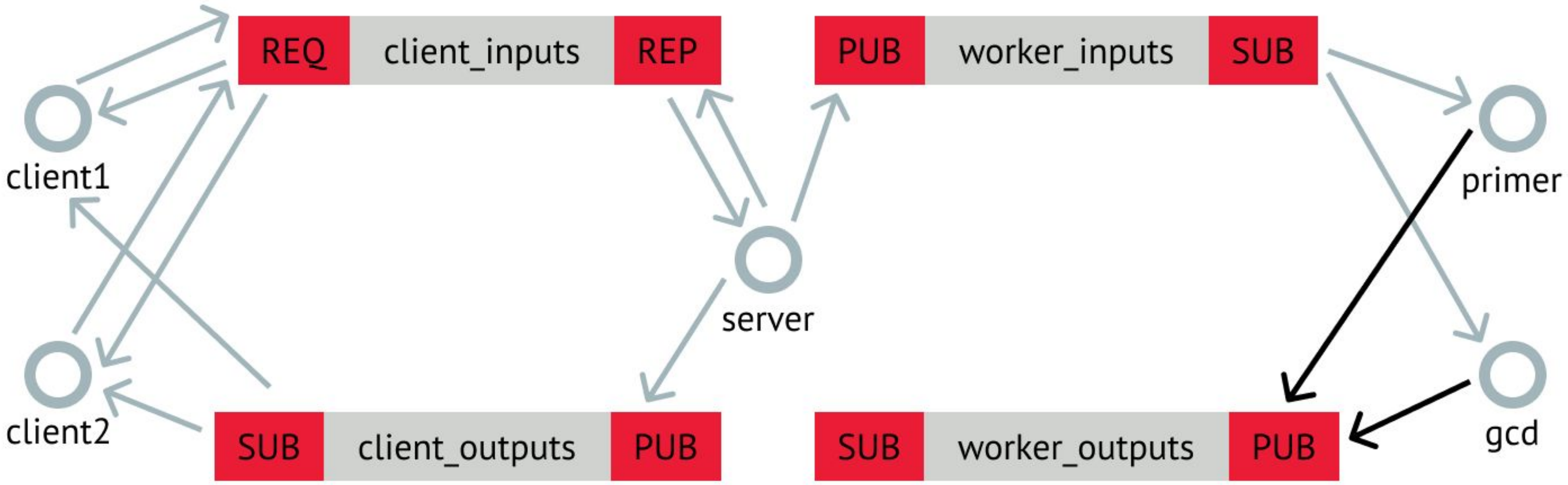

REQ
client_inputs
REP
PUB
worker_inputs
SUB
client1
primer
server
client2
gcd
SUB
client_outputs
PUB
SUB
worker_outputs
PUB

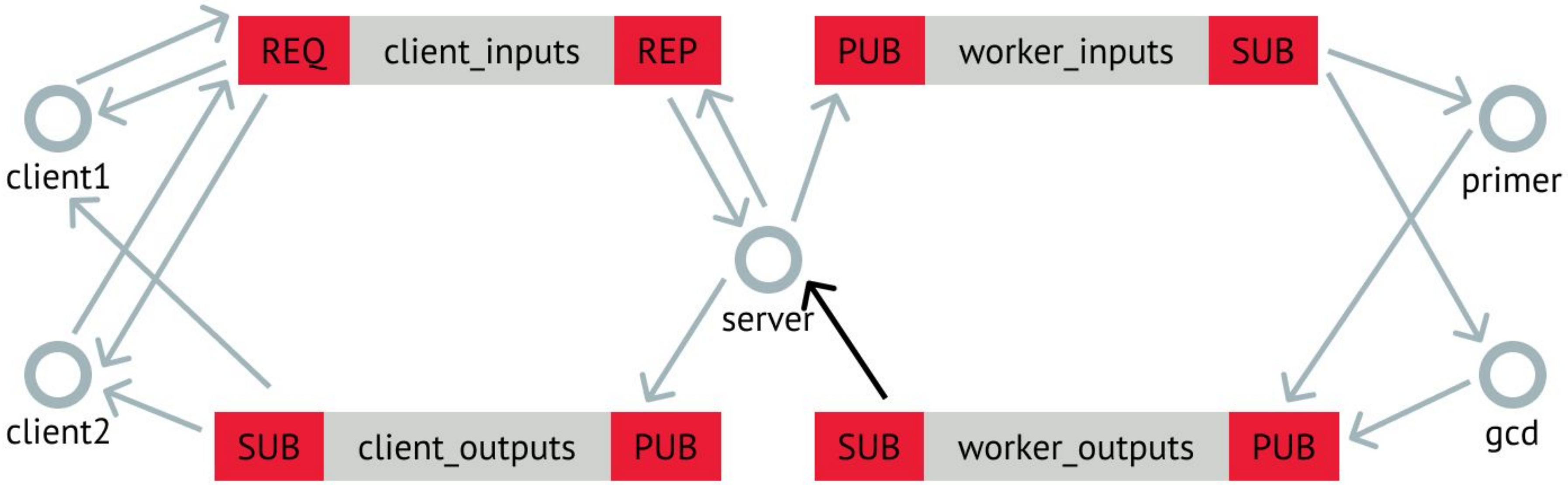

REQ
client_inputs
REP
PUB
worker_inputs
SUB
client1
primer
server
client2
gcd
SUB
client_outputs
PUB
SUB
worker_outputs
PUB

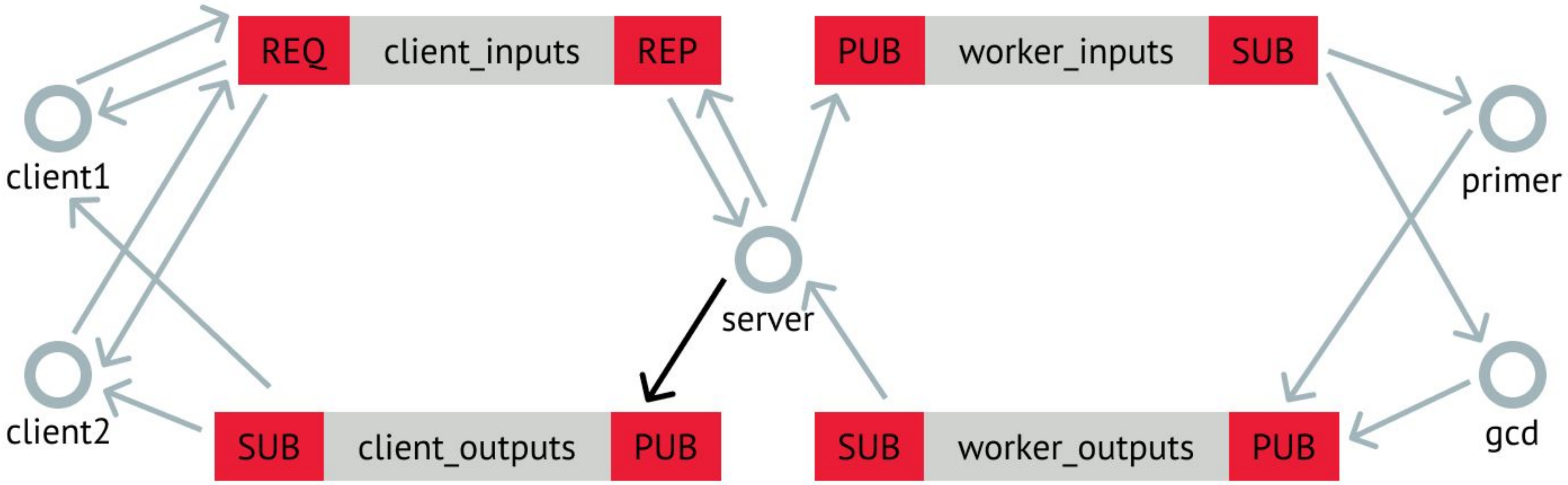

REQ
client_inputs
REP
PUB
worker_inputs
SUB
client1
primer
server
client2
gcd
SUB
client_outputs
PUB
SUB
worker_outputs
PUB

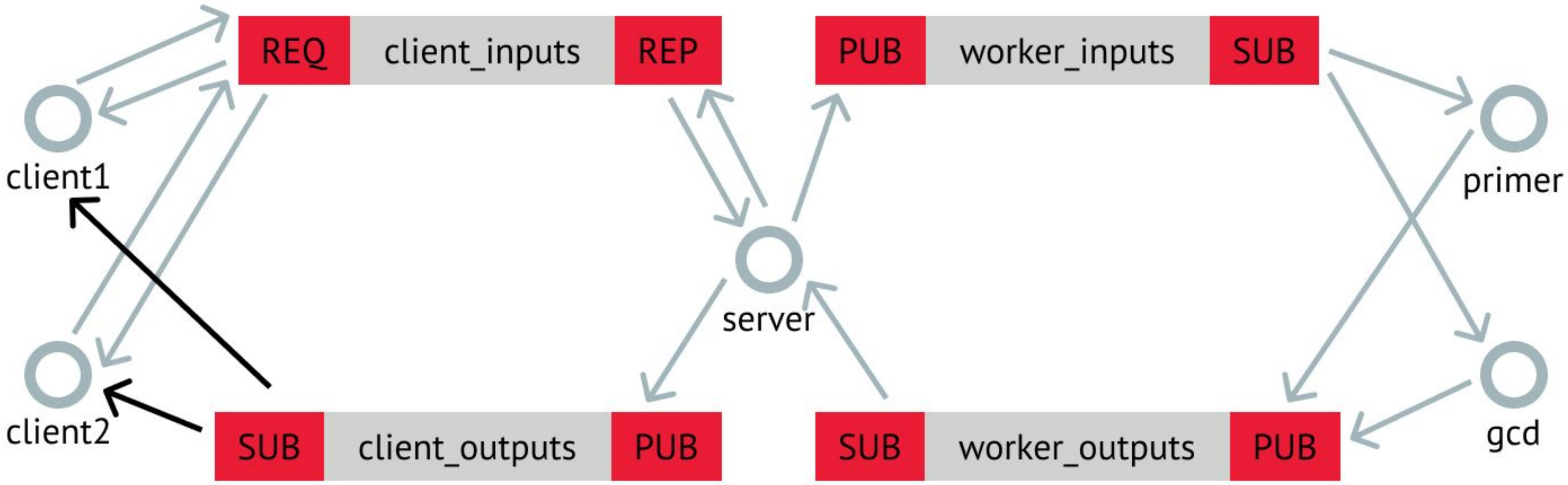

REQ
client_inputs
REP
PUB
worker_inputs
SUB
client1
primer
server
client2
SUB
client_outputs
PUB
SUB
worker_outputs
PUB
gcd


REQ
client_inputs
REP
PUB
worker_inputs
SUB
client1
primer
server
client2
gcd
SUB
client_outputs
PUB
SUB
worker_outputs
PUB


REQ
client_inputs
REP
PUB
worker_inputs
SUB
client1
primer
server
client2
gcd
SUB
client_outputs
PUB
SUB
worker_outputs
PUB


REQ
client_inputs
REP
PUB
worker_inputs
SUB
client1
primer
server
client2
gcd
SUB
client_outputs
PUB
SUB
worker_outputs
PUB


REQ
client_inputs
REP
PUB
worker_inputs
SUB
client1
primer
server
client2
gcd
SUB
client_outputs
PUB
SUB
worker_outputs
PUB


REQ
client_inputs
REP
PUB
worker_inputs
SUB
client1
primer
server
client2
gcd
SUB
client_outputs
PUB
SUB
worker_outputs
PUB

Useful pieces of code

```
import socket

context = zmq.Context()

sock = context.socket(zmq.SUB)

# we bind the socket, as a server

sock.bind(f"tcp://127.0.0.1:{port}")
sock.setsockopt_string(zmq.SUBSCRIBE, '')

msg = sock.recv_string()

# on PUB side:
# sock.send_string(“new message”)
```

Useful pieces of code

```python
import socket

context = zmq.Context()
sock = context.socket(zmq.SUB)
sock.connect(f"tcp://127.0.0.1:{port}")
sock.setsockopt_string(zmq.SUBSCRIBE, 'isprime')

# set a timeout for receive, make it non-blocking
sock.RCVTIMEO = 100

try:
    msg = sock.recv_string()
except zmq.Again:
    pass
```

Useful pieces of code

```
try:
    while True:
        line = input("> ")
        if len(line) != 0:
            # send request to client_inputs
            # receive confirmation
        try:
            while True:
                # try to receive from client_outputs
                # print if got anything
        except zmq.Again:
            pass
except KeyboardInterrupt:
    print("Terminating client")
    sys.exit(0)
```