Introduction

In this session we will see the distributed messaging framework zeromq. The guide is available at http://zguide.zeromq.org/. In particular, we will use it’s Python API http://zeromq.org/bindings:python. From the point of view of distributed systems we will study different messaging patterns that describe the flow of communication between interconnecting systems. We will see the following patterns: Client/Server, Publish/Subscribe, Push/Pull.

Question 1

a) The most basic pattern is Client/Server, where client sends a request and server replies to the request. Create a server (with a socket type zmq.REP) that can receive requests, print them and send back a response to the sender. Create a client (with a socket type zmq.REP) that sends messages to a server and waits for a response.

b) Test it by sending some number of requests to a server each time waiting for a response.

c) Run server at two different ports. Connect the client with two different servers simultaneously. Test it in a similar way.

d) Try starting the client and then starting the server, does it still work?

Question 2

a) Another messaging pattern is called Publish/Subscribe. Here senders of messages (publishers) do not send the messages directly to specific receivers (subscribers). Messages are broadcasted without any knowledge of receivers. Published messages have topics and the subscribers usually set filters on these topics. Create a publisher (zmq.PUB) and make it periodically send messages on various topics. Create a couple of subscribers (using zmq.SUB) each receiving messages on specific topics of their interest.

b) Create multiple publishers and connect to it one subscriber. Experiment with different frequency of messages coming from different publishers.

c) Try starting a publishing service first and then the subscribing one. What happens? Does the subscriber receives the messages that were published before it started.

Question 3

a) The last pattern we will study today is called Push/Pull. Push and Pull sockets let you distribute messages to multiple workers, arranged in a pipeline. A Push socket distributes sent messages to its Pull clients evenly.

b) Create a producer server (using zmq.PUSH) that sends a number of messages to its connected Pull clients.

c) Create a couple of consumer servers that receive messages from the producer (using a connection with zmq.PULL). Make these consumers modify the received message in a unique way

d) Create a result collecting server connected to all the consumers through zmq.PULL.

e) Connect all the consumers to this collecting server using zmq.PUSH and send each modified message to the result collector. Note that you should start your result collector and workers before you start your producers.