Design Requirements
For
Communication Module ,
Sensor Manager
And
Monitoring

Internals Of Application Server

Team Report

Group 6 Team 1

Submitted By -
- Pratik Nashine (2020201021)
- Mudit Kumar Bhansali (2020201067)
- Ayushi Maheshwari (2020201053)

Index

1. Introduction to the Project

Introduction

The Internet of Things (IoT) strives to connect devices remotely for seamless functioning and ease of operations. For developers, an IoT platform provides a set of ready-to-use features that speed up development of applications for connected devices as well as take care of scalability and cross-device compatibility and allows developers to spread out the applications, remotely collect data, secure connectivity, and execute sensor management.

Thus, an IoT platform can be wearing different hats depending on how you look at it. It is commonly referred to as middleware when we talk about how it connects remote devices to user applications (or other devices) and manages all the interactions between the hardware and the application layers. It is also known as a cloud enablement platform or IoT enablement platform to pinpoint its major business value, that is empowering standard devices with cloud-based applications and services. Finally, under the name of the IoT application enablement platform, it shifts the focus to being a key tool for IoT developers.

Some Features Provided are:

- Interface for uploading an Application Package to deploy the application.
- Interface for analyzing statistics and health of the platform.
- Interface for scheduling various algorithms of applications.
- Interface for managing and uploading details about various sensors deployed.
- Deployer service for Load Balancing and deployment of algorithms.
- Security service for managing user authentication and authorization.
- Scheduler service for scheduling applications and algorithms.
- Heath Manager service for analytics and fault tolerance.
- Sensor manager service for sensor data binding and sensor management and registration.
- Action manager service for getting a variety of outputs and to execute actions based on algorithm requirements.

Communication Module, Sensor Manager & Monitoring

Communication module: This will provide communication between sensor module and algorithm manager.

Services Provided by this Module are:

- Handling the communication channel and communication framework of the whole system.
- Handling the transfer of request and response among various modules.
- Keeping communications online.
- Secure and virtually lag less transfer of data among the various components.

Sensor Manager - Collects data from sensors present at different locations and binds it with respective algorithms and will help in registration of sensors, processing sensor data.

The **purpose** of the sensor module is -

- to manage interaction with the sensors, present at different geographic locations and collecting different data from the environment
- collecting data from the sensors
- processing sensor data so that it can be used by different applications implemented on top of our platform
- storing data from sensors in a database.
- Giving information about sensors being active or not.
- Binding data collected from server to the algorithms used by applications as per their access permissions.

Monitoring: Monitors all the modules present in our platform and checks if they are working properly, detects problems or abnormal behavior and informs about it to the fault tolerance manager.

- **Capability**: Keeps track of the status of proper functioning of all the different components, so that if there is some issue we can resolve and our platform works smoothly.

2. Use Cases

2.1 Usage Scenarios:

- Smart home system application -
- Traffic analysis system -
- Smart City -
- Health Monitoring - Glucometer, Blood Oxygen level

2.2 List of what the users can do with the solution provided by our team

1. Users can call for the specific data from the sensor which is registered and want processing from that data and want the algorithm to run according to our data.
2. Monitor will have a monitoring of each and every module of our platform, to help in proper functioning of our platform.
3. Users can register new sensors with our platform.
4. Communication is provided between different modules with the help of communication moduleUsers can take decisions based on result of the algorithm execution

2.3 Users of the platform

- **Platform Manager** : The platform manager starts the platform and monitors the health and performance of the services provided by the platform.
- **Application Developer** : Application Developer deploys the application on the platform.
- **Application Administrator** : Application Administrator will be able to monitor the running applications.
- **End User** : The end user who is going to use the services provided by the application deployed on the platform**.**

2.4 Requirements

The main requirements of the platform(contain sensor manager , monitoring and communication module) are as follows:

- Sensor manager and module:Important for registration of sensors for our platform and passing data when in need, take input provided by the user about data to be used from which sensor.

- Communication module:Provide communication between Sensor and sensor manager and even between all the other modules of our platform
- Monitoring:Provide health status of our platform by providing health status of each and every module.

3. Test Cases

Sensor Module:

Test Case 1: - Controlling the sensor – checking if the command sent to the sensor is working correctly or not.

Input – Action (like switching off AC)

Output – Checking if AC responded or not

Test Case 2: - Checking if registration of a new sensor is successful or not?

Input – Sensor information via json file

Output – Sensor simulated or not

Test Case 3: - checking simulation of real time sensor, sending continuous data

Input - sensor type and location of sensor sending Real Time data

Output – stream of data sent to algo and corresponding analysis.

Test Case 4: - extracting data from sensors based on geographical location and type of sensor.

Input - Geographical location and Sensor type

Output – All sensor ids with given location and type

Test Case 5: - Checking if data sent to application via communication module is successful or not.

Input – Data from sensor to communication module

Output - output of corresponding algorithm matches the data that is sent by the sensor

Communication Module:

Test Case1: - If two modules are involved in communication then they are able to send and receive data from each other.

Input: One end will send message

Output: The same message should be received by the other end.

Test Case2: - Parsing is the second case while testing for communication module we should have checked for data parsing happening correctly or not.

Input: actual data

Output: parsed data

Test Case3: - How to handle load and manage communication?

Test Case4: - How different types of data to be handled from different sensors in case of sensor data.

Test Case5: - How to handle dynamic behavior of Kafka.

Monitoring:

Test Case1: - This service will communicate with different components and get the status of all the components of the system.

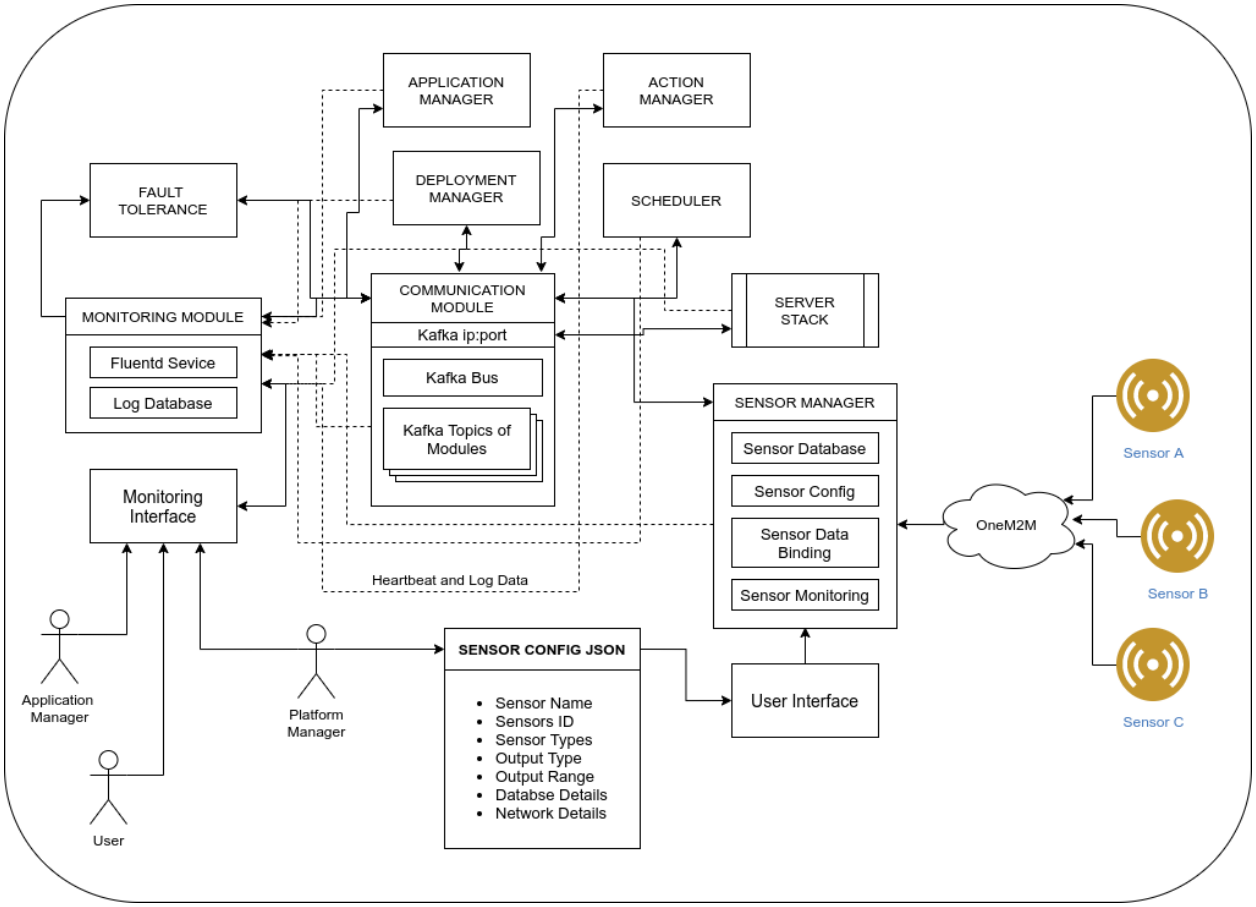
Input: Connection with all the modules/components in order to get status

Output: Will get status of each and every

Test Case2: - Check will be there on the heartbeat of each component and send response to the repository.

4. Solution design considerations

4.1 Design Big picture



4.2 Environments to be used

- 64-bit OS(Linux)
- Minimum RAM requirements: 4GB
- Processor: Intel i3 5th Generation

4.3 Technologies to be used

Docker: Applications will run inside the docker container.
Why: Isolates the environment in which applications will run and allows different applications to run with different libraries and different versions of the libraries and abstracts the underlying OS of our infrastructure.

Flask
Why: Tool provided by it will help to expose Api endpoints for the platform to use and it gives developers a good base framework of request/response management to work with.

Bootstrap
Why: it will be used for UI to build responsive web applications.

Kafka
Why: it will be used for managing communications among various modules

MongoDB
Why: For use as repository and Database

Python
Why: for deploying and developing platform

Amazon AWS
WHY: for hosting platform on cloud

4.4 Approach for communication & connectivity:

We are using Kafka to enable communication between different components of the platform. Initially the different platform components will be registered with the Kafka broker and will be sending status/messages/acknowledgements in predefined format.

4.5 Interaction with other Components (Internal design overview)

- **Servers, Sensor Manager and Action Manager**
  Server requests the sensor data from the sensor manager based on the application deployed and sends the action details to the action manager.

- **Monitoring and Fault Tolerance and Others**
  Monitoring module receives status updates, heartbeat pings and logs from all the modules and fault manager uses this information to reschedule failed services, applications and services.

4.6 Wire and file formats

- The sensor configuration file is json.

5. Application Model and User's view of system

Key elements

1. **Application Config file** - Contains info about the application used i.e., type of sensor used, name of sensor, controls, executables - all the scripts used in application.
2. **Sensor Config file -** contains details about the sensor used by the platform like sensor name, id, type, output type, output range, network details etc.
3. **Application Platform -** Sends config file to the platform and receives output from the platform.
4. **Sensor Manager -** Collect data from sensors and store it or preprocess it for algorithms using it, in case of request from algorithms provide them with sensor data via communication module.
5. **Sensor Registration -** if the application needs new sensors, then it will request the platform to add it, and the sensor registration module will store all information about the new sensor in the database and register it.
6. **Platform manager -** Manages the flow of the application and when a new application is received, provides it with resources.

Key Steps in whole application development Process -

Develop:

- Understanding the platform and develop application based on a Config file containing -

```
{
        Sensor types: [ Temperature Sensor , Camera Sensor, Light Sensor] ,
        Controller Types: [ Street Light Controller ]
        Algorithms: {name: id}
}
```

Algorithm Script -

- contains info about all the sensors to be used, along with their id and location.
- what elements are to be controlled.
- what type format of data is needed.
- processing and interference from the data received.
- Notification to be sent.

- First a config.tar file must be provided by application developers which contains two information like application config.xml and scripting files, and now config.json contains algorithm interface template information and scripting files contain actual algorithms.
- The developed application must be compatible with our platform.

Deploy and configure:

- Now these details can be used and our sensors should be registered and then details like the running of sensors is given by the developer according to which our sensor will run and now when the app dev wants any sensor data then he/she will request for that .
- And then the sensor manager will provide the corresponding data from that sensor pass on data to the deployer which runs that on our algorithm and sends the output to the action handler, which will further process the output in a way that depends on sensor type.

Run and monitor:

- Contain info like when to run a particular sensor or whether it will run always or on schedule and monitoring of the system.
- Application will be executed on a docker instance with the dependencies running.
- Monitoring will be done on platform services and in case of a services failure the application will handle it with the fault tolerant system.

6. Key Data structures

Sensor Type Registration

- Sensor ID
- Sensor Type
- Sensor Data Type
- Sensor Data Rate

Sensor Instance Registration

- Sensor Type
- Sensor Location
- Sensor Purpose
- Sensor Status

7. Interactions & Interfaces

APIs

- **Sensor API –** The developer can use the sensor API to interact with the sensors and can request or control the sensors.
- **Monitoring API –** The deployer component will provide an interface/UI to the platform developer to check the health and status of the resources (containers in which applications are running and nodes). The platform developer can access the logs of a particular container and forward it to the application developer if needed. Low Level Design (for each of the modules)

8. Low Level Design :

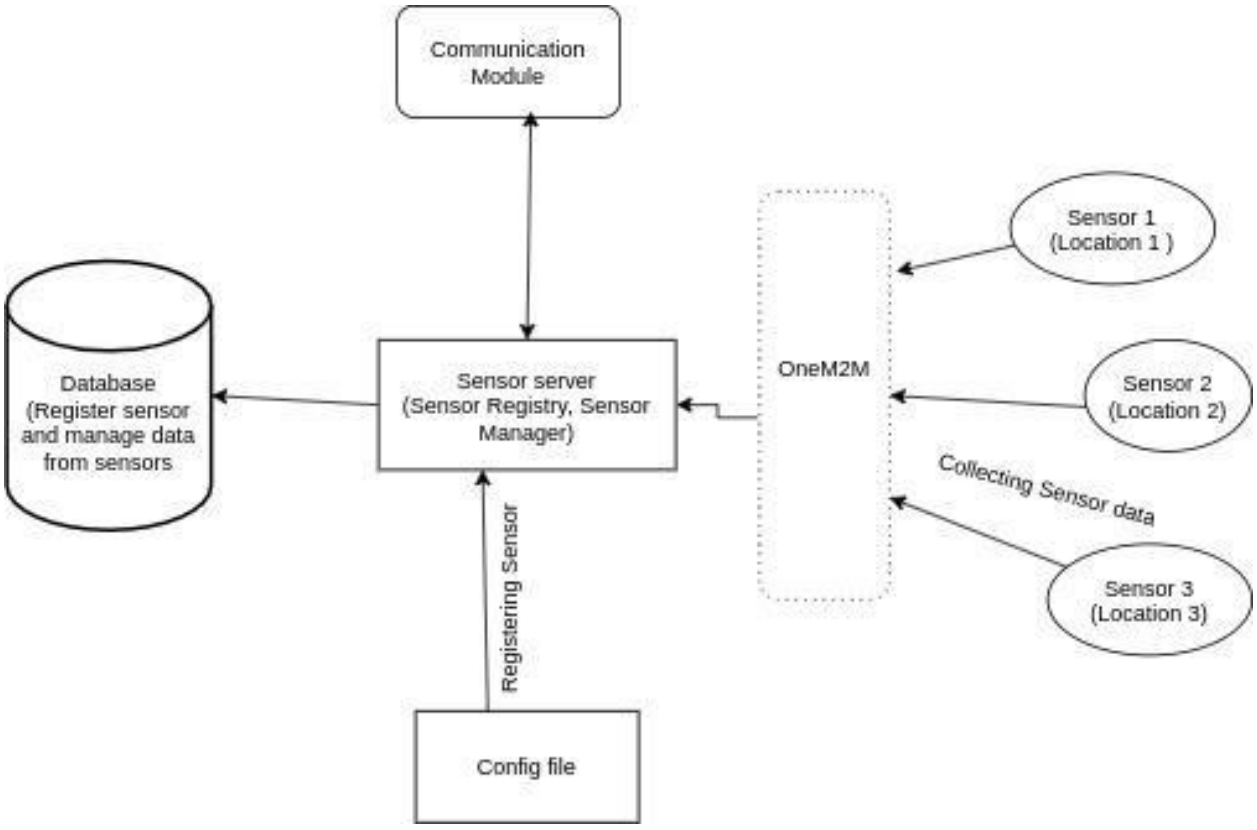
Lifecycle of the module:

Sensor Manager - Will inform about the status of sensors to the monitoring module continuously, also if some algorithm requests for data from the sensors then sensor manager will provide connection between the instance of sensor and the algorithm such that the data requested by the algorithm is provided.

While in case of **Monitoring**: This will always be on and keep record of all the components and stop when our platform is down.

And the **communication module** will also be always active as whenever some part of the application requests data from some other part, the communication module needs to provide communication to them.

Sensor manager:



Working Of Sensor Manager:

- When an algorithm needs some data from the sensor manager to provide results to the user, it will request data from sensor manager via communication module, sensor manager then on the basis of request will decide whether the request can be fulfilled or not, is valid or not.
- If the request is not valid, an error message is sent to the algorithm.
- If the request is valid, then the sensor manager will provide the algorithm with kafka topics for the instance of the sensor needed. And hence the algorithm can use data in chunks or streamlined or a single value as per the requirement of the algorithm.
- Sensor manager will also monitor the status of the sensors and will notify the monitoring module, when some sensor goes down.

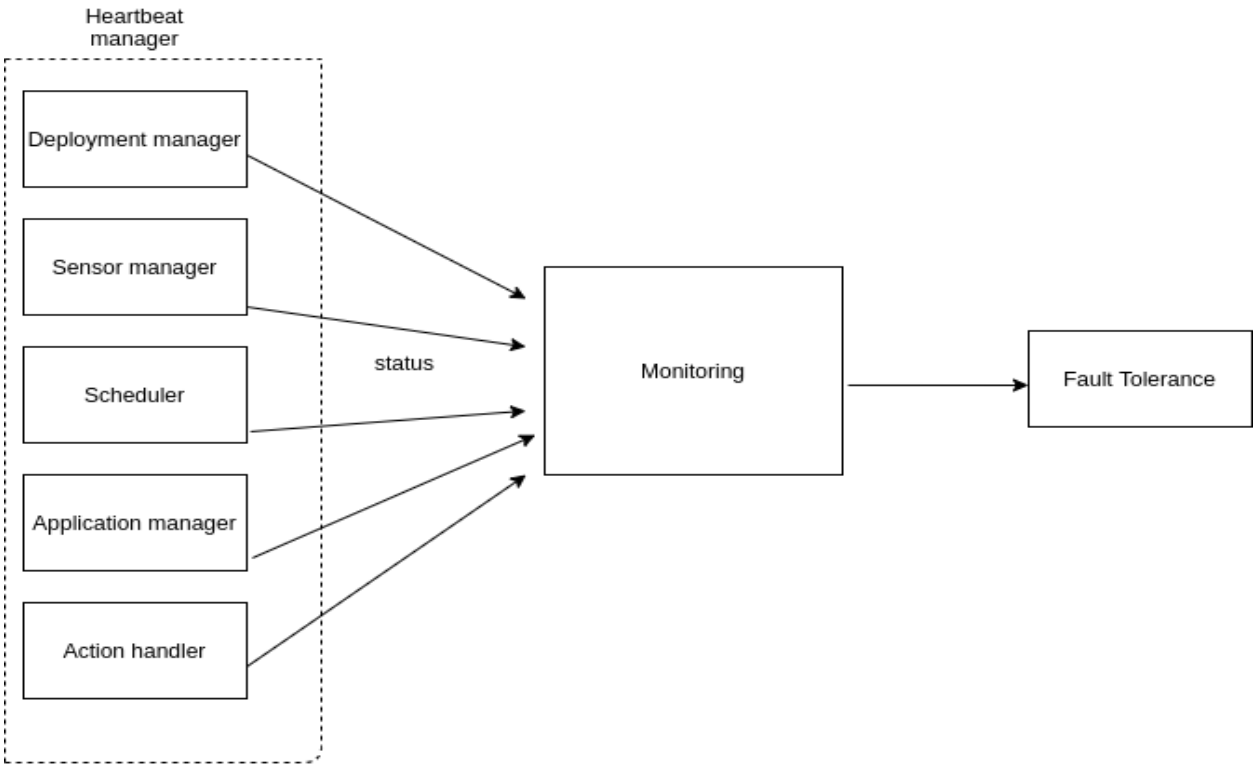
The **submodules** present are -

- **Sensor Registration** - takes config file as input from user containing details about sensor and then makes its entry in the database. And then connect to that sensor to collect its data.
- **Sensor Server** - Sensor server has all the information about sensors present and relates to interface to receive sensor data and database storing sensor data.
- **Data Binding** - It binds collected data to the algorithms which are implemented on top of the platform.

Interactions with Other Modules:

- **Communication Module -** To provide sensor data and other sensor related details requested by Communication Module.

Monitoring -



Working of Monitoring module:

It involves in providing status from different modules of platform in order to check and monitoring of proper functioning of our platform.In this monitoring is connected to each and every other module via communication channels provided by communication module and these modules will send their status at some intervals of time and in this we our monitoring module will get to know about status of different modules and if there's any fault then fault tolerance will take care of that.

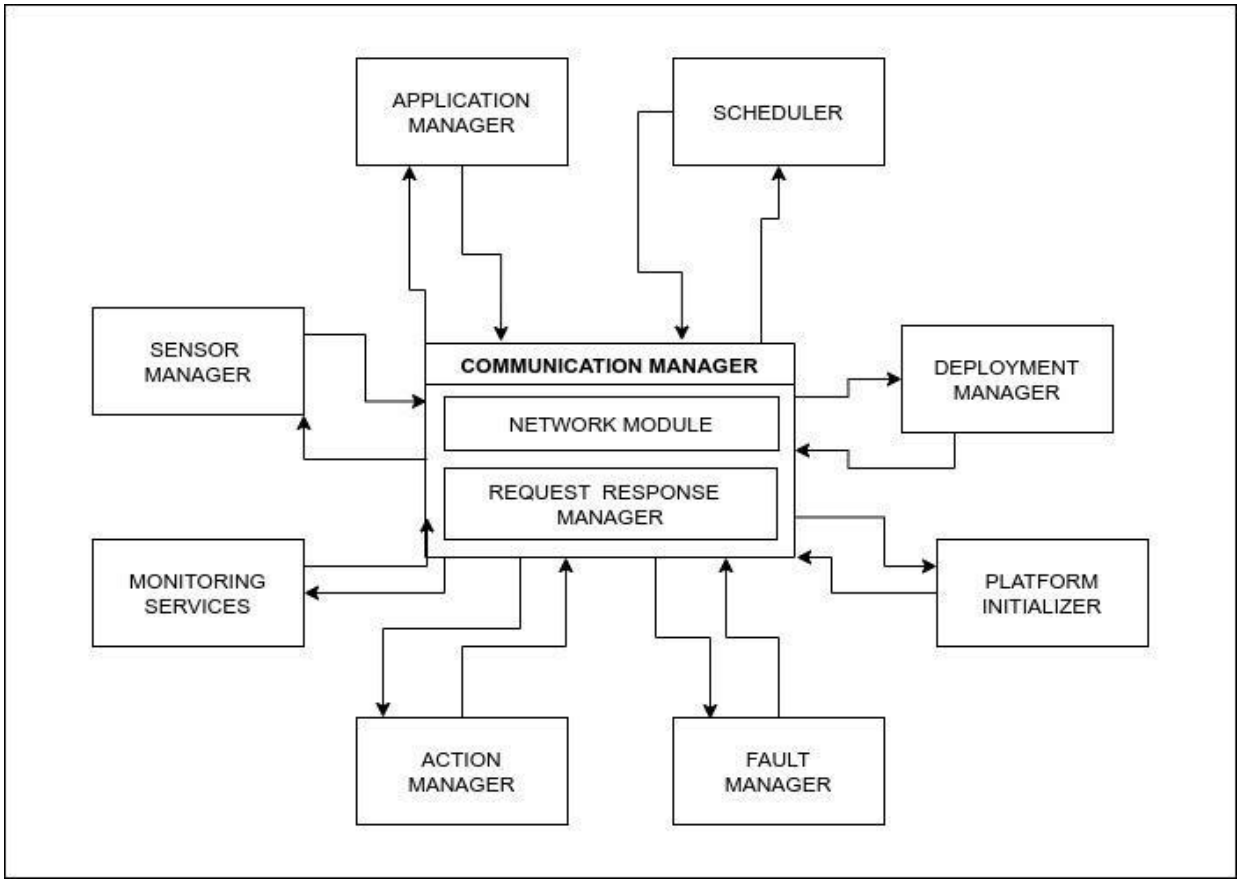
- **Subsystem: Heartbeat manager** is the **subsystem** in this part which will keep track of health i.e. proper functionality of different components in our platform like of Deployment manager, sensor manager, scheduler, application manager and action handler.

Interaction with other modules:

Monitoring will have connection with all the other modules whose health status it is checking, i.e. **Deployment manager, sensor manager, scheduler, application manager and action handler.**

Communication Module

The efficiency and speed of any distributed system depends greatly upon its communication manager which handles how different components of the system will communicate with each other and transfer the requests and responses among them based on some communication standard.



Sub Systems of this Module are:

- **Network Module:** The network module will basically implement our communication standard using Apache Kafka which will help in the setup of the communication channels between all the modules.
- **Request Response Manager :** This module will deal as a traffic router and will help with the routing of requests and responses to their respective senders and receivers.

Interactions with other Modules are:

All the modules will be integrated with this module to be able to communicate among each other natures of such interactions are as follows:

- **Action manager** accepts actions to be performed and sends the responses required.
- **Sensor manager** accepts requests for data binding and sends the responses to the application managers and other modules.
- **Fault manager** requests the stat report from Monitoring services and send the response based on the report to the deployment manager and other modules
- **Monitoring services** pings all the modules for their stats and stores it in the log database and responds to requests from fault managers.
- **Deployment Manager** accepts requests for applications to be deployed from scheduler and fault manager.
- **Application manager** sends the request for required data to the sensor manager and application details to deployment manager and scheduler.
- **Scheduler** sends the request to the application manager to fetch the application and sends the request to the deployment manager to deploy an application.
- **Platform Initializer** initializes all the components and sends their respective configurations to them.