Internet of Things Apps Platform

A Project Requirement Document

Submitted by

Team 1 :

Sourabh Dhanotia	201405605
Lokesh Walase	201405597
Pankaj Shipte	201405614

Team 2 :

Atul Rajmane	201405529
Prerna Chauhan	201405544
Aditi Jain	201405549
Abhinaba Sarkar	201405616

Team 3 :

Veerendra Bidare	201405571
Swapnil Pawar	201405513
Abhishek Mungoli	201405577

Under the guidance of

Mr. Ramesh Loganathan

For the course

Internals of Application Server

IIIT, Hyderabad

February, 2015

Contents

List of Figures

Chapter 1

Introduction

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

Functional Overview

2.1 Key Functionalities

i. Register new sensor with the server

ii. Simulate time variant sensors with the help of device factory

iii. Send data from sensors to its corresponding gateway (At gateway)

iv. Convert sensor data as per format of communication protocol

v. Send sensor data to Filter server using TCP socket (At filter server)

vi. Authenticate data received from gateway

vii. Process the received data as per the defined rules

viii. Store the processed data, source info and other required details into the database(At application tier)

ix. Listen to the requests from end applications

x. Convert these requests into the format of filter server and forward the same(At filter server)

xi. Listen to the requests from application tier and process the request

xii. Pull data from database or gateways according to the request type

xiii. Send processed data to the application tier

xiv. Process the data received from filter server and extract meaningful information for the app and send the result to it

2.2 Block Diagram

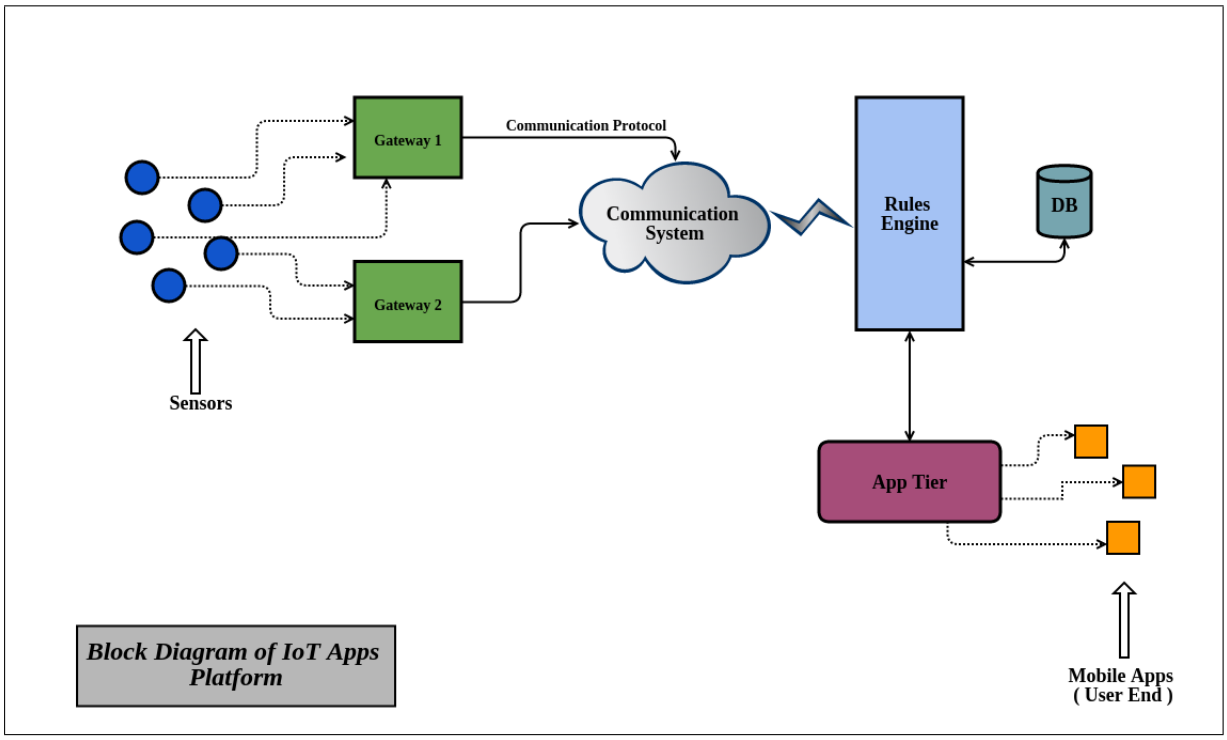

Figure 2.1: *Block Diagram of IoT Apps Platform*

2.3 Brief Description of Components

i. **Sensors :** Generate data

ii. **Gateway (Raspberry pi):** Receives sensor data. Convert it to format of communication protocol and send it to the filter server using TCP sockets.

iii. **Filter Server (Apache Jena):** Listen to requests from App tier. Fetch the required data from Gateway or DB. Authenticate its source and send response to app tier

iv. **Database (Mongo DB):** Filter server uses DB to store processed data.

v. **App Tier:** Listen to requests from applications. Convert them into proper format of filter server. Get the required data and respond back to the apps.

vi. **Apps:** Android apps that deliver the final information to the end user.

2.4 Three Major Parts

i. Gateway and Sensor implementation

ii. Implementation of Filter server

iii. Implementation of Application tier and app interface

Chapter 3

Use Cases

3.1 Features

i. Platform enables remote devices to operate, be monitored, managed and controlled as though they were connected directly to tablet or smartphone

ii. Connections are secure and private. Each device is given a unique identity so that it can be contacted directly and controlled remotely.

iii. Also provides near real-time decision making capabilities with unprecedented low-cost and simplicity

3.2 List of Usage Scenarios

i. **Guided Parking System :** It is designed to ease the parking process of a four wheeler in big parking facility. Typically, the available parking space is either not kept track of, or done manually. These inefficient ways of handling parking space results in Car owners moving back and

forth between the floors searching for free parking space. In order to tackle this problem, an IoT application which uses camera to identify free parking space can be built.

ii. **WildLife Monitoring System :** An IoT application built on this platform can also be used to make a census of animals in remote forest areas. Making the census manually, in such areas is simply not possible as some animals are mysterious, some dangerous, some nocturnal etc. An IoT application with motion sensors and cameras in animals frequenting areas like river banks, lakes etc fetches many photographs of the animals. Using unique identification features of animals (like stripes on the body is unique for Tigers), we will not only get an accurate count of some species of animals, but also spot animals previously thought to be extinct or not found in the region.

iii. **Gas Leakage detection System :** In some manufacturing industries, some gases are used, which can be poisonous. Although these gases are stored in strong containers, with normal wear and tear the container can become weak and hence lead to gas leakage. An IoT application with gas sensors spread across the gas storage facility can be used, to respond to such catastrophic scenarios quickly.

iv. **Vehicle Tyre Temperature Monitoring :** In summer days, when a vehicle is riding on a highway for a long period of time, its tyre temperature is likely to rise to a level that can cause tyre burst. This is one of the causes of accident on national highways. Thus we can install

temperature sensors near vehicle tyres which can alert the driver when the tyre temperature crosses a threshold and prevent accident.

v. **Smart Garden :**Create a small network of sensors in the garden, which automates the gardening.One such application for use is within a greenhouse, which may need to monitored in case it gets too hot for the plants, then venting of air could be automated. Similarly the soil moisture content can be monitored and depending on the moisture level, water can be supplied using automation. Another thing that can be monitored is the sunlight level which will prevent plants from getting excessive sunlight and drying up.

vi. **Automated Railway Barricades :** It is quite costly to put a man to monitor railway barricades at every railway crossing. Thus we can install motion/vibration sensor on railway track after some interval of distance (say about 3-4 kms) which can alert the barricades enclosed in that distance and automatically barricade can be closed.

3.3 Test Case

For the use case - "Guided Parking System", the following scenarios are possible :

i. The user at the parking building will turn on his app send request to the server. The server responds with exact free parking slot details(for eg - floor number, section number) on his app. This slot is the

nearest one to the user and he is given full directions details to reach the destination.The user is expected to reach his destination within a time period.

ii. If a free parking cell is identified and the parking cell is allocated to a user, a session is maintained for certain period of time within which the user is supposed to reach the parking cell. On failure to do so in that period of time, that cell is deallocated from that user and reused to allocate to other users.

Chapter 4

Overview of IoT App Platform project

- **Definition :** An IoT App Platform is the way that digital information technology can integrate the physical world to the online world to provide a common interaction platform. It can be viewed as a global infrastructure for the information society, by interconnecting (physical and virtual) things based on, interoperable information and communication technologies towards providing advanced information services.

- **Scope :** IoT promises to bring smart devices everywhere, from the fridge in your home, to sensors in your car,even in our body. This application platform offer significant benefits: helping users save energy, enhance comfort, get better healthcare and increased independence: in short meaning happier, healthier lives.

- **Registry & Repository :**A registry is a list of items with pointers for where to find the items, like the index on a database table or the card catalog for a library. A repository stores the actual items, like a database table itself or a library's shelves of books. If you lose a registry,

the items still exist; you just may need to reindex them. If you lose a repository, the items are gone.

- **Sensor Registry & Sensor Repository :**Sensor Registry keeps logs of all the recently active sensors while Sensor Repository contains all information about every devices, their performance and last activity details.

- **Logic server [Application Platform as a Service (aPaaS)] :** Logic server simplifies the technical aspects of creating and deploying applications because it's easier to maintain, scalable and tolerant to faults. Logic server authenticates data received from gateway,processes the received data as per the defined rules and stores the processed data, source info and other required details and provides it to the end-user.

- **Location Services :**Location services are registered in the registry. The location service is provided by GPS, or by Sensor ID. As sensors are not mobile under this circumstances,they are used for location as well.

- **Mobile Interface** The mobile-app interacts with the App-tier to get the knowledge about info to be displayed . Depending on type of signals received it displays the corresponding message.

Chapter 5

Interaction and Interfaces

5.1 User Interactions

i. User will launch the app and as he/she interacts with it, according to the app logic a RESTful service is called with the required parameters sent in a JSON format.

ii. Filter server processes the data received as to deduce the requested action before executing the target service.

iii. Service generates the appropriate response and sends it back to the requesting app.

5.2 Module to Module Interactions

i. **User -** End users interact with the system through the apps running on handheld devices.

ii. **Rules Server -** This is the server which will be responsible to process all the users requests. It will interact with other subsystems to achieve

desired results, such as in case of Smart Parking System it will interact with the systems like, Location and Maps etc. It is also responsible for processing data from various gateways and then storing it to the Sensory Data Repository , processing of data includes Unmarshalling of data received from the gateways , and applying some checks on data and retrieving information from raw data received.

iii. **Sensory data Repository -** It has the information related to various parking lots, which is used by Rules Server to answer the queries generated by the Controller of App Server. This repository is updated by the rules server periodically.

iv. **Gateways -** These are responsible for collecting data from various sensors and forwarding it to the rules server. There can be any number of gateways between the sensors and the rules server, they are basically used to forward the sensory data to the rules server.

5.3 File/Wire Format Definitions

5.3.1 Sensor Packet

These are the packets transmitted by sensors.

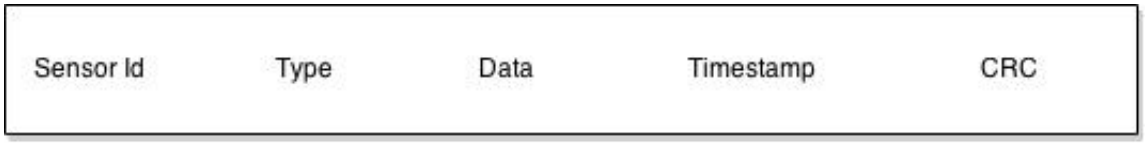

Figure 5.1: *Sensor Data Packet Protocol*

- **SensorID:** It is a unique ID given to each Sensor.

- **Type:** It describes the type of data that sensor senses in our case it is Heat, Proximity and motion sensor.
- **Data:** It signifies the data associated with the sensor, like its position , or data generated by it.
- **Status:** It will show whether the corresponding parking spot is free or reserved.
- **Timestamp:** It will contain the time at which the packet is generated.
- **CRC:** It will contain checksum for error detection.

5.3.2 User Packet

User packets will be generated by the user, when he/she launches the application.

Figure 5.2: *Device Data Packet Protocol*

- **DeviceID :** Its a unique ID of the user.
- **Type:** It will show the type of packet, ie. whether its request or response packet.
- **Data:** It will contain the data like users location from where it has generated the request, or other parameters needed by the Rules server.
- **Timestamp:** It will contain the time at which the packet is generated.

- **CRC:** It will contain checksum for error detection.

Chapter 6

Brief overview of each of the modules

6.1 Gateway & Sensors

- Wireless sensor networks almost invariably interact with a central architecture model. In our architecture, different types of registered sensors (viz, motion, temperature and proximity, etc) send raw data to the gateways.
- The gateway through its listener daemon listens to the incoming data from the sensors. The sensors send data to multiple gateways as a precaution against gateway failure.
- The gateway then converts the data to the required format as per the communication protocol.
- The gateway then sends the formatted data to the central filter server through TCP/IP

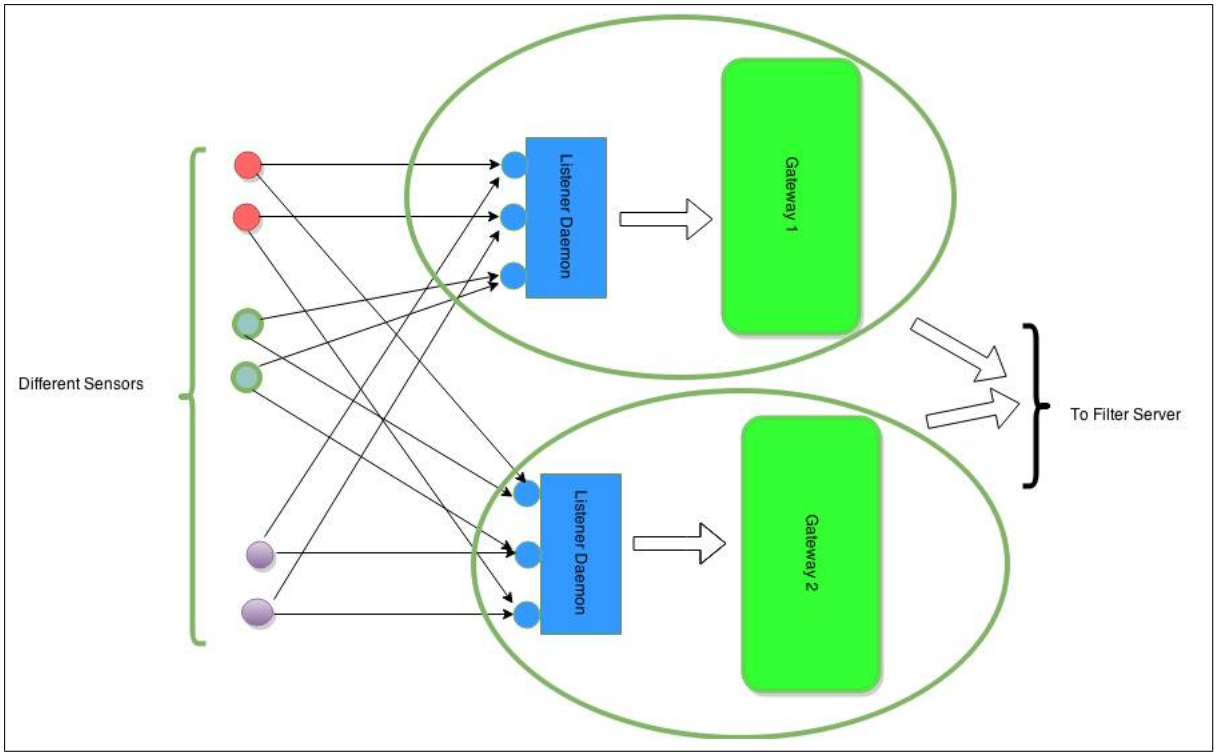

Figure 6.1: *Block Diagram of Gateways and Sensors*

6.2 Filter Server

This segment of the platform is the crux of the system. This acts as a bridge between the Gateways at one end and the App Tier on the other. Filter server receives data from the Gateways and runs this data against pre-defined rules. Each type of sensors has certain set of rules which are defined at the time of its set up. A rule describes an action that is requested to be taken when a certain threshold value is reached. The parameters specific to a rule are stored on the database which is requested every time rule needs to be referred.

Filter server communicates with the App tier of the system through RESTful APIs and uses JSON representation for the data. For example, if for some activity of a user on the Mobile app, App tier needs to match rules then

it will make such a request to a corresponding RESTful service and send along the required data in JSON format. And Apps in turn may decide to take a befitting action based on the result returned which as well is in JSON format.

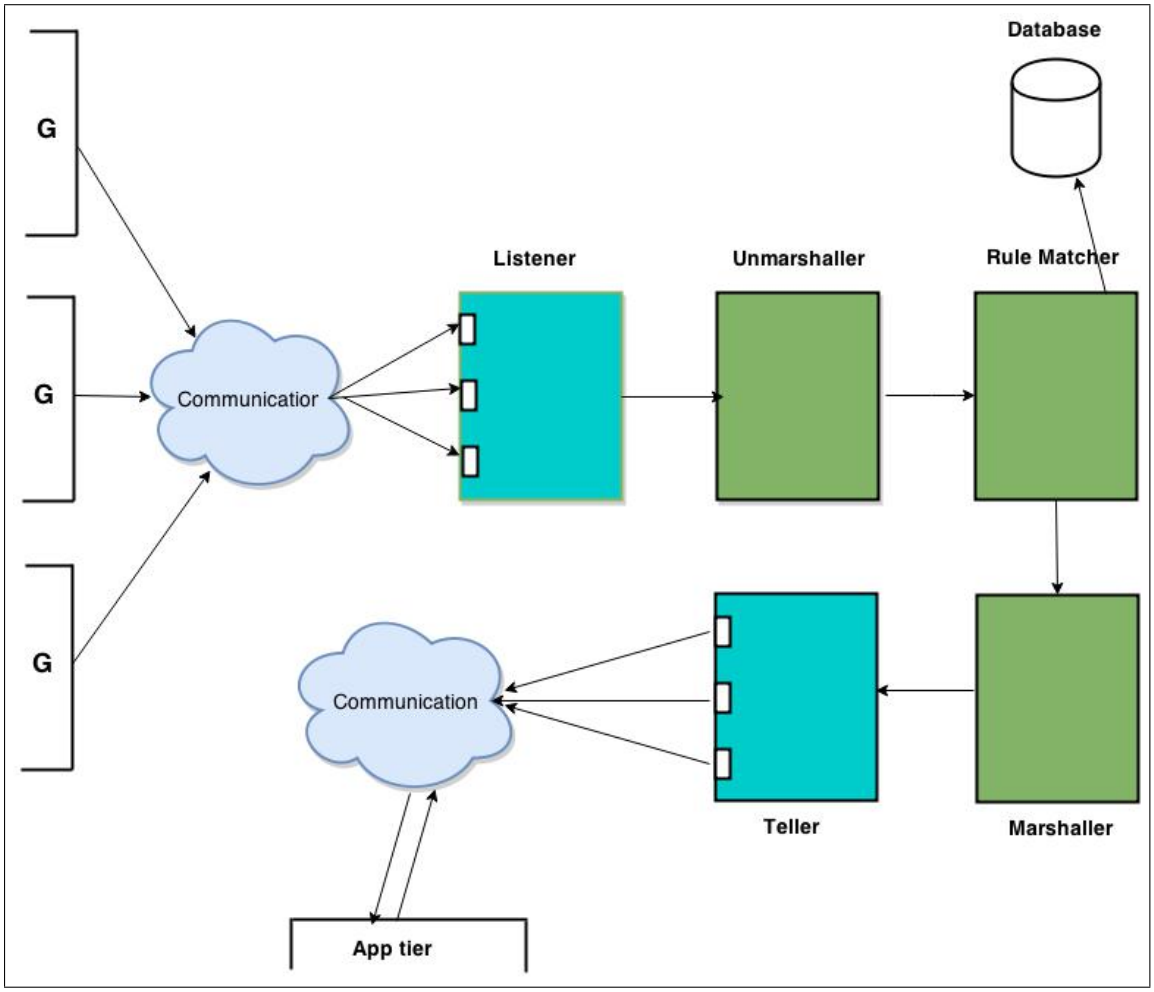

Figure 6.2: *Block Diagram of Filter Server*

6.3 Subsystems and their services/capabilities

6.3.1 Web Server

- **Listener daemon -** Keeps listening to the data pushes from gateways and passes along the data received to the unmarshaller.

- **Unmarshaller -** Converts the data received in a format that can be used by the rule matcher.
- **Rule matcher -** Refers to the database for the rules and runs them against the data received from gateways
- **Marshaller -** Converts the decisions of the rule matcher into a format that can be sent over the network to the App tier
- **Teller daemon -** Keeps listening to the Marshaller and passes along the data received from the marshaller to the App tier

6.3.2 Database Server

All of the metric data corresponding to each of the rules are saved on the database. Rule matcher hits this whenever it has to run rules against some data.

6.3.3 App Interfaces

- For each type of mobile client a different app has to be provided since such apps are dependent of the underlying system. So there will separate apps for iOS, Android and Window phone.
- These interfaces initiate RESTful API requests and are capable of taking an appropriate action based on the response returned from the filter server.

6.4 Application Tier

- This segment of the platform consists of the end user portion which is iOS/Android apps residing on usersâĂŹ handheld devices. UserâĂŹs interactions with the app are sent over to the filter server and the app changes its state based on the response from the filter server.

- All the communication between the App tier and the filter server is through the RESTful services and the format used for sending over the data is JSON.

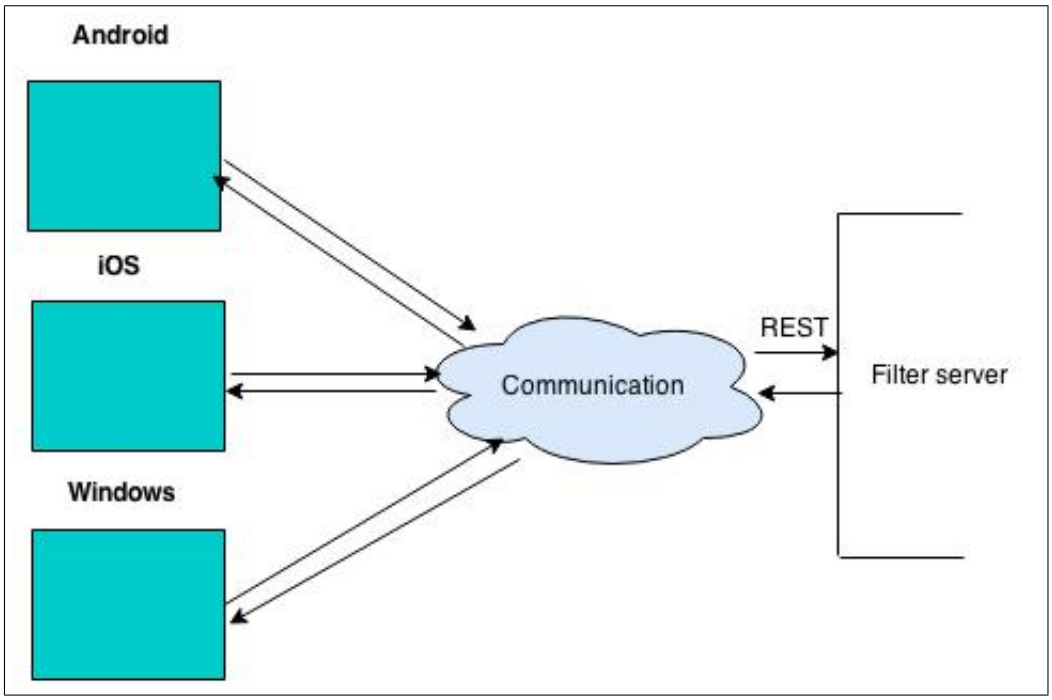

Figure 6.3: *Block Diagram of App Tier*