**IIIT\_H/ Spring 2015 : Group 4**

**Internals of Application Servers**

**Project: Design Document**

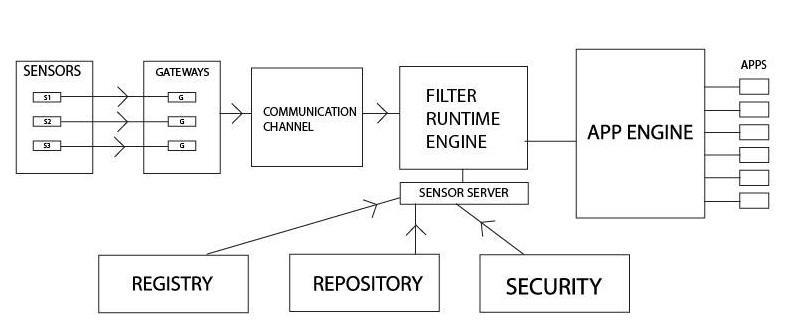
**1. Intro to the Project :**

This project aims to build a sensor network based on IoT platform and develop an interactive android application that receives multiple type of data from sensor network. This application shall allow users to view information such as temperature, traffic or sensor type data, generate reports and control related devices from their android device.

*Functional overview of Team’s module :*

Following are overall modules of this project

1. Sensors, Gateways, and communication.
2. Filter server and App engine
3. Registries, Repository and client apps



**2. Test cases :**

2.1Test Scenarios:

*1. Normal operation scenario:*

Input : Sensor data from sensor and corresponding objects at each module.

Expected output : Each value should be checked against threshold at filter server and alert should be generated for values above threshold temperature.

False alarm should not be generated at values below threshold temperature.

*2. No sensor available Scenario:*

Input : heartbeats of each sensor at gateway:

Expected output: Platform should continuously check heartbeats and if no sensor is available, gateway should wait for appropriate amount of time and then notify system administrator.

*3. Corrupted Sensor data scenario:*

Input : Data received at server is corrupted.

Output : Server should employ a CRC check and if found corrupted data should be discarded.

*4. Faulty Sensor Data:*

Input : Data received at server is out of feasible range.

Expected Output : Range values should be checked at server and appropriate action should be taken at server level.

*5. Link failure test scenario:*

Input : Try breaking links between various modules.

Expected output : System should generate appropriate error messages at specific link failure. Rest of the system should behave accurately.

*6. Performance Test Scenario:*

Input : Normal Load : average requests to the server

Heavy Load : large number of requests to the server

Expected Output : Server should handle load correctly and should not get overwhelmed by heavy load.

*7. Security Test Scenario:*

Input : Malicious sensor data and tempered packets at server

* Authentication test case : +ve test case. Trying with correct credentials.
* Authentication test case : -ve test case. Trying with incorrect credentials.
* Check for memory leaks in the code.

Expected Output: Server should detect the temeping and appropriate action should be taken like discarding the packet.

*8. Efficiency Test Scenario:*

Input : Measure the time interval for interaction in each module.

Expected Output: Each time interval should fall in feasible limits.

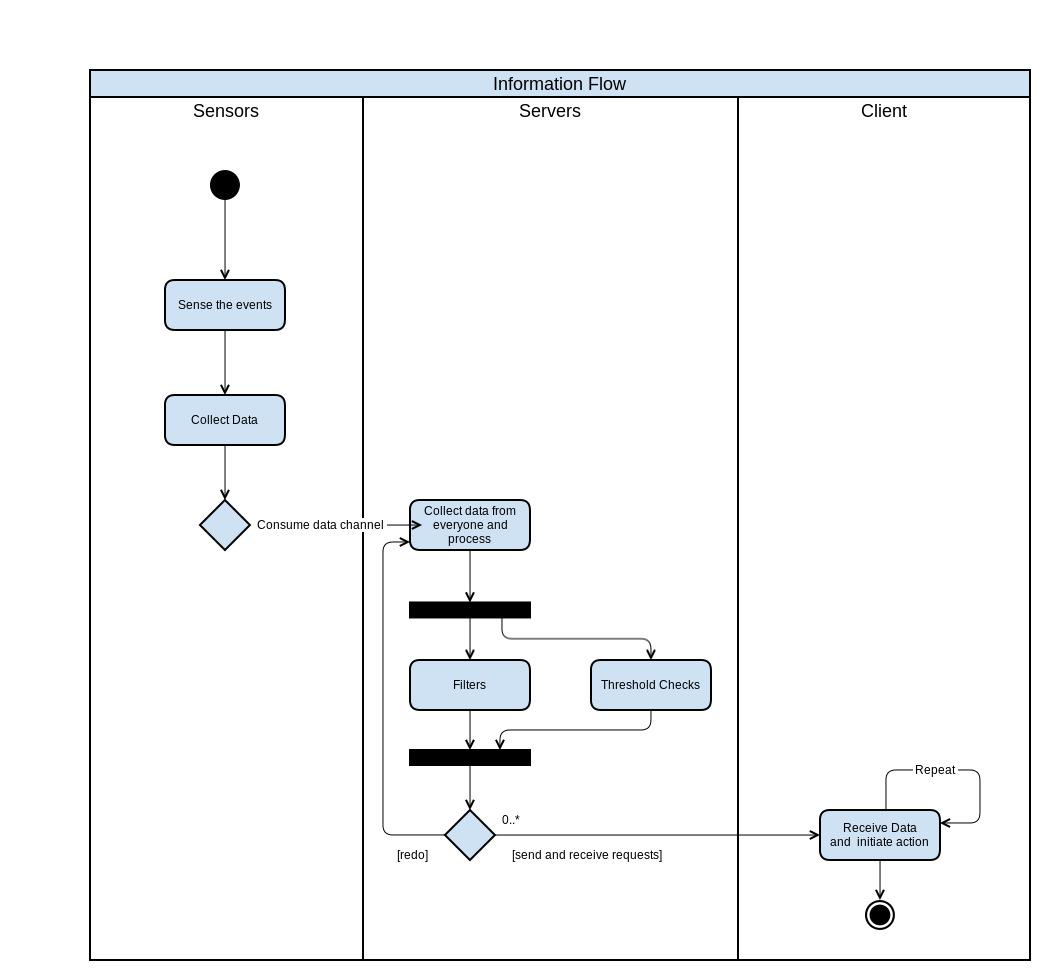
2.2 Overall project test cases :

*Integration Test Cases:*

* Sensors should detect accurate values and report them to gateways.
* Gateways should be able to accept data sent by sensors and communicate to server.
* Server should be able to receive the information and process them as per the requirement.
* Load test : Server should not be overwhelmed by any kind of load on the system.
* Client Application should be able to communicate to server as per its own logic and trigger events when the specific condition is met.
* Negative Test case : If server is down, client application should not trigger the event based on cached value.
* Performance Test : Test system performance under different kind number of users, no of requests and network traffic and generate a report.

**3. Solution design considerations :**

3.1 Design big picture



3.2 Environment to be used

* Android,Linux, Noobs, Apache server
* Simulate sensors, Raspberry pi/simulator, filter server
* Remote procedure calls, data interactions

3.3 Technologies to be used

* Node.JS, C++/Java, MongoDb
* Node.js is good for writing server side script and it manages 2-way communication very well.
* MongoDb does not have overhead relational database so it is fast. Also it uses noSQL which ideal for our platform.
* Java is useful as it is platform independent and simple. It can also be used for coomunication.
* Node.js will be used for writing filter server and MongoDb will for database interactions.

3.4 Approach for Device gateways

- We will be using raspberry pi for initial gateway.

- We will use computers as to simulate remaining gateways.

3.5 Devices type/interfaces and information structure

- *Sensors*

*Types:* Temperature, humidity, camera, smoke, water level, seismic , traffic

Interfaces with gateways

I*nformation structure:* sensor id, sensed data

- *Raspberry pi*

*Type:* Gateway

Interfaces with sensors and filter sever

*Information structure:* <IP, port,<type, sensor id, location, sensed data, timestamp, checksum> >

- *Filter server*

*Type:* Web server

Interfaces with gateways and App engine

*Information structure:*

<type, sensor id, location, sensed data, timestamp, checksum>

<App engine location, registered app type>

3.6 Devices registry & repository

*Registries:*Registry contains the sensor information. Sensor id, sensor type and installation timestamp.

*Repository:*Repository contains data related to sensors.

3.7 Approach to get device information (event streams)

Raspberry pi will detect any new sensors in its vicinity and will keep track of live sensors using heartbeats from sensors. When a new sensor boots up , it sends its id and type information to nearest gateway. Gateway in turn will forward this information to filter server and filter server will check the registry. If the sensor information is not present in the registry , it will create a new entry.

3.8 Device info proxy server

Registry is a database which is used to store the sensor information such as sensor id, sensor type and sensor installation timestamp. Registry will be helpful and capable of differentiating between the various sensors and their data. It can be used as device info proxy server.

3.9 Communication overview

Communication will be done using Sockets , RMI or REST API depending upon appropriate resource availability.

3.10 Filter Server

All apps register with the filter server stating their requirements. When the filter server gets data from the sensors it sends only the relevant data, that is, only the data required by the app.

3.11 Logic server

Filter server will also work as logic server.

3.12 Interactions between modules

* + Sensors to Gateway : Sensors send temperature data along with sensor id to the connected gateway over wireless channel. Gateways listen on a specific port for incoming data and provide an interface for sensors to communicate. This interaction will be implemented using java sockets. One sensor will be connected to only one gateway.
  + Inter-Gateway Routing : Gateways are used to route temperature data collected by sensors to filter server. Multiple gateways can be interconnected and can follow a routing protocol so that if any link between gateways goes down, data can still be routed to filter server using some other path. They will recognize other gateway nodes within range, set up simple routing tables of adjacencies, and discover likely paths to the appropriate integrators/ filter server.
  + Gateway to Filter server : Gateways aggregate, prune and bundle the data, then encapsulate in IP for delivery to the filter server as a small data “stream” . This data packet is sent to the filter server in real time over the TCP/ UDP communication channel. Filter server listens on the specific port for incoming packet.
  + Filter server to App engine : Filter server provide interfaces to access different kind of sensory data. App engine interacts with Filter server by calling the registered service method through RPC at regular intervals or whenever needed

3.13 Wire and file formats

* + *Sensors to Gateway:* <sensor id, sensed data>
  + *Gateways to Filter server:* <IP, port,<type, sensor id, location, sensed data, timestamp, checksum> >

3.14 User’s view (end UI on mobile)

There will be an Android app which will provide following features:

List all services

Sensor Data

Thresholds

Locations Selection

Alerts

**4. User’s view of system :**

***I. Applications for Sensors:***

*1. Weather report and prediction:*

The heat of the atmosphere is largely influenced by the sun and the degree to which it warms the air, land and oceans. The land and water retain heat and continue to heat the atmosphere after the sun disappears. The heat creates warm front of air. The warm air is pushed upward at a sharp angle, causing moisture to condense rapidly. Heavy precipitation is often the result. Warm Fronts occur when a mass of warm air passes over a mass of cold air at a moderate angle. So using these temperature sensors located at different strata of atmosphere will observe the exchanges between cold front and warm front which can be used to predict the weather and forecast rain.

*2. Implementing government actions:*

When the temperature of the air goes beyond 47-50 centigrades, Government takes excessive measures to keep the temperatures in check like pouring water on ground to cool it down. The sprinklers used can be controlled by android app which will use the services of this system.

*3. Critical temperature alert (Heat/ Fire detector):*

If the temperature data sent by sensors have sudden change in their reading, the app will sent out an warning message to the user and trigger the alarm. The app can automatically make call to fire department.

*4. Sanitation:*

Detecting blockage in drain pipes and exact location of blockage. Providing coordinates of blockage to the application user. The application will automatically detect the blockage and provide location to the user.

*5. Riots detection:*

Detecting heavy crowd in city and classifying them as rioting crowd or not on the basis of image processing and speech processing. The application will continuously sense and process image and speech to detect any sensitive activity.

*6. Fire Detection in City:*

The various sensors will be located in city such as heat sensors along with cameras, which will collect the data. Heat sensors will provide a value if this goes beyond a threshold, we can process image related to the particular sensor, and verify if it is false alarm. Sensors will also provide location of the fire.

*7. Detecting Natural Disasters:*

Predicting and detecting the natural calamities. On the basis of various current environmental factors sensors will collect values for these factors causing natural calamities and if threshold is crossed for these factors the application will give warning to the user.

*8. Damage analysis of building:*

Using this application, sensors will provide the status of the building. If the earthquake or any disaster cause any damage to building, end sensors located in the building will provide possible damage to the building.

*9. Class room attendance:*

The sensors will be located in classroom or any area confined under observation. Sensors will provide raw data such as image of the area under observation. Using face recognition we can provide information to the user.

*10. Flood detection:*

The sensors will be located in the dam or a river which will provide water level information on periodic basis. By processing this information if level increases beyond a threshold we can notify the end user with possible flood and damage. We can also use this data to determine the duration of water supply in city.

*11. Infrastructure decision based on traffic sensor data:*

Collecting traffic data and analysis is used make many infrastructure related decisions like where to build a bridge for pedestrians, or where to build a flyover or where to mark a zebra crossing, which road should be made one way for better traffic routes.

***II. Required Config Files:***

1. Config File in the sensors to set its parameters

2. Config File in the Gateways that will set parameters to enable REST-ful services

3. Config File in Filter server that will determine what data to filter and what to send

to the application server

4. Config File in the Application server for deployment of the Apps

**Analytic Rules** are defined in the config file of the filter server.

**Structure of the File** will be in XML format.

**III. *Deployment and setup:***

Apps will be deployed on the App Engine. It will communicate with apps and Filter server with the help of REST-ful services. Analytic rules are defined at filter server. When It gets the data above threshold then it will notify the application port at app engine.User also need to setup config file at app engine, filter server and gateways.

App Engine provides an interface that helps to deactivate it. After deactivation, services related to the application is not served by app engine. User

sends reactivation request then app engine serves app requests.

**IV. *User interactions:***

The user will write an application for the platform and deploy it on App server.

He can configure the sensor and Gateway based on his requirement. He has also configure the filter server to control what data is sent to the App.

**5. Key Data structures :**

Details of deice types :

Sensor type, sensor id stored at registry.

Wire formats

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sensor id | Sensor Location | Sensor data value | Timestamp | Checksum |

Persisted data

Registry database

Rules Database

APIs- interaction data objects

App Logic definitions

User’s UI (mobile)

List all services

- Sensor Data

- Thresholds

- Locations Selection

- Alerts

**7. Interactions & Interfaces**

APIs

* *Sensors:*

Sensors are the input devices which feeds data to gateways. They sense temperature of surrounding and provide it to gateways. The data is transmitted through byte stream. Sensors needed to be configured. This configuration is provided through external configuration file(like XML file).

APIs for sensors are

1. send(byte array[])

send data in the form of byte stream to gateway

1. format()

pack the data into the format provided by gateway.

1. configure(file xmlFile)

configure the sensor based on details provided in xml File.

* *Gateways:*

Gateways are the middle layer devices. They gather data from various sensors and collectively send them to app server.

APIs for gateway are

1. receive(byte array[])

receive stream of bytes from sensors.

1. format()

pack the data into the format provided by app server.

1. configure()

configure a gateway. Configuration include which sensors are connected to this gateway,type of sensor,routing protocol information,port on which the communication is happening etc.

1. forward( destinationIP, routingInfo)

check the destination IP of received packet and forward on the outgoing link based on routing information

Type Handler API

Load new type handler

Stop existing type handler

Get gateway list

Rest APIs

Threshold checking

Get sensor data at logic server

Register a device

Update XML config

Authentications APIs

Credential Checking

Only Authorised access to API

8. Persistence

- *Server config:*

You will be asked to fill following details while configuring the server.

* Enter filter values for the each app.
* Also specify the data format received from sensors.
* Also set up the timeout for the each data.
* Mention Database connection details like port and IP address.
* Specify communication port.
* You have to mention supported devices.
* Specify types of gateways along with no. of gateways.
* Expose API for registering and unregistering

- transient state :

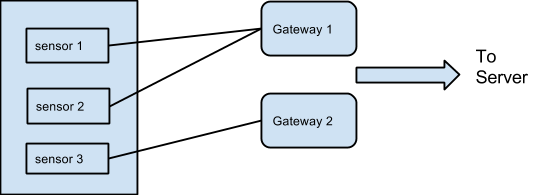
Sensor information and DB files

**6. Low Level Design (for each of the 3 modules:**

1. Sensors, Gateways, and communication.
2. Filter server and App engine
3. Registries, Repository and client apps

* **Sensors, Gateways and Communication channel:**
  + *Functional Overview:*
    - **Sensors:** Sensors will collect the raw data and forward to gateways.
    - **Gateways:** Gateway will simply route the data to server for processing.

* + *One block diagram giving more details on this part:*



* + *List of sub-systems in this part*
    - Sensors and Gateways

* + *List of services/capabilities in the part*
    - **Sensors:** Sensor will provide raw data. Capability of sensors is not specific as sensors are of different types.
    - **Gateways:** Gateways will be able to detect the best route to the server and forward the data to central server, considering traffic on different routes.
* **Filter server and App engine:**
  + *Functional Overview*
    - **Filter Server:** Filter server will process the data and check for the values w.r.t threshold. And forward these values to app-engine.
    - **App Engine:** App-Engine will fulfill the data need for various application.
  + *Block Diagram*

* + *List of subsystem in this part*
    - Filter server, App-Engine, Various applications.
  + *List of service/capabilities in the part*
    - **Filter server:** Filter server will be able to filter the data according to various threshold of different sensors and forward to app-engine. Various sensors may have different threshold values, filter server must be able to process it according to particular sensor.
    - **App-Engine:** App-Engine will be able to provide data to applications.

* **Registries, Repository and Security:**
  + *Functional Overview*
    - **Registries:** Registry contains the sensor information. Sensor id, sensor type and installation timestamp.
    - **Repository:** Repository contains data related to sensors.
    - **Security:** Security for client application, communication module and filter server.
  + *Block Diagram*
  + *List of Sub-system in this part*
    - Registry, Repository and Security for client app and communication module.
  + *List of service/capabilities in the part*
    - **Registry:** Registry is a database which is used to store the sensor information such as sensor id, sensor type and sensor installation timestamp. Registry will be helpful and capable of differentiating between the various sensors and their data.
    - **Repository:** Sensor data repository.
    - **Security:** This module capable of providing security to communication channel, server and the clients accessing the data on the server.

* **Interactions between this and other parts:** 
  + Sensors to Gateway :

Sensors send temperature data along with sensor id to the connected gateway over wireless channel. Gateways listen on a specific port for incoming data and provide an interface for sensors to communicate. This interaction will be implemented using java sockets. One sensor will be connected to only one gateway.

* + Inter-Gateway Routing :

Gateways are used to route temperature data collected by sensors to filter server. Multiple gateways can be interconnected and can follow a routing protocol so that if any link between gateways goes down, data can still be routed to filter server using some other path. They will recognize other gateway nodes within range, set up simple routing tables of adjacencies, and discover likely paths to the appropriate integrators/ filter server.

* + Gateway to Filter server :

Gateways aggregate, prune and bundle the data, then encapsulate in IP for delivery to the filter server as a small data “stream” . This data packet is sent to the filter server in real time over the TCP/ UDP communication channel. Filter server listens on the specific port for incoming packet.

* + Filter server to App engine :

Filter server provide interfaces to access different kind of sensory data. App engine interacts with Filter server by calling the registered service method through RPC at regular intervals or whenever needed.

***Group 4:***

***1. Gangasagar Patil***

***2. Poorva Bhawsar***

***3. Sushant Makode***

***4. Kedar Biradar***

***5. Omprakash Shewale***

***6. Nikhil Barote***

***7. Harshad Jalan***

***8. Kapil Chhajer***

***9. Diwas Joshi***