WiFi bandwidth differetiation service using smart contract

Tian Runxin, 2020.3.12

Outline

1. Tools : The development tools I used.
2. Demo & Features
3. Some progress: Utils and Test cases
4. Problems
5. Plans

Tools

1. Geth - Go Ethereum: Official Go implementation of the Ethereum protocol.

```
geth <other-options> --dev --dev.period 1 # PoA
```

2. Docker
3. Truffle Suite
   - **Truffle**: the most popular development framework
   - Ganache: One click blockchain testnet.
   - **Drizzle** (Font-end): Fast response.

Geth

Clique PoA consensus

```
node_1  | INFO [03-11|22:12:31.516] Sealing paused, waiting for transactions
node_1  | INFO [03-11|22:12:31.517] Commit new mining work
node_1  | INFO [03-11|22:12:31.521] Commit new mining work
node_1  | INFO [03-11|22:12:31.524] Successfully sealed new block
node_1  | INFO [03-11|22:12:31.525] 🔗 block reached canonical chain
node_1  | INFO [03-11|22:12:31.525] 🔨 mined potential block
node_1  | INFO [03-11|22:12:31.526] Commit new mining work
node_1  | INFO [03-11|22:12:31.528] Sealing paused, waiting for transactions
```

Docker

- For deployment convenience, I builtup a docker container, which can easily run the **PoA** blockchain network.

Why PoA (Clique PoA consensus)

- Allow blocks to be mined as-needed without excessive CPU and memory consumption.
- Produce blocks when there are transactions waiting to be "mined".

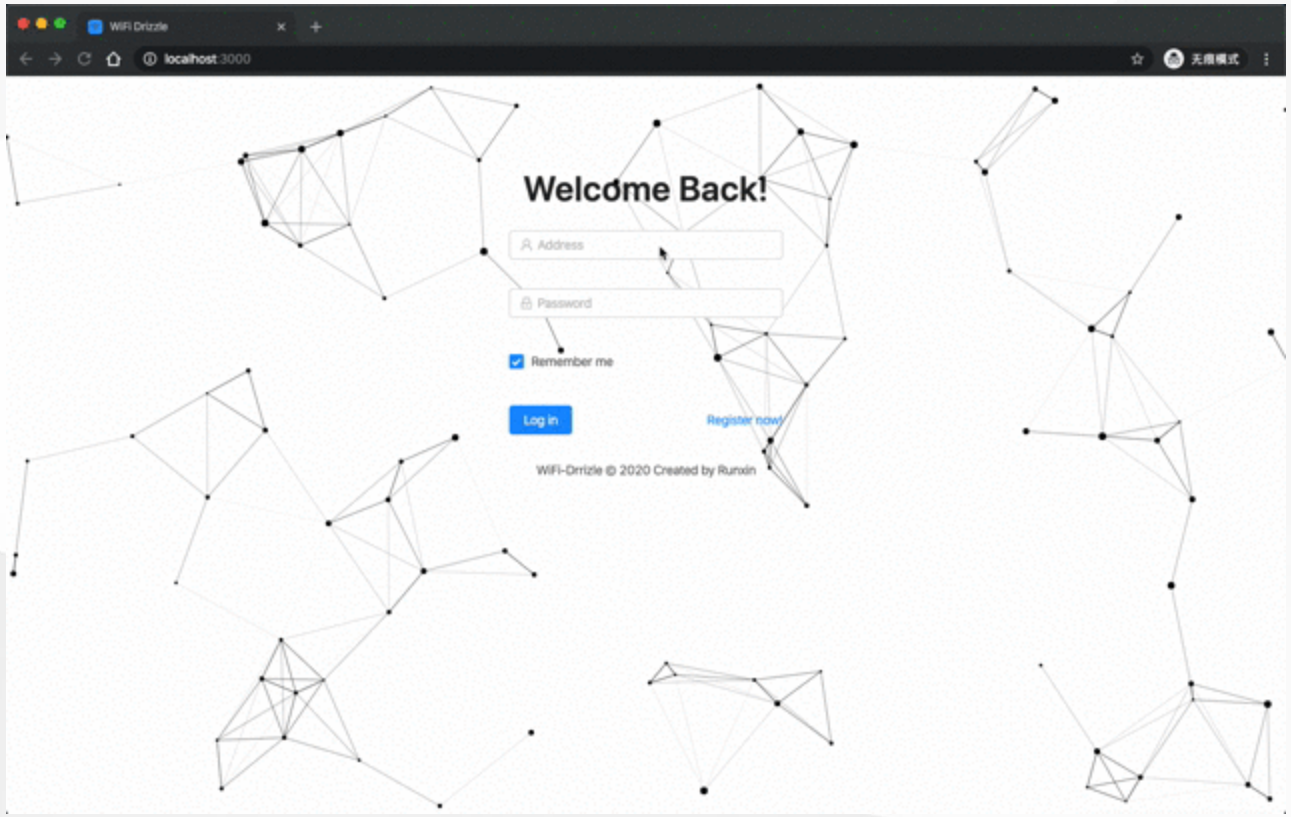
Truffle Suite

```
(base) ➜  drizzle-box git:(dev) truffle migrate

Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.



Starting migrations...
======================
> Network name:    'development'
> Network id:      19990119
> Block gas limit: 0x5fdfb1
```

1. Truffle:
   Contract compilition, deployment, and testing.

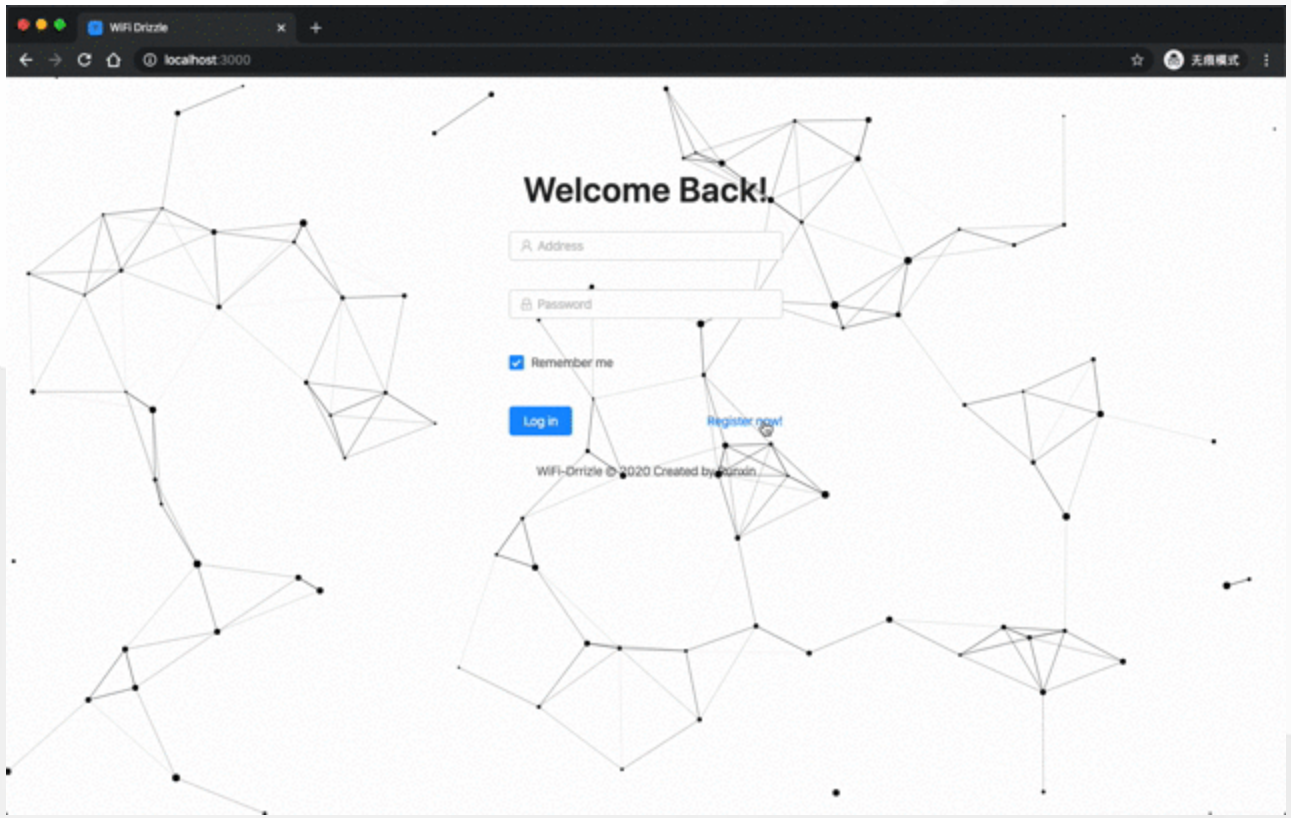
2. Drizzle:
   React-based font-end framework.
   (Fast response but buggy)

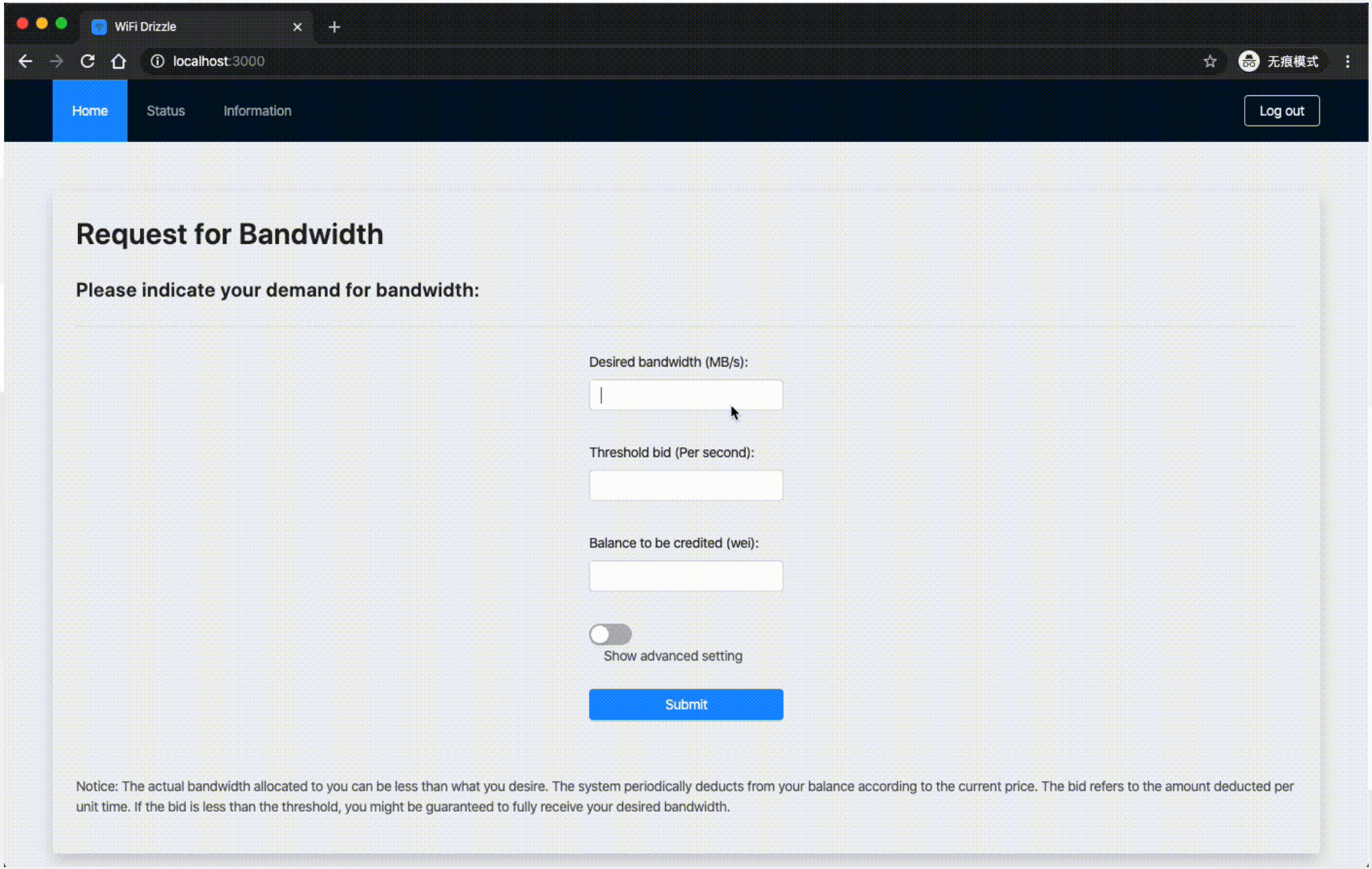
Features - Log in

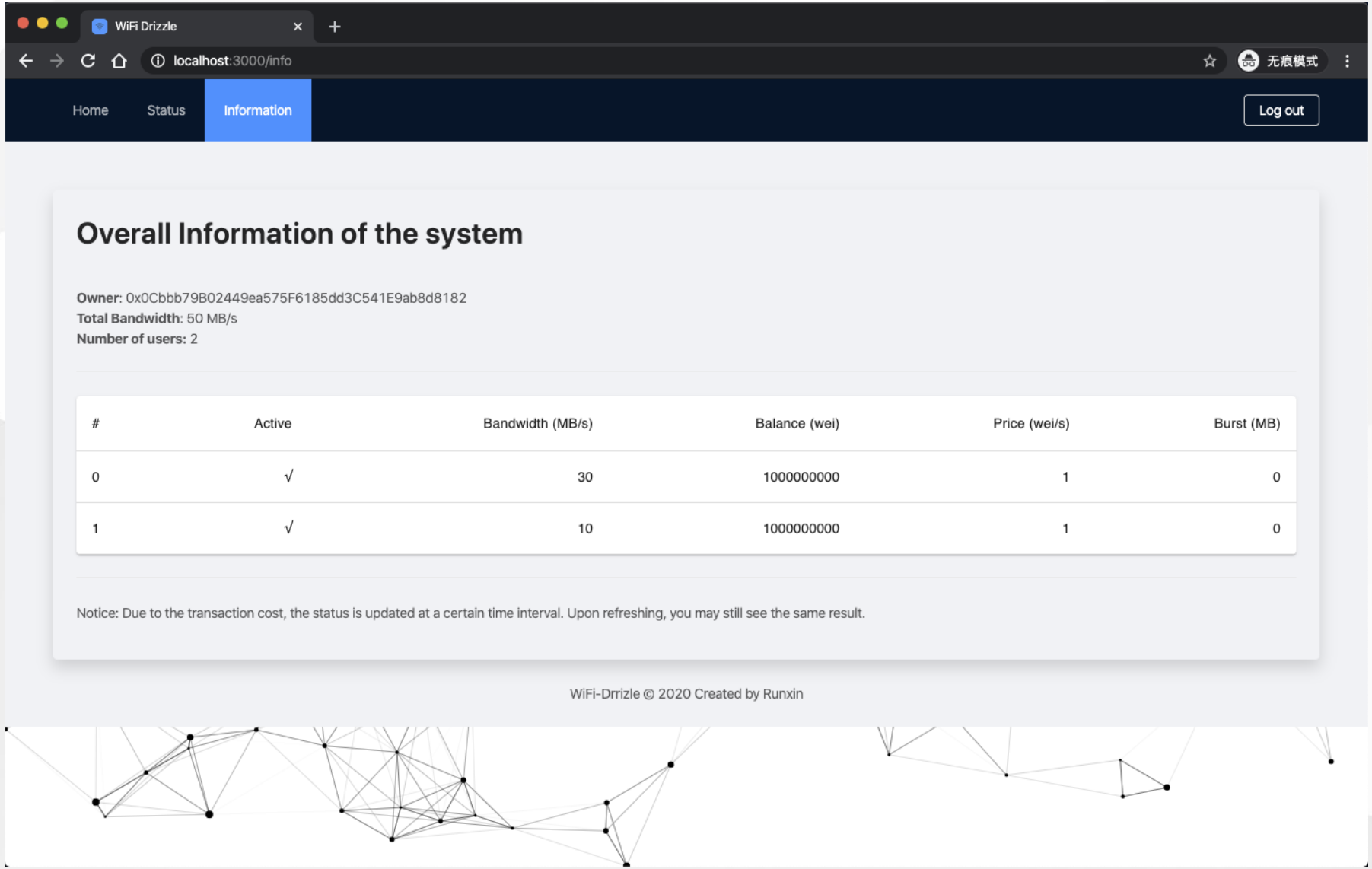


- Use `web3.eth.personal.unlockAccount` to verify the user.
- Both `password` and `privatekey` are able to verify.

Features - Register




WiFi Drizzle
localhost:3000
无痕模式
Home
Status
Information
Log out
Request for Bandwidth
Please indicate your demand for bandwidth:
Desired bandwidth (MB/s):
Threshold bid (Per second):
Balance to be credited (wei):
Show advanced setting
Submit
Notice: The actual bandwidth allocated to you can be less than what you desire. The system periodically deducts from your balance according to the current price. The bid refers to the amount deducted per unit time. If the bid is less than the threshold, you might be guaranteed to fully receive your desired bandwidth.


WiFi Drizzle
localhost:3000/info
无痕模式
Home
Status
Information
Log out
Overall Information of the system
Owner: 0x0Cbbb79B02449ea575F6185dd3C541E9ab8d8182
Total Bandwidth: 50 MB/s
Number of users: 2
Active Bandwidth (MB/s) Balance (wei) Price (wei/s) Burst (MB)
0 √ 30 1000000000 1 0
1 √ 10 1000000000 1 0
Notice: Due to the transaction cost, the status is updated at a certain time interval. Upon refreshing, you may still see the same result.
WiFi-Drrizle © 2020 Created by Runxin

WiFi Drizzle

localhost:3000/info

Home Status Information Log out

Overall Information of the system

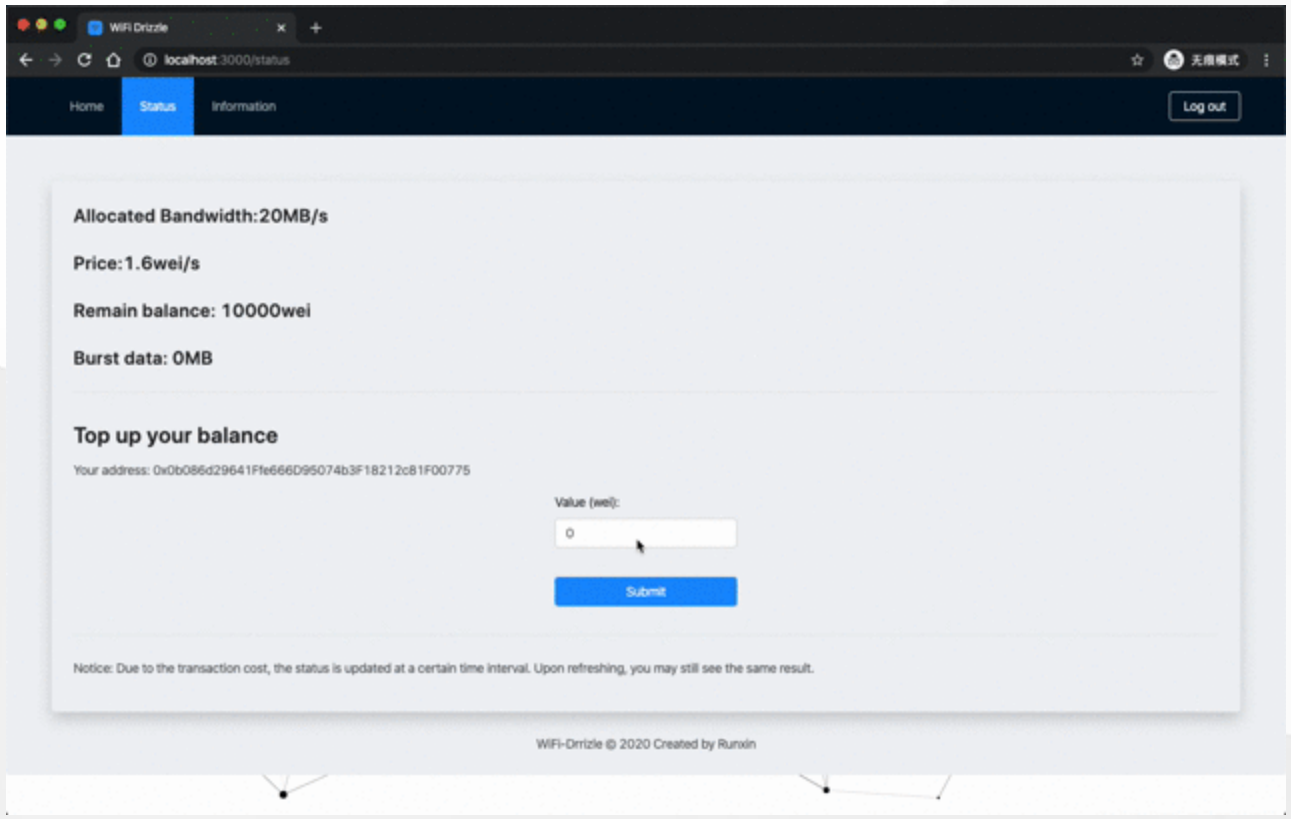
Owner: 0x0Cbbb79B02449ea575F6185dd3C541E9ab8d8182
Total Bandwidth: 50 MB/s
Number of users: 2

#	Active	Bandwidth (MB/s)	Balance (wei)	Price (wei/s)	Burst (MB)
0	√	30	1999994930	1	0
1	√	10	1999998310	1	0

Notice: Due to the transaction cost, the status is updated at a certain time interval. Upon refreshing, you may still see the same result.

WiFi-Drrizle © 2020 Created by Runxin

Features - Top up balance



Some Utils

```
scripts
├── clear_allocation.js
├── clear_balances.js
├── get_balances.js
├── get_num_users.js
├── initial_balances.js
├── pre_allocation.js
└── target_allocation.js
```

Test cases and result

```
4   contract("WiFiAllocation", accounts=>{
5     it('The total bandwidth should be 50 MB/s.', async () => {...});
12
13    it('The owner should be the account[0]. ', async () => {...});
18
19    it('There should be 5 users.', async () => {
20      const instance = await WiFiAllocation.deployed();
21      await instance.uponConnection(10,5,0, {from:accounts[1], gas:3000000, value:1});
22      await instance.uponConnection(10,6,0, {from:accounts[2], gas:3000000, value:1});
23      await instance.uponConnection(10,10,0, {from:accounts[3], gas:3000000, value:1});
24      await instance.uponConnection(10,4,0, {from:accounts[4], gas:3000000, value:1});
25      await instance.uponConnection(10,1,0, {from:accounts[5], gas:3000000, value:1});
26      const numUsers = await instance.numUsers.call();
27      assert.equal(numUsers,  expected: 5,  message: "Wrong # of users!")
28    });
29
30    it('Display allocated bandwidth', async () => {...});
39
40    it('There should be no user.', async () => {...});
50
51  });
```

Test result

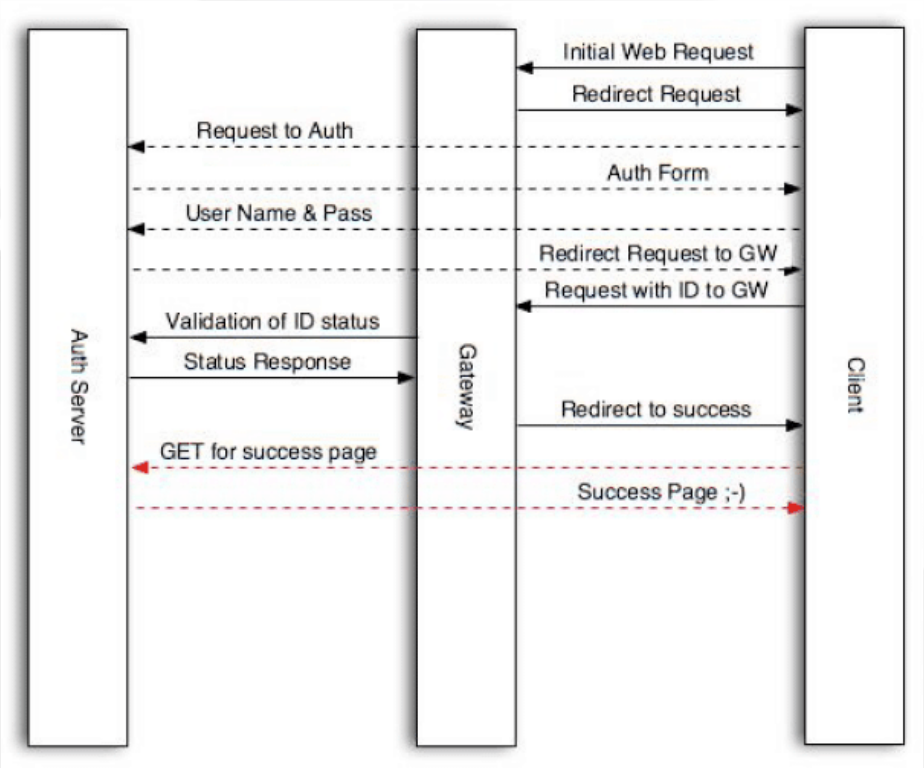
```
Contract: WiFiAllocation
  ✓ The total bandwidth should be 50 MB/s.
  ✓ The owner should be the account[0].
  ✓ There should be 5 users. (2249ms)
```

which means it takes around 450ms
for every user entering the system

Problems

1. Can not run docker or geth on raspberry pi.
    - tried solutions:
        1. install a 64-bit OS (failed)
        2. reinstall OS (trying)
2. Whether use a new wifi control framwork.

WiFi Dog



- An open source captive portal solution.
  (with DNS cache poisoning to redirect the user to authentication page)
  1. Location-aware delivery of internal or external content
  2. Authentication and authorization
  3. Centralizaed network monitoring

Plans

1. solve the problems with raspberry pi
2. try the captive portal solution approach