Blockchain-based Sensor Data Integration

Group 5 Topic B
Samir Duvelek
David Kirchsteiger
Luca Moroldo
Konstantin Strümpf

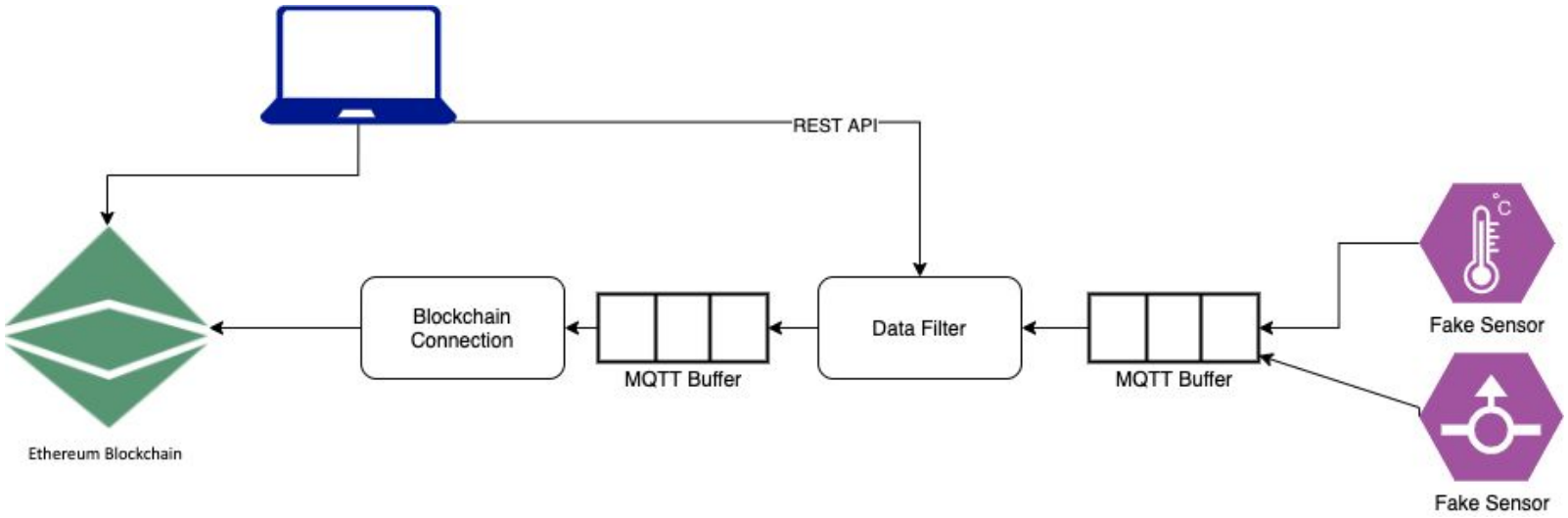
Use Case

- Insurance company
- Two roles (users)
  - Admin
    - **Sets filters** for relevant data points (e.g. wind speed > 150 km/h)
  - Insurance Expert
    - Can see **“extreme” data points** (events)
- Company benefit (blockchain)
  - Data can not be tampered with

DEMO

```
pragma solidity >=0.4.22;

contract SensorData {

    bytes[] public items;

    event Notification(string message, uint index);

    function addDataItem(bytes memory data) public {
        items.push(data);
        emit Notification('New data item!', items.length - 1);
    }

    function getLatestDataItem() public view returns (bytes memory) {
        if(items.length > 0){
            uint idx = items.length - 1;
            return items[idx];
        }
    }

    function getDataItem(uint32 itemId) public view returns (bytes memory){
        return items[itemId];
    }
}
```

Architecture



Tech Stack

- Fake Sensors: **Python + Pandas**
- Message Broker: **Eclipse Mosquitto**
- Data Filter: **Java** and **Spring Boot** Framework
- Blockchain Publisher: **Python + Web3**
- Web UI: **Angular**
- Docker-compose for local orchestration
- Deployed on GCP with docker-compose

Thoughts on Scaling

Towards fail safety and high-availability

- MQTT cluster (e.g. HiveMQ, RabbitMQ)
- MQTT Quality of service “*at most once*” -> “*exactly once*”
- Replicate Data Filter:
  - Filter params could be stored in a shared in-memory database
- Replicate Blockchain Publisher:
  - Independent (i.e. no dynamic params)
- Authenticate Filter Backend

Thank you!

Open for questions