Message Queues

Large Scale Distributed Systems

Message queue systems provide functionality that would be complex to build correctly and efficiently on top of standard TCP sockets. They support generic patterns such as request/reply interactions, publish/subscribe dissemination and task distribution. All this while handling buffering, re-connection, message atomicity and other properties. Here we will become familiar with the basic functionality of 0MQ (ZeroMQ) and some the supported patterns.

Suggested tasks for first class:

- Choose a language binding, e.g. C, and install the 0MQ system. Refer to `https://zeromq.org`.
- Using the guide at `https://zguide.zeromq.org` and the information in the *Basics* chapter, code and try the REQ/REP pattern. Try running the client first and then the server.
- If using C, download *zhelpers.h* and try to simplify the REQ/REP code used earlier.
- You can also try the PUB/SUB examples and play a bit with them.

Suggested tasks for second class:

- Study the `zmq_poll()` mechanism. It allows efficient access to multiple sockets from a single thread.
- Adapt the publisher in the first chapter to broadcast weather updates for PT zipcodes (4 digits). Adapt the client subscriber, using the poll mechanism, to fetch data from the two publishers, US and PT.
- Implement and test the *Shared Queue* pattern (with DEALER and ROUTER Sockets).
  - Try running a single client and worker, with the broker.
  - Try adding another worker and see the distribution of requests.

  - Try killing and re-starting the broker.

- Adapt the pattern above for the publish/subscriber case by using XSUB and XPUB patterns.