Internet of Things Apps Platform

A Project Design Document
Version-1.0

For the module
Application Tier

Submitted by

Team 1 :

Sourabh Dhanotia	201405605
Lokesh Walase	201405597
Pankaj Shipte	201405614

Of
Group Number - 5

Under the guidance of
Mr. Ramesh Loganathan

For the course
Internals of Application Server
IIIT, Hyderabad

23rd March, 2015

Contents

List of Figures

Chapter 1

Introduction to the Project

The next wave in the era of computing will be outside the realm of the traditional desktop. In the Internet of Things (IoT) paradigm, many of the objects that surround us will be on the network in one form or another. Thus environment around us will be an invisible mesh of information and communication systems constantly exchanging data, talking to each other. This results in the generation of enormous amounts of data.

This data have to be dealt with at three different levels -

- Storage - Store the relevant data in required format in a database.
- Processing - Process/filter the data as per systems' requirements.
- Delivery - Deliver the required data to the end-user in a specific format.

IoT is heterogeneous in nature as it consists of a variety of hardware devices, interacting with software-systems, each system having different communication protocols. Hence creating an app for a new IoT system is not an easy task. To ease this problem of heterogeneity, we propose to create

- IoT Apps Platform. The primary objective of the IoT Apps Platform is to provide a generic (protocol and hardware independent) platform that will make the data from IoT available to the mobile app. It will majorly speedup the process of creating different mobile apps since the intrinsic details of handling the IoT data are taken care of, thanks to the platform. Thus our platform will seamlessly handle the storage, processing and delivery of the data that is generated by the Sensors and make it available to the app.

Chapter 2

Key Functionalities

The following are the key functionalities of the Application Tier module :

- Listen to the requests from end applications
- Convert these requests into the format of filter server and forward the same to it
- Receive processed data from Filter server
- Process the data received from filter server and extract meaningful information for the app and send the result to it

Chapter 3

Test Cases

In order to test the module, following test cases will be used -

- In guided Parking system, The user will connect with the platform and ask for a free slot. If 2 users requests at the same time, the platform should not assign the same parking slot to them.
- In wildlife sanctuary, when an animal comes in the area of the sensor twice, He should be counted only once.
- In Gas leakage detection system, the sensors should not report the normal gas pressures as a leakage.
- In vehicle type temperature monitoring, The application tier should notify the application irrespective of the active or inactive status (emergency situation)
- In smart garden, it should be monitored properly as which plant needs water. (Do not overwater a plant)

Chapter 4

Overview of Design

4.1 Block Diagram

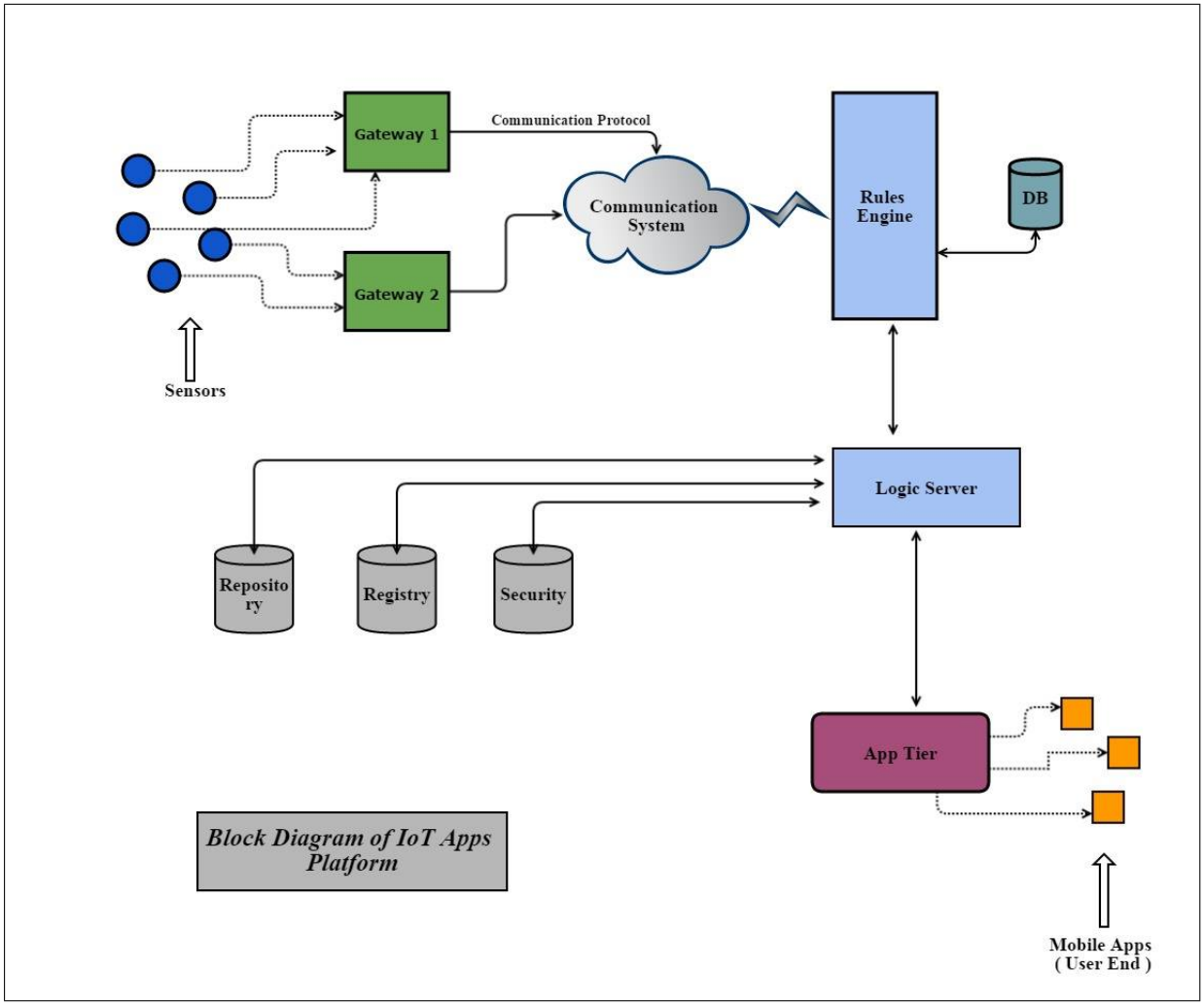

Figure 4.1: *Block Diagram of IoT Apps Platform*

4.2 Functionalities

1. Listen to the requests from end applications
2. Convert these requests into the format of filter server and forward the same to it
3. Receive processed data from Filter server
4. Process the data received from filter server and extract meaningful information for the app and send the result to it

4.3 Basic Work Flow

1. The applications communicate to this module through some communication medium like BLE or Wi-fi.
2. The applications gets registered with the platform by sending a registration request to the application tier
3. The registered applications then send requests through the same media used for registering with the platform
4. The app tier performs some processing on the request from the applications and forwards it to (in proper format) to the filter server.

Chapter 5

Low Level Design

5.1 Brief Overview of the Application Tier Services

1. Application tier will start its services as soon as any application requests for the same.

2. Applications demand services from application tier dynamically.

3. Application tier will run as long as server is running. If for any reason like maintenance server goes down, the application tier will go along with it.

4. Application tier has a set of supported services.

5. Application developer need to use services only from the domain of Application tier.

6. Application also needs to follow the same format of service as provided by application tier.

5.2 Life Cycle of the Module

1. **Logic Server** acts as a bridge between the application tier and filter server and registry server.

2. **Repository Server** comes in to the picture if we want to add new device(either sensor or gateway) in the platform.

3. **Registry Server** is a dynamic lookup table, holding the data of the current active devices.

4. **Security Server** handles the authentication mechanism for the platform.

5.3 List the sub modules and block diagram

- Logic Server
- Repository Server
- Registry Server
- Security Server

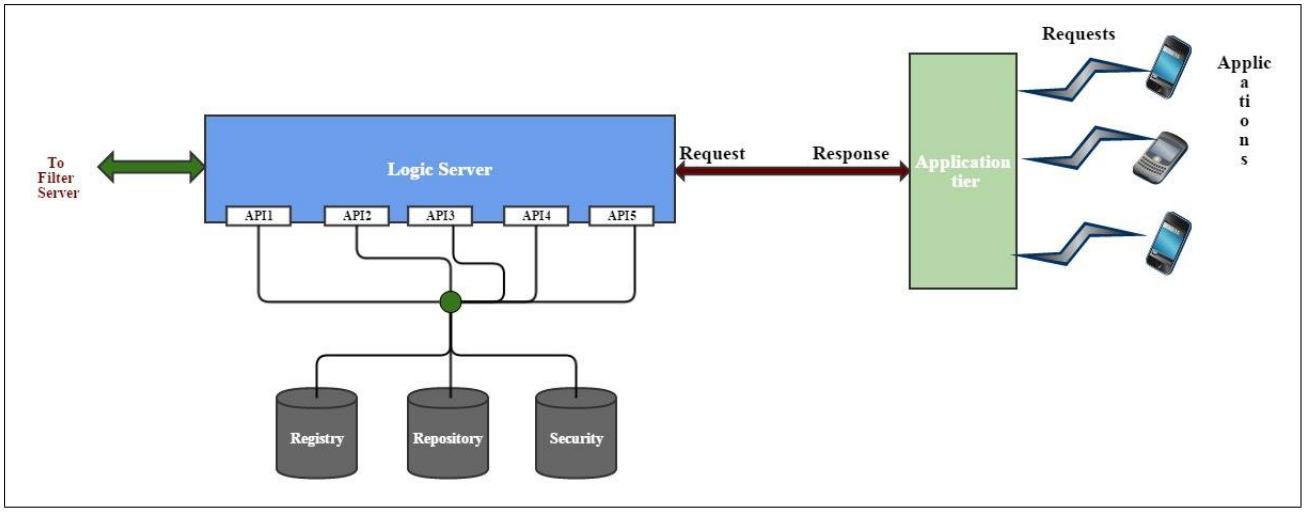

Figure 5.1: *Block Diagram of Sub Modules*

5.4 Logic Server APIs :

1. **Boolean createFilter(sensor id , sensor_type , start_time , interval , end_time , repetition , action_required/message)**

   Return type : True (created successfully) / False(error occurred)

   Description : Create a filter on user request

   - Sensorid : id of the sensor
   - Start_time : beginning of time from when filter is to be applied
   - Interval : duration of the filter
   - Repetition: how often the filter should be applied (daily/weekly)
   - Action_required/message : The action to be performed when this particular condition is met (Signal / Log)

2. **Boolean removeFilter(sensor id , sensor_type , filterid)**

   Return type : True (removed successfully) / False(error occurred)

   Description : Remove a filter on user request

- Sensorid : id of the sensor
- Start_time : beginning of time from when filter is to be applied
- Filter_id : id of the filter to be removed

3. **Boolean checkSensorStatus(sensor id ,sensor_type)**

   Return type : True (sensor is alive) / False

   Description : Check whether Sensor is alive or not

   - Sensorid : Id of the sensor
   - Sensor_type : Type of the sensor

4. **Boolean AuthenticateRequest (Device identification)**

   Return type : True (device is registered) / False

   Description : Authenticate request (whether device is registered with platform)

   - Device_Identification : IMEI no (for mobiles) / MAC addresses
   - Return type : true (device is registered) / False

5. **Data[] CallBackAPI(sensor id , sensor_type , start_time , interval , end_time , Data_Constraint)**

   Description : Retrieve the data according to the time and data constraints specified by the request. Query the database if time is in past

   - Sensorid : id of the sensor
   - Start_time : beginning of time from when filter is to be applied

- Interval : duration of the filter
- End_time: time of End of request
- Data_Constraint: Retrieve data when constraints met

5.5 Brief overview of each sub module

1. **Request Handler :** Listen to the requests from end applications
2. **Request Validator :** Convert these requests into the format of filter server and forward the same(At filter server)
3. **Request Processor :** Store the processed data, source info and other required details into the database(At application tier)
4. **Request Dispatcher :** Listen to the requests from end applications

5.6 Key Functionalities of sub modules

1. **Repository Server :**
   - Registory server is accessed only at boot time
   - It stores list of all the devices which are connected to system.
   - Repository Server would fetch all the data from the database on its startup every time. It would contain the details of each and every device that exists in the system.
   - Repository server would be updated by the administrator whenever a new device is added to the system

2. **Registory Server:**

- Repository Server stores health of each device connected to system
- It also helps system to determine which sensor was down for what duration.
- Registry Server stores the dynamic details of the current devices present in the system
- It keeps itself updated with current status of each device.
- It also maintains transaction log.

3. **Security Server:**

- Security server is used to validate the requests coming from applications.
- It forwards only those requests to logic server and filter server which are coming from registered application.

4. **Logic Server:**

- Logic server interacts with applications designed on platform to receive their requests and also responds with appropriate data and callback.
- Logic server is responsible for the providing interface to external applications.
- It converts user queries to filter server understandable format

5.7 Interactions between sub-modules

1. Repository server interact with database to verify list of all connected device.

2. Repository server communicated with database whenever a device is added updated or deleted

3. Registory server will interact with the logic server to store the log of transactions taking place into the system. Security server communicates either logic server to forward only those requests which are authenticated

5.8 Interaction with other modules

1. The registry server will contain the runtime details of the devices mentioned in repository server. It will interact with sensors via gateways to get the current status of each device which will be used for health monitoring of sensors

2. Registory server will interact with the filter server to store the log of transactions taking place into the system.

3. Repository server will communicate with Gateways at boot time to inform them about all sensors configured in system.

4. Logic server communicates with filter server to forward the request coming from applications and also accepts callbacks in turn as a response to these requests.