User Configuration Manual for Research Project: Decentralized Based SDP for IoT Ecosystem in Edge Computing

Prabhav gaur
x22245316@student.ncirl.ie

---

1. Introduction

This manual provides detailed instructions on the required software tools, configurations, and settings needed to successfully replicate the experimental setup of the research project titled "Decentralized Based SDP for IoT Ecosystem in Edge Computing." It assumes that the reader has basic knowledge of the necessary programming languages, blockchain technologies, and IoT frameworks.

---

2. Software Tools and Libraries

2.1. Python Environment

- **Version**: Python 3.8 or later
- **Key Libraries**:
  - `web3`: For interacting with the Ethereum blockchain.
  - `azure-iot-device`: For connecting to Azure IoT Hub.
  - `pandas`: For handling data and logging metrics.
  - `python-dotenv`: For managing environment variables.

2.2. Ethereum Blockchain

- **Node Provider**:
  - Use a public provider like Infura or a locally hosted Ethereum node.
- **Smart Contract Deployment**:
  - Solidity Compiler Version: `^0.8.0`
  - Deployment Platform: Remix IDE or Truffle
  - Network: Ropsten (Testnet) or Mainnet (for production)

2.3. Azure IoT Hub

- **Azure Subscription**: Ensure you have an active Azure account and IoT Hub instance.
- **Device Provisioning**: Devices must be registered in the IoT Hub, and keys must be generated for each device.

2.4. Development Tools

- **IDE**: Visual Studio Code or PyCharm (with Solidity plugin for smart contract development).

3. Configuration Details

3.1. Environment Variables Setup

Create a `.env` file in the root directory of your project with the following content:

```
# Azure IoT Hub Configuration

IOT_HUB_HOSTNAME=your_iothub_hostname_here

DEVICE_ID=your_device_id_here

DEVICE_KEY=your_device_key_here

# Blockchain Configuration

BLOCKCHAIN_URL=your_blockchain_url_here

CONTRACT_ADDRESS=your_contract_address_here

CONTRACT_ABI='[ json ]'

PRIVATE_KEY=your_private_key_here
```

3.2. Blockchain Setup

1. **Smart Contract Deployment:**
    - Compile the provided Solidity contract using Remix IDE or Truffle.
    - Deploy the contract on the Ethereum network using meta mast sign in (Ethereum Testnet).
    - Note down the contract address and ABI.

2. **Web3 Connection:**
    - Use `web3.py` to establish a connection to the Ethereum network via the provided node URL.
    - Ensure your wallet is funded with sufficient ETH for transactions (use faucets for Testnet).

3.3. Azure IoT Hub Configuration

1. **Device Registration:**
    - Register all the IoT devices intended for simulation in your Azure IoT Hub.
    - Obtain the connection string for each device.
2. **IoT Hub Configuration:**
    - Configure the IoT Hub to accept connections from the devices using the connection strings.
    - Set up necessary routes and endpoints for data processing.

3.4. Python Script Configuration

1. Load the `.env` file in your script to configure environment variables.
2. Initialize the IoT Hub client using the `azure-iot-device` library.
3. Set up Web3 with the blockchain URL and the private key for signing transactions.

---

4. Running the Experiment

4.1. Start the IoT Device Simulation

1. **Run the Python script:**

    python IOT.py

    - This will simulate the interaction of registered IoT devices, process data at the edge and fog layers, and log communications on the blockchain.
2. **Monitor the Logs:**
    - The script will generate blockchain logs, communication logs, and metrics logs in CSV format.
    - Review these logs for analysis and validation.

4.2. Managing the Experiment

- Stopping the Simulation: Use `Ctrl+C` to gracefully stop the simulation.
- Log Review: Logs are saved in `metrics_log.csv` and `communication_logs.csv` files for post-experiment analysis.

---

5. Troubleshooting

- **Blockchain Connection Issues:**
  - Ensure your blockchain node (e.g., Infura) is reachable.
  - Verify that the correct `BLOCKCHAIN_URL` is set in the `.env` file.
- **IoT Hub Connectivity:**
  - Double-check the device credentials in the IoT Hub.
  - Confirm that the `DEVICE_ID` and `DEVICE_KEY` match the IoT Hub configuration.
- **Smart Contract Errors:**
  - Ensure that the smart contract ABI and address are correctly referenced in the script.
  - If using Testnet, verify that the wallet has sufficient funds for transaction fees.

---

6. Conclusion

This configuration manual outlines the necessary tools, environment settings, and steps to replicate the experimental setup of the research project. Proper adherence to the configurations provided will ensure that the simulation runs as expected and that the results are consistent with those reported in the research study.