0.1 Source Archive

Alongside this report is the accompanying code for this proof-of-concept system containing everything needed for deployment. Project layout closely follows the system design **??** with the code for each node being contained in its own folder.

0.1.1 Building

As I have used Python for each node in my project, I was able to create a *setup* file to create the correct Python environment which can be called using *'pip3 install .'* while in the nodes' root directory.

As I am using docker, building manually is not recommended as there is a lot of environmental setup which must be done. Each nodes' project contains a *Dockerfile* which is used to setup, build and run each application. Each node folder contains a *bin* directory inside which are various executables one of which, *docker_entrypoint*, is ultimately where we start the systems but the recommended way to start an individual node is to use the *build_and_run* executable which combines these steps for ease of use.

0.1.2 Running

You can independently start each node through Docker with the *'docker run'* command (conveniently you can run *build_and_run* with the parameter *runme*).

This is unnecessary however as Docker provides a tool, Compose, which is used for defining and running multi-container Docker applications. This means I can define a *docker-compose.yml* file in the root directory which will start all of the nodes, with the correct parameters, in the correct order.

This is the recommended way to run this application so calling *'docker-compose up'* while in the root directory should be the most hassle free.

0.1.3 Notes

- **Helper Files** - There are a number of helper scripts located in the *HelperScripts* folder which allow for easy interaction with the Docker nodes while running.
- **Blockchain download** - As this runs a full Ethereum node and must download the full blockchain before it can push transactions, an initial download of around 20GB will occur on first run.