Working with Publish/Subscribe Pattern Using ZeroMQ

Use Cases

- In the case of publish/subscribe pattern, ZeroMQ is used to establish one or more subscribers, connecting to one or more publishers and receiving continuously what publisher sends (or seeds).
- Publish/subscribe pattern is used for evenly distributing messages across various consumers. Automatic updates for **scoreboards** and **news** can be considered as possible areas to use this solution.
- Socket type(s) used:
  - zmq.PUB
  - zmq.SUB

Publisher Example: pub.py

- Create a "pub.py" using nano (nano pub.py) and paste the below contents.

```
import zmq
import time
# ZeroMQ Context
context = zmq.Context()


# Define the socket using the "Context"
sock = context.socket(zmq.PUB)
sock.bind("tcp://127.0.0.1:5680")
id = 0
```

Publisher Example: pub.py

```
while True:
    time.sleep(1)
    id, now = id+1, time.ctime()
    # Message [prefix][message]
    message = "1-Update! >> #{id} >>
    {time}".format(id=id, time=now)
    sock.send(message.encode())
    # Message [prefix][message]
    message = "2-Update! >> #{id} >>
    {time}".format(id=id, time=now)
    sock.send(message.encode())
    id += 1
```

Subscriber Example: sub.py

- Create a "sub.py" using nano (nano sub.py) and paste the below contents.

```
import zmq
# ZeroMQ Context
context = zmq.Context()
# Define the socket using the "Context"
sock = context.socket(zmq.SUB)
# Define subscription and messages with prefix to accept.
sock.setsockopt(zmq.SUBSCRIBE, b"1")
sock.connect("tcp://127.0.0.1:5680")
while True:
    message= sock.recv()
    print(message.decode())
```

Subscriber Example: sub.py

- Note:

  Using the .setsockopt(..) procedure, we are subscribing to receive messages starting with string **1**. To receive all, leave it not set (i.e. "").

Usage

- Our **pub.py** is set to work as a **publisher**, sending **two different messages** - simultaneously - intended for different subscribers.
- Run the publisher to send messages:

```
# python pub.py
```

- On another window, see the print outs of subscribed content (i.e. 1):

```
# python sub.py!
1-Update! >> 1 >> Wed Dec 25 17:23:56 2018
```

-