Simplified Ethereum-based prototype

Simplified B-RAN prototype using go-ethereum(geth).

Prerequisites

- *Ubuntu 20.04*.2.0 LTS
- *Python* 3.7
- *Geth* 1.9.13-stable

  The approach to installing geth can be found here.

Running the tests

You can run the following command from the root directory to start the test.

```
sh sim.sh
```

`sim.sh` executes `sh start_ue.sh`, `sh start_ap.sh` and `sh start_miner.sh` successively, thus starting geth nodes corresponding to user equipment, access point, and miner respectively, and we use "resmon" to measure the runtime metrics of each node. Then, it constructs Ethereum private network with these geth nodes by executing `sh build_env.sh`.

After the private Ethereum network is built, `python3 ap_server.py` and `python3 ue_client.py` are executed, which enables the user equipment to deploy smart contract `BranService.sol` to local blockchain network in a Poisson distribution (the total number of contracts is MAX_REQ_NUM, which can be modified in ue_client.py) and enables access points to obtain their own service requests (i.e., smart contracts of BranService) in the local blockchain network, start the service after valid confirmation of the blockchain, and then perform fee settlement.

Directory and file description

`ap_keystore`, `ue_keystore`, and `miner_keystore` stores the private key of user equipment, access point, and miner respectively. In `genesis.json`, the Ethereum account addresses corresponding to these private keys are pre-allocated with a large number of ether, which is convenient for the implementation of B-RAN service in the tests. If you want to modify the corresponding account address, please copy the new "keystore" file to the corresponding directory and replace the account address in `genesis.json`.

After running `sh sim.sh`, there will be several more data files in the current directory.

- `ue_mon.csv`, `ap_mon.csv`, and `miner_mon.csv`: contains the runtime data of user equipment, access point, and miner during the test.
- `size_req.csv`: contains the size of the request data sent by user equipment at each timestamp.

- `actually_serv_num.csv`: contains the number of valid requests serviced by access point at each timestamp.
- `rcv_serv_num.csv`: contains the number of requests that reached the access point at each timestamp.
- `req_start_time.csv`: contains the timestamp of each request sent from the user equipment.
- `serv_start_end_time.csv`: contains each B-RAN service start and end timestamps requested by user equipment served by the access point.

Authors

- **Ruiwei Guo**
- **Yuwei Le**