CMPE 281 Cloud Technologies
SmartAgCloud Component Design

Component: Smart Agriculture Node

Xiaosa Yang

4/23/2019

1. Smart Agriculture Node

1.1 Component overview

Objective and Purpose: There are two node types that comprise the smart agriculture node class. One is a sensor node which provide an interface for the sensors to communicate with the cluster node. The other one is the cluster node which provides an interface for the sensor nodes to transmit their data to the edge server. A sensor node will contain one or several agricultural sensors and will relay this data to the assigned cluster node for data storage. A cluster node will connect to an array of sensor nodes and transmit their data to the edge server.

Function Scope and Usage: The sensor node has the ability to collect data from sensor and transfers the data to the cluster node. It also has control functions to allow node creation, delection, as well as configuration of the node and update sensor status. The cluster node connects to the sensor nodes within its range and gather data from the sensor nodes to be transferred to the backend for data storage.

Sensor Node

1. Connect to sensor
2. Get Data from sensor
3. Connect to cluster node
4. Send Data to cluster node

Cluster Node

1. Connect to sensor node
2. Get Data from sensor node
3. Save Data from sensors to local database
4. Send Data to database server from local database

1.2 Component interfaces

Functionalities of Sensor Node:

1. Connect to sensor
    a. Parameters
        i. Sensor
2. Get Data from Sensor
    a. Parameters
        i. Sensor ID

    b. Return
        i. Sensor Data
3. Connect to Cluster Node
    a. Parameters
        i. Cluster Node ID
4. Send Data to Cluster Node
    a. Parameters
        i. Cluster Node ID
        ii. Sensor data

Functionalities of Cluster Node:

1. Connect to sensor node
    a. Parameters
        i. Sensor ID
2. Get Data from sensor node
    a. Parameters
        i. Sensor node ID
    b. Return
        i. Sensor data (includes sensor id, timestamp, data value)
3. Save Data from sensors to local database
    a. Parameter
        i. Sensor data
4. Send Data to database server from local database
    a. Parameter
        i. Sensor data

Note: Sensor data includes sensor id, timestamp, data value.

API Design for node communication:

Sensor and sensor node will communicate through MQTT protocols, once sensor node reads data from sensor, it sends a message to the cluster node it belongs. The cluster node receives the message that contains sensor data, and connects to its local database, it then stores the data at the local database.

API Design for Cluster Node send sensor data to backend server/database:

When cluster node is ready to send sensor data to backend database, it sends the backend server a request to access the database, once the connection is established, it retrieves sensor data from local database and pass the sensor data as objects to the backend server.

Local Database API Design:

Each cluster node will have its own local database for temporary sensor data storage. Since the sensors are constantly collecting data, its assumed there will be huge amount of sensor data per cluster node. For now, we will use mongoDB for the local database, where the updated data will be appended to the database as long as the sensor status is on.

When sending the sensor data in each cluster node to the backend database server, the data values will be passed in in JSON format through the database API call, upon checking the status of sensors.

Data example in JSON format:

```
{

   "_id" : ObjectId("1bd9ceaabf81cc88a9f13777"),
   "sensor_id" : "1970084hsj038ba3028ab9303038ab1",
   "value" : [
      {
         "date" : ISODate("2013-01-09T06:04:571Z"),
         "temp" : 44.0
      },
  ]
}
```

```
mycol
> db.mycol.find().pretty()
{
        "_id" : ObjectId("5cbf0dfb08375f837e4cba1e"),
        "Sensor ID" : "19700eb9568a4b75ac4c436a0b1e3461",
        "Sensor Type" : "temperature",
        "Sensor Status" : "on",
        "Timestamp" : "2019-04-23T06:07:07.835021",
        "Sensor Value" : 69
}
{
        "_id" : ObjectId("5cbf0dfb08375f837e4cba1f"),
        "Sensor ID" : "a3cd19e3a61c42d2a8cb21b3a0b4a2d2",
        "Sensor Type" : "temperature",
        "Sensor Status" : "on",
        "Timestamp" : "2019-04-23T06:07:07.885514",
        "Sensor Value" : 79
}
```

Figure 1.2.1: Example of sensor objects in MongoDB database

1.3 Component functions and behaviors

Program class templates with some attributes and functions:

```
Class BaseSensor
{
        Private string  sensorID;
        Private string Type;
        Private string Status; //On, off, activated, deactivated, maintenance
        Public void setStatus();
        Public string GetData();
        Private void CollectData();

Class SunlightSensor: Class BaseSensor
{
        Public SunlightSensor();
        Public void setStatus();
        Public string GetData();
        Private void CollectData();
}

Class WaterSensor: Class BaseSensor
{
        Public WaterSensor();
        Public void setStatus();
        Public string GetData();
        Private void CollectData();
}

Public Class BaseNode
{
        Private string NodeID;
        Private String latitude;
        Private String longitude;
        Public void TurnOn();
        Public void TurnOff();
        Public void Activate();
        Public void Deactivate();
        Public void SetMaintenanceMode(bool value);

}
```

```
Class ClusterNode: Class BaseNode
{
        Public string getNodeID();
        Public void setSensorNode();
        Public string GetDataFromAllSensorNodes();
        Public void RemoveSensorNode(string nodeID);
        Public void AddNode(SensorNode node);
}

Class SensorNode: Class BaseNode
{
        Public string getNodeID();
        Public void setSensorNode();
}
```

This next section demonstrates a state diagram for the different states that a sensor can be in, a sensor node and cluster node share the similar state behaviors like a sensor.

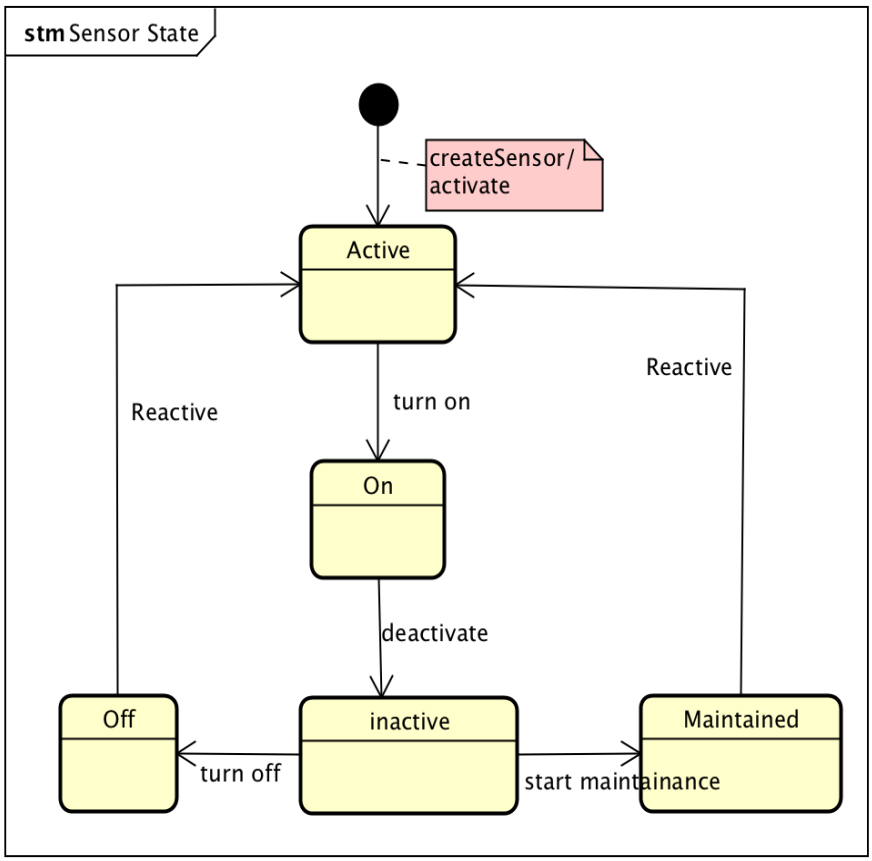

Figure 1.3.1: State Diagram for sensors

1.4 Component business logics

Business rules to be considered:

- Determine max sensor number per sensor node
- Determine max sensor node per cluster node
- Determine max cluster node per farm
- Sensor installation fee waived when subscribe to our services
- Loyal service subscribers (Maybe more than 3 years) receive 15% discount on data storage in our backend server
- Free local database storage

1.5 Component graphic user interface design

This is a user case demonstration for the API interactions when IOT manager(actor) wants to establish a new cluster node, a new sensor node, and a new sensor.

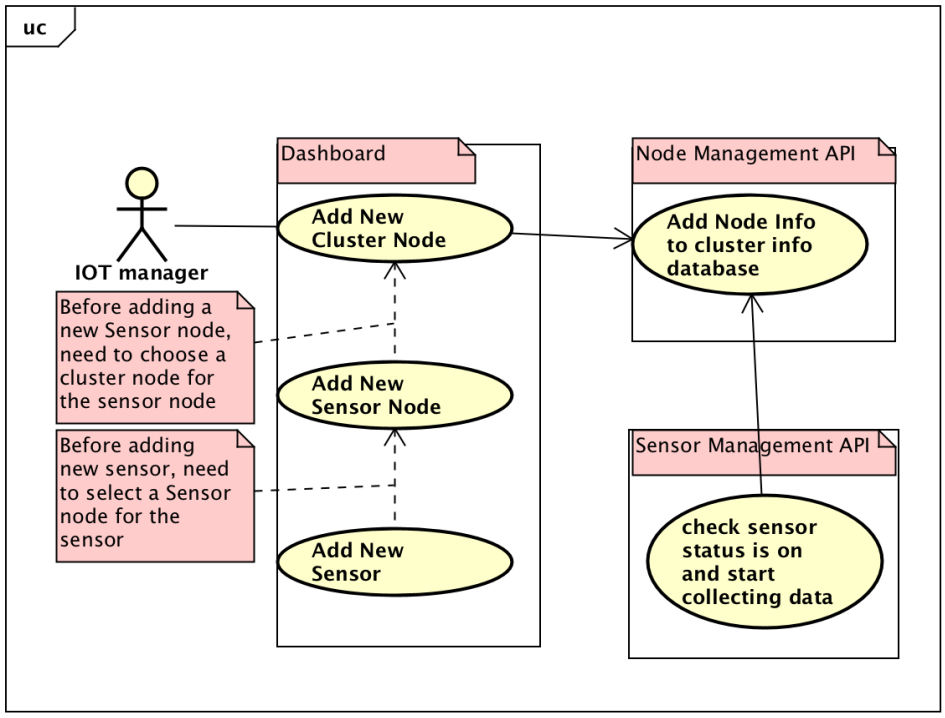

Figure 1.5.1 User Case Demonstration

Below are screenshots of the GUI design:

Welcome to Yang's Farm!

Add Cluster Node
Add Sensor Node
Add Sensor
Show Sensors

Add a Cluster node

NetworkID:

ClusterID:

Latitude:

Longitude:

Add

Add a Sensor node

ClusterID:

SensorNodeID:

Latitude:

Longitude:

Add

Add Sensor

SensorNodeID:

Sensor Type:

Number of sensors needed:

Add

Sensor ID	Sensor Type	Sensor Status
68d90be9f6c045198a5759631c65dd84	water	on
1a0469bf1a374b8797f9df1ba4589d72	water	on
a5b5c76073874ab4bc02a1bc51891b51	water	off
c78386416c7a4ce493e002115c36acf2	temperature	on
963d4b26ecde4deeac73c7caa748edc8	temperature	off
d60bb75929974c5d89fea6950d84f30d	temperature	on
98634dd4d1374c739a58e12bee4c6336	temperature	on
ae9f415a928a4e3ea567c7e7ab876b9f	temperature	off
2b75fb290c0d47e19bb671d58c9c29c7	water	on
1d8deb9ca1824883ad9e535a33e0045e	water	off
1cf3e2b049cf4397a46ab0d99cc25a92	water	on
74a167aa9b504a468efca683cc92591e	water	on