LAB 11: Learn Blockchains with Python

Simulating one, empowered by MQTT

Background

- We want to simulate a distributed system responsible to keep a **consistent** view on the **history of transactions**...
- ...you know well that we need it to defuse the **double spending** problem, don't you?

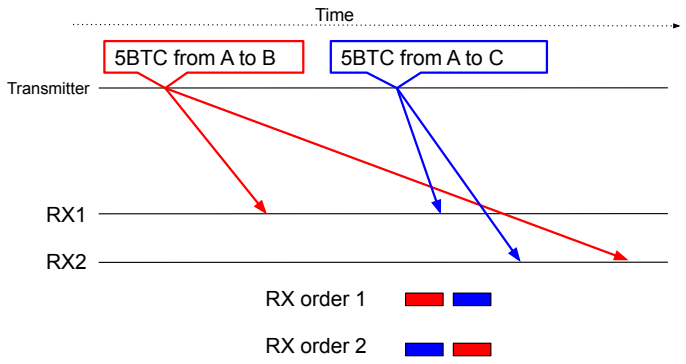

Figure 1: Propagation Delays raise the double-spending problem, which requires a consensus protocol to define a globally agreed, unique order of TRXs

- So we want multiple nodes to keep this DB in form of... a **BLOCKCHAIN**

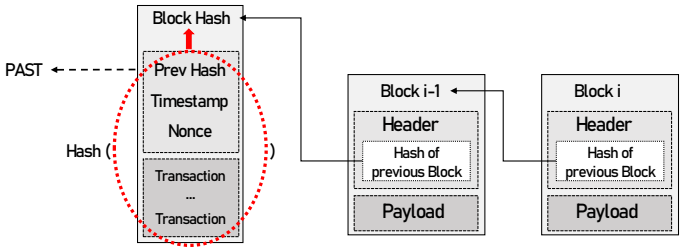

Figure 2: Fundamental elements of a blockchain

- Anyway, let's focus for today only on the main, high-level perspective of what should be the **lifecycle of a TRX** in a blockchain-based distributed system

Sketch idea of today

Keeping the illusrated TRX-lifecycle as leading example today we will implement/setup:

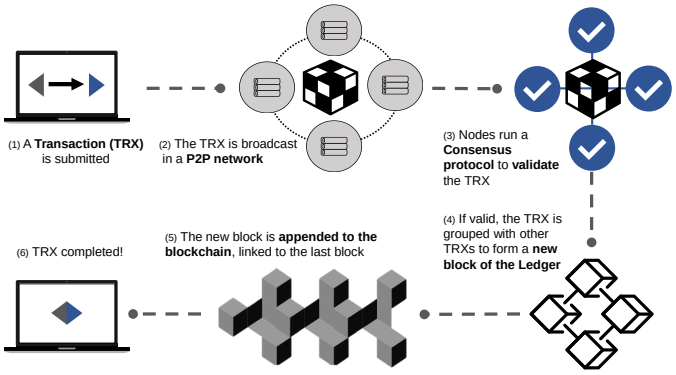

Figure 3: Lifecycle of a TRX

1. TRX generation with each miners regularly injecting new TRXs
2. TRXs and BLOCKs propagation in broadcast, simulating all-to-all communication[1] thanks to MQTT
3. We DON'T VALIDATE TRXs (for today they represent fake content for our blocks)
4. Still, we force the publication of blocks subject to the requirement of including a **Proof of Work**

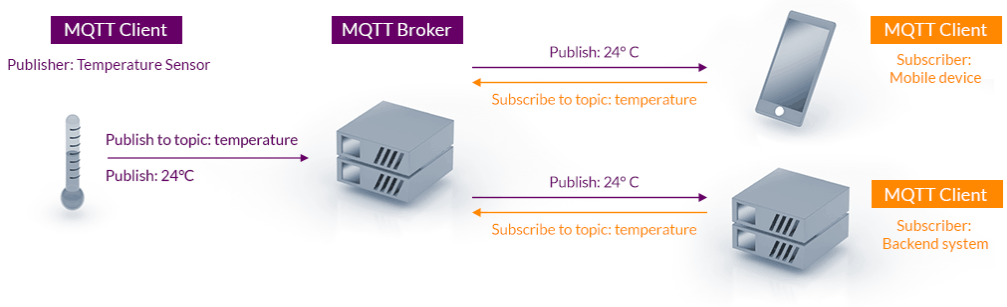

Figure 4: MQTT architecture

Architecture

- Our systems will be made by many *miners*.
- Each miner can *publish* NEWTRX and NEWBLOCK and subscribes to NEWTRX and NEWBLOCK topics
- Each miner *continuosly* injects NEWTRX –> **genTRX** thread
- Each miner *continuosly* tries to mine block –> **mining** thread

According to this architecture, we are left to code more or less just these:

1) Miner initialization
2) **genTRX** and **mining** Continuous daemon threads
3) SEND/RECEIVE handlers for NEWTRXs and NEWBLOCKs

are you ready??? Let's go!!!

[1] or relay networks, e.g., the one of Algorand

Installation

```
sudo apt install mosquitto
sudo apt install mosquitto-clients
pip install paho-mqtt
```

Coding together

1. We want many miners running on localhost...
   - miner.py –> miner Class
   - runminer.py –> usual "bootloader" script
   - util.py –> to keep code not too much messy :)
2. Initialize MQTT clients and message handlers
   - 2.1 Init also blockchain with **GenesisBlock()**
3. Implement runminer.py and deploy one miner, check with mosquitto-clients it can receives messages
4. Implement **genTRX**, deploy two nodes, check they exchange TRXs
5. MINING... WHILE TRUE:
   - 5.1 Prepare the **workingBlock**
   - 5.2 Compute proof –> not a good proof? continue
   - 5.3 Good proof?!? You mined a bloooock!!! Publish it!
     - 5.3.1 Blocked TRXs are not pending anymore...
     - 5.3.2 Append block to blockchain
     - 5.3.3 Propagate block (MQTT publish)

Online resources

- how to use mqtt in python
- Managing Mosquitto on UBUNTU
- How to use Mosquitto clients