Internet of Things

Design Outline

Group 2

**Introduction**

IOT based sensor monitoring system will serve many purposes. Sensors can be of different types, whether it be temperature sensors, humidity sensors, pressure sensors etc. Since all the different types of sensors have different type of data they transmit, it is very difficult to make use of the data in its raw format. Hence, we provide a platform that allows application developers to build and deploy the apps that make use of data that we receive from the sensors and provide essential information to the users. For e.g. temperature sensors can be used for central ac system. The temperature from different sensors can be read and analysed to find out whether the temperature in the corresponding area should be increased or decreased.

The platform abstracts the middleware of the system and the APIs are exposed to the developers. They don’t need to worry about the data format (as data from different sensors will have different meanings), the communication protocol etc., as all these are taken care by the platform. Rather, they just need to focus on what app they want to make and how will the app deliver what it promises to the target audience using the api’s provided. An efficient level of abstraction is implemented so the user only sees what’s important to him and all the other secondary things are hidden. Since, the platform caters to heterogeneous class of sensors, the data from the sensors are read and classified into different categories based on the requirement.

**Test Cases:**

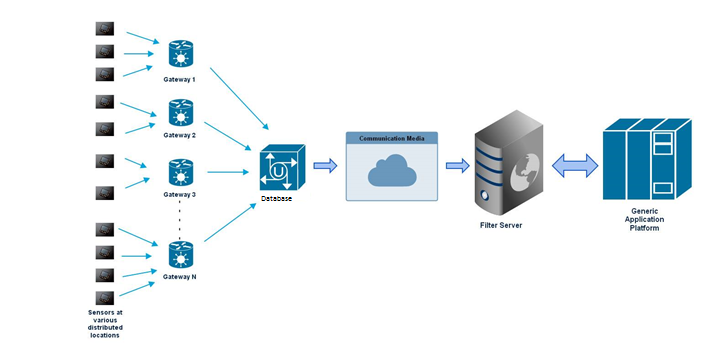
1. The sensor is properly sending the required data or not i.e. whether the sensor is properly sensing or not.
2. Sensor is located at the correct location and must be active.
3. Packets send by sensor is lost or received at the gateway (Must be acknowledged).
4. Filter server shall be filtering the data according to the type of applications.
5. Data sent to the different applications should be same for same type of sensors.
6. Developer data and authentication should be made confidential.
7. In case the application is not active, the data must be sent to/from the central database.
8. Malicious applications must be detected by the system using the rules engine and no data must be sent to such an application.
9. The central database must store all input data from the gateway irrespective of the activeness of an application.
10. Rules engine should properly store all the details of the application.
11. Failure of Master Gateway results in the failure of the system and thus specific measures should be taken to ensure 100% uptime.
12. When a sensor is registered but database storage fails due to connection failure or some other problem, then sensor will try to register with the system again and again thus degrading the performance.
13. When the sensor registers, it should be ensured that every new registered sensor must have some unique id that the gateway provides.

**Integration Testing**

When all the modules are integrated, things to be checked are:

1. Verify that the sensors send data continuously to the gateway and the gateway properly receives it. Also verify, that the data it sends is of the correct type.
2. Since there is a mesh of gateways, the routing table should be managed to transfer the data to the filter server. Verify the correctness of the routing table.
3. When a sensor registers, its data is entered into the sensor repository. Verify, that the database connection is properly working and sensor information is correctly stored in the database.
4. All the data received from the sensor needs to be archived. Hence, verify the storage of data. Also avoid storing malicious data in the database.
5. Gateway to filter server connection should be properly working, otherwise the data will be lost.
6. Filter server must properly communicate with the gateway as well as the active application database so that, when it needs to forward data to application, it can consult the database and find out whether the application requesting data is active or not.

**Solution Design Consideration**



**Environment**

OS: It is independent of the OS as the platform is developed in JAVA which is platform independent. Both Windows and Linux would work fine.

The Master Gateway will be based on javascript on an Android device while the Filter Server will require a Windows/Linux machine to be run on. The Master Gateway will interact with the Filter Server sending the sensor data as it receives.

**Technologies**

**Node.js** - It is a platform built on **Chrome's JavaScript runtime** for easily building fast, scalable server-side and networking applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices. Node.js will be to produce dynamic data through the sensors. It is used for communication throughout our platform.

**MongoDB:** It is an open-source document database, and leading NoSQL database. It avoids traditional table-based relational database structure in favour of JSON-like documents with dynamic schemas . Since our application platform uses MongoDB as the backend database, we need MongoDB to be installed on the development system.

**Java:** Since the application are basically built on Android Platform, we require Java to be present in the system.

**Android SDK:** It is software development kit that enables developers to create applications for the Android platform. We require the SDK to be present in the system so as to be able to develop applications that benefit from the platform.

**Android Device**: Android device with javascript support will function as our gateway which receives data from the sensors and sends it to the filter server when required. It is a kind of simulated push.

**Approach for Device Gateways**

All the sensors will be connected to some of the gateway within the mesh of gateways. The individual gateways will send this data to Master Gateway which will archive the sensor data in the database as well as send it to filter server for further processing.

**Device Types**

The devices may be of any kind that are capable of working over Bluetooth technology. These send data to another Android device running javascript acting as gateway for the sensor data.

**Devices Registry and Repository**

All the devices that are used within the system must be registered with the platform and their information must be stored in the database, this will help to determine whether the data received at the gateway is coming from a valid device as well as help determine the identification of the sensor i.e. its location, id, etc. This data will be stored in Repository database. The registry server will keep track of which sensors are currently online and which are offline.

**Approach to get device information**

Logic Server will use the Query API to interact with the registry server to get the information about the devices that are currently active and the type of devices. After getting the list of devices, it will then use the COMMAND API on filter server to ask for its data.

**Communication Overview**

First of all, the data sensed by sensors are sent to the respective gateways which collect the information. This information is dumped in a database for archival purposes. The gateway identifies the type of sensor and data by consulting the Sensor Type Repository. Since, raw data cannot be sent across the communication medium, gateway encloses the data, type, location and other properties concerning the sensor into a protocol header which is then transferred across the communication medium using RESTful web services on TCP/IP protocol. The filter server then filters out only those sensor data that are being requested by the apps that are built using the APIs provided by the platform. The Sensor type repository uses MongoDB as the database and the Gateways are instances running on Android with Javascript support system to simulate the functionality of an actual gateway.

**Filter Server**

RESTful services are used for interaction between Gateways and filter server. Filter server is created using Node.js technology. Since filter server receives data of all the types of sensors that are installed in the area, it must filter out only those that are requested by the application, i.e. an application requesting temperature information must not be shown humidity information and vice versa. For this, the filter server consults the sensor type repository and then sends only valid data to the application. Also, the filter server validates the source of generation of data. This is primarily required from the security standpoint otherwise malicious data might be shown to the user.

**Logic server (aPaaS)**

**Application platform as a service (aPaaS)** is a cloud service that offers development and deployment environments for application services. The Logic server requests the data from filter server after querying the registry server for active devices based on type and location. It uses either command API or callback API to register a request which is fulfilled after the filter server has required data to send.

**File Formats**

JSON format is used to transfer data within the network. The same type of format is used to store data within the MongoDB database thus, it becomes easy to retrieve and work with the data.

**User’ View (Mobile UI)**

The developer can build an android application for user’s mobile device. The app will use the APIs provided by the platform to retrieve data according to the requirement of the user.

**User’s view of the system**

The application for each type of data will be an android application that a user can install on his/her mobile device and view the data. For e.g. an application showing temperature information of a particular location will request temperature data from the filter server and show it to the user in the desired manner.

**Configuration Files**

A configuration file is required to maintain the IP Address and Port information of the gateway and the filter server. Also information regarding connection to the database must also be maintained and its buffering properties must also be maintained.

**Structure of Files**

Since, all the data communicated within the platform is done using the JSON format, it consists of key-value pairs. The same data format is used in MongoDB for storing data in the database.

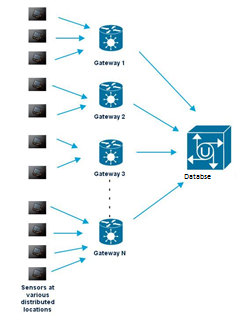
**Deployment**

There are 3 individual components to the platform: sensors, gateways and the filter server. Sensors will independently run and continuously send data to the gateway it is connected to. The gateway needs to run on a different machine (in this case, a Raspberry Pi system). Multiple instances will be run so as to create a mesh of gateways. The filter server also is an independent entity and hence needs to be run separately. These are run on individual servers so that failure of one does not lead to failure of the other things.

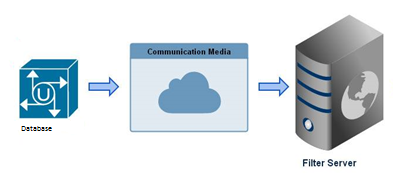
All the components will be started/stopped individually. The sensors will act as an entity that continuously sends the sensed data, the master gateway does both of the work, sending and receiving and same happens with the filter server.

**Interactions and Interfaces**

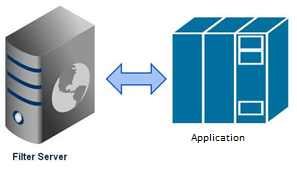
**Sensor-Gateway Interaction -** Socket programming is used for interaction between sensor and gateway. There is a collection of gateways which are simulated using an Android with Javascript support system. The sensors send their sensed data to the gateways they are connected and all these gateways then send the collected data to the database for archival as well as to the filter server forward in the network based on the request.

****

**Gateway-Filter Server Interaction** - RESTful services are used for interaction between Gateways and filter server. The Gateway creates a protocol header containing the required information to transfer across the communication medium. The communication medium is the Internet and the data is sent over the TCP/IP protocol. The message header containing information like id, type, location etc. is transferred to filter server for further processing.

****

**Filter Server-Application-** The application through logic server first queries whether the sensors it demands is online or not. The query is sent to the registry server which consults the registry database and then sends out a list of active servers meeting the query criteria. After getting the list of required devices, the logic server register a callback for the data by using either the callback api or the command api. The filter server when has the required data sends it to the logic server for displaying it to the application.



**Persistence**

1. Sensor/Device information like location, id, type must be stored in the database so that new sensors can be added and old ones be removed.
2. Active Application Information – The applications that are registered to use the platforms services must be stored in a database with their current state. When the filter server transfers data to the application, it first consults the database whether the application is active or not. If the application is found inactive then the data is discarded otherwise it is forwarded to the application.
3. Sensor Data – All the sensor data must be archived for future references. The Master Gateway after it receives the data from the sensor, also stores the data in the database as well forwards the same information to the filter server.