Basics of ZeroMQ

Pushpendre Rastogi
January 31, 2021See `https://zguide.zeromq.org/docs/chapter1/` for more details.

ZeroMQ is a wrapper on top of TCP (and also three other transport protocols). It provides sockets, but these sockets come in pairs that map more closely to real applications. Also one ZeroMQ socket can have multiple underlying connections, which means that one ZeroMQ socket is actually a container for multiple underlying trasport layer level sockets.

There are three different types of sockets in ZeroMQ

1. REQuest - REPly Socket pair
2. PUBlist - SUBscribe socket pair
3. PUSH - PULL socket pair

The basic steps for using ZeroMQ are:

1. **Make context**
2. **Make socket**
3. **Configure Socket**
4. **Plug in** Call bind the socket to transport layer address and port if the process is a server and wants to make itself visible to others. Otherwise call connect. bind associates the socket with its local address, that's why server side bind s, so that clients can use that address to connect to server. connect is used to connect to a remote server address, that's why it is client side, connect to server is used. The parameter to bind and connect are protocol dependent. See ink for details on tcp.
5. **Data IO** Use the sockets in the application. All IO happens asynchronously on its own thread(s). The rule of thumb is to one IO thread per 1 GBPS.
6. **Destruction**

In each language the ZeroMQ api provides functions that implement these steps. So you can write a worker in Rust or C++ and a client in python. The difference between REP and REQ in the table below comes from making a server in C++ and a client in Python.

Step	C++	Python
Make context	`#include <zmq.hpp>; zmq::context_t context (1);`	`import zmq; context = zmq.Context()`
Make socket	`zmq::socket_t socket (context, ZMQ_REP);`	`socket = context.socket(zmq.REQ)`
Configure socket		
Plug in	`socket.bind ("tcp://*:5555");`	`socket.connect ("tcp://localhost:5555")`
Data IO	`zmq::message_t request; socket.recv (&request); zmq::message_t reply (5); memcpy (reply.data (), "world", 5); socket.send(reply)`	`socket.send(b"Hello"); message=socket.recv()`
Destroy	`zmq_close (socket); zmq_ctx_destroy (context);`	