International Institute of Information Technology, Bangalore

Smart Water Networks
Project Report

Abhijith Madhav Kumudini Kakwani

December 2, 2014

Contents

1 Introduction

1.1 Problem

Water needed for the IIIT-B campus is sourced in three ways

1. IIITB has its own source of water in the form of three functional borewells. Water is pumped out of these borewells for almost twenty hours a day.
2. Water supply from BWSSB for a limited amount of time each day.
3. Almost 20000-30000 liters of water per day is procurred from commercial water tankers.

Currently there is no insight into how water is being used and about whether its use is optimal or not. Informal estimates term the per-capita water consumption within the campus as excessive.

1.2 Objective

There is a proposal to make the water distribution network of the IIIT-B campus a smart one with the installation of sensors in the network. Our system intends to plug into this smart network and work on the data that the sensors produce.

Specifically this project intends to

1. Be general purpose enough to plug into any small to medium sized smart water network(An institute, a corporate campus, a gated community, a farm etc)
2. Help in better water management by offering insights into water usage across the network.
3. Help in maintaining the water network by detection and reporting of anomalous events.

1.3 External Dependencies and Assumptions

1. Installation of flow, quality and level sensors at suitable points in the water network to make it a smart one.
2. Installation of a SCADA like system with a ODMS(A historian) from where sensor data can be read off.

In the absence of the above we have implemented a stop-gap sensors simulator which will fill database with psuedo sensor data until real sensors are deployed in the network.

2 Scope And Requirements

The focus was on architecting, designing and implementing the below

1. Supervisor mobile apps which
    - Detail water usage patterns.
    - Notify user on anomalous events and other happenings in the water network.
2. Architecting and implementing a modular backend storage system to
    - Represent the smart water network.
    - Manage all the data generated by the sensors.

We are limiting the extent of our implementation to **not** include the below

1. The mobile app for field personnal which assigns issues to them and helps them update the status of these issues.
2. Web UI's to construct the smart water network. We will load the structure of the smart water network in batch mode to our backend systems and then work on the same.
3. A visual geospatial representation of the water network where a centralized view of the whole network is possible. Here each network asset has a visual representation and its parameters can be inspected by clicking on them.

2.1 Use Cases

Use Case 1	**Insights into the water network**
Actors	Administrators
Activities	• Will be able to see the breakdown of water usage across aggregations with options to drill down to specific granularity. Aggregations can be buildings, blocks, floors etc. Aggregations can be nested. • Will be able to inspect health parameters of the water network. Health parameters can quality of water, electricity used, level of water in storage etc.

Use Case 2	**Predictions**
Actors	Administrators
Activities	• Will get predictions regarding future water usage based on historical water usage patterns so that planning for rationing/procurement of water can be done.

Use Case 3	**Anomaly Detections**
Actors	Administrators
Activities	• Will get notifications when there is a threshold breach for any of the sensors or when there is a leak

3 Architecture And Design

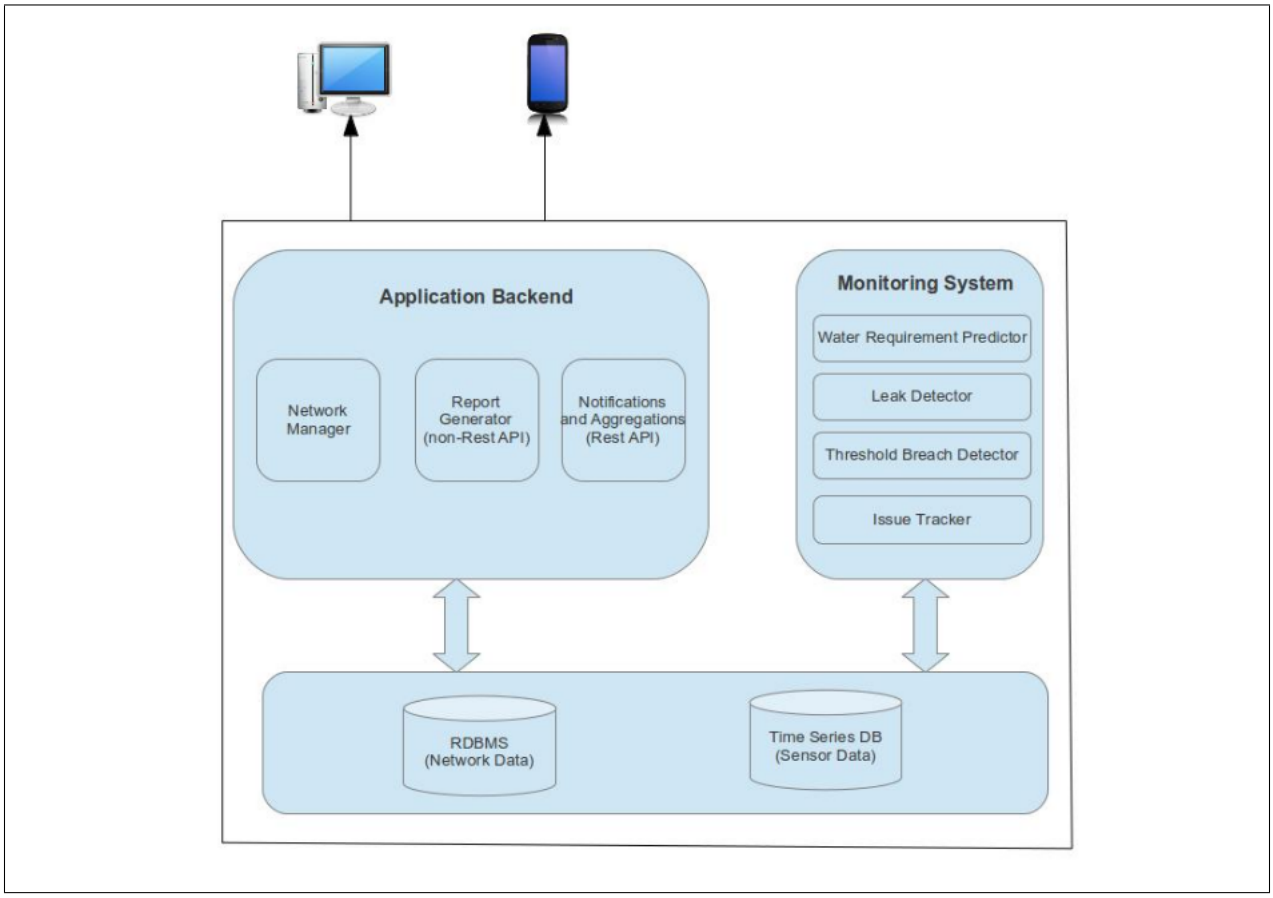

Figure 3.1: Architecture Diagram

3.1 Application Backend

3.1.1 Data Store Design

Representing The Water Network

The structure of the water network is stored in a relational database. The schema for the same is documented below. Common properties across all assets are stored in the asset table whereas differing properties are stored in an EAV table(asset_property_value_map). A EAV based table is being used to enable future addition of different assets with different properties into the database

without change of schema of the database or proliferation of tables.

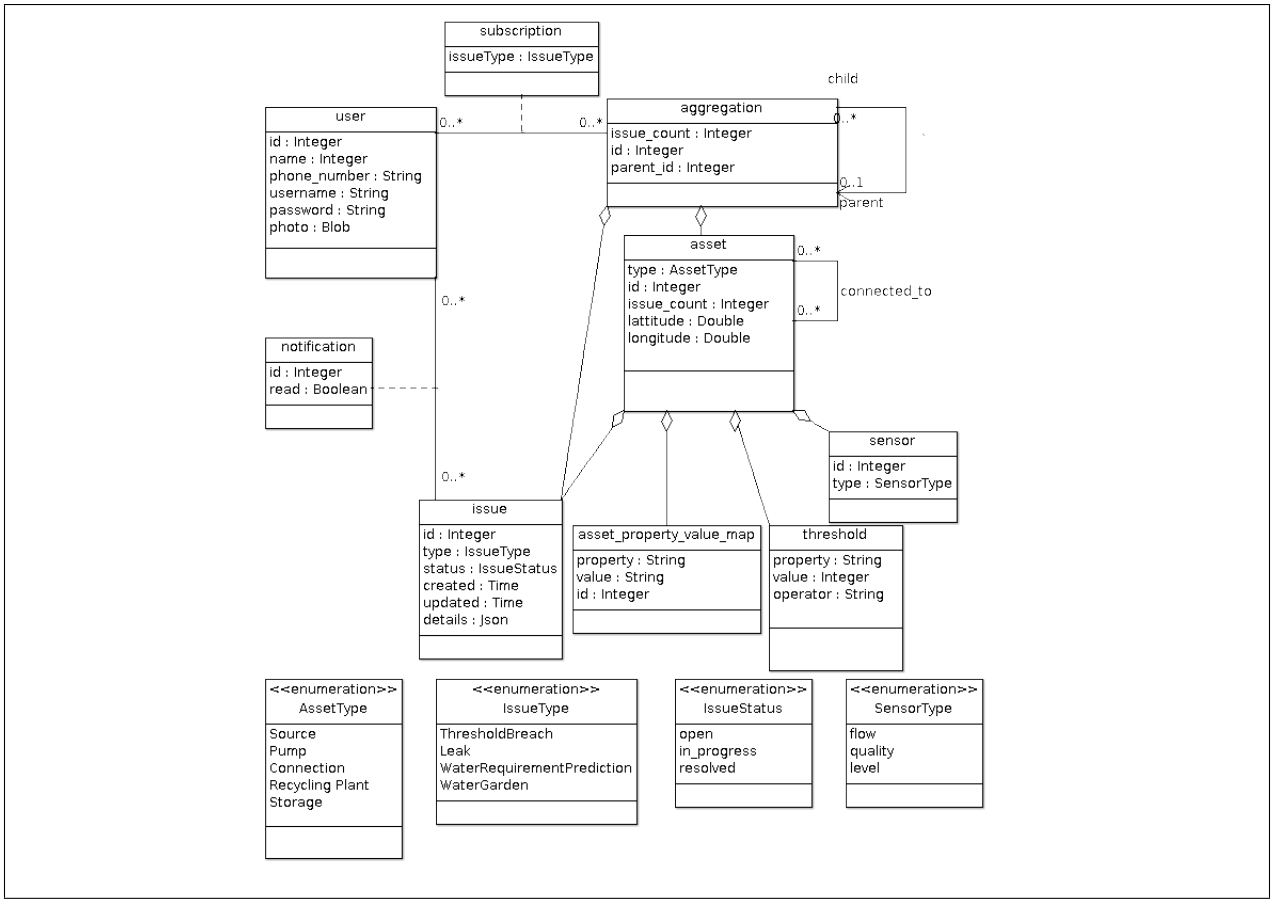

Figure 3.2: Architecture Diagram

Storing Sensor Data

The sensor(psuedo) data is stored in a time series database to help with time based queries. The entry point for this data is the sensor id which is stored in the network database.

- The sensor id represented in the assets table is an application defined reference to entries in a time series database table.
- All sensors(data simulator) write to a table(sensors) in the time series database, entries of which have the following structure.

```
{
        sensor_id : <sensor id>,
        label1 : <value for label1>,
        label2 : <value for label2>,
        .
        .
        .
}
```

3.1.2 Monitoring System

Scans the sensor data continuously for anomalies and reports the same to the issue tracker. Anomalies include breaches in threshold levels, leaks(not implemented) and external complaints(not implemented). It also reports daily water requirements according to it's prediction(Not implemented).

Data Simulator

This module simulates the sensors in the network and generates psuedo sensor data.

3.1.3 Web Application

Provides a REST API to query the following entities

- Notifications.
- Aggregations
- Assets

Provides a non-REST API to get the below data

- Water consumption break up.
- Water consumption trends.

3.2 Android Application

The android application provides three main views.

3.2.1 Notifications

Shows notifications by consuming the notifications REST API provided by the backend

3.2.2 Reports

Generates two reports, one on the water consumption break up and the other on the water consumption trend. Consumes the Non-REST API provided by the backend for the same.

3.2.3 Network Health

Provides a hierarchical view of the health of the network.

4 Technology And Implementation

The entire source code is versioned at
`https://github.com/smartwatermanagement/self-healing-water-networks`.

4.1 Data Store

The relational database used is MySQL. A database dump is available at `/SWNBackend/DB SQL FIle/swndb.sql` The DAO's for access of the same are at `/SWNBackend/src/dao`. The current database contains a sample water network shown below.

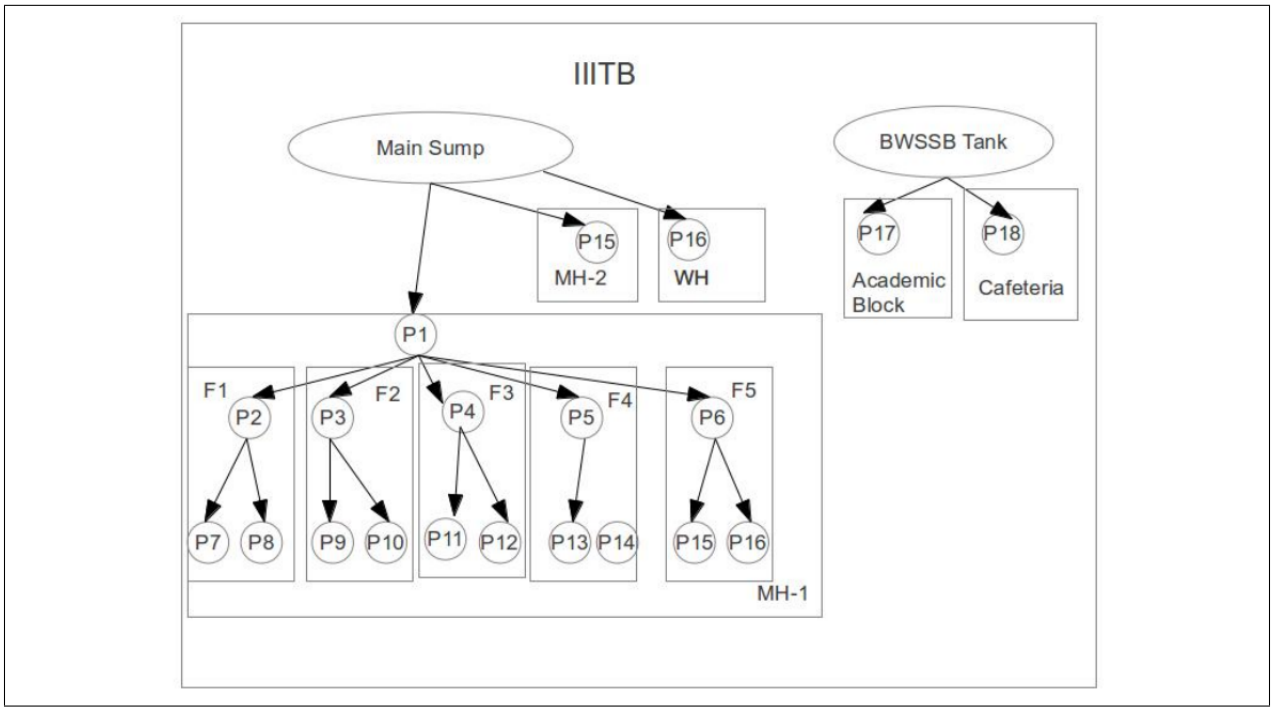

Figure 4.1: Sample Water Network

Sensor data is stored in a time series database called InfluxDB. When the actual sensors are deployed it is hoped that only a sensors data access object will have to be implemented. The current DAO object is at `/SensorsDataDAO`.

4.2 Monitoring System

`/MonitoringSystem`

A multi-threaded application which spawns off threads to do the following

- Generate pseudo data for the sensors table.
- Detect breaches in threshold levels set for sensors.
- Leak detector(Dummied for now).
- Water requirement predictor(Dummied for now).
- Data simulator to generate psuedo-sensor data.

Besides these, it contains an issue tracker module to create and modify issues and notifications.

4.3 Web Application

`/SWNBackend`

The web application is hosted inside an Apache J2EE application container. It uses the **Struts2** framework to establish an MVC architecture. Further, the **Struts2 REST plugin** and the **Struts2 JSON plugin** are used to set up REST API's.

4.3.1 Constructing The Water Network

`/SWNBackend/src/model/WaterNetwork.java`
The structure of the network is read from the database only once, when it is referred to for the first time, after the application backend is brought up. It is expected that updates to the network when implemented in future write through the memory. The representation of the network looks like a forest of trees each rooted at a source or a storage asset node after construction.

4.3.2 Calculating Water Usage And Trends

`/SWNBackend/src/action/nonrest`
The above required data are calculated by traversing up and down the water network data structure and getting the requisite flow data by making calls to the sensor database.

5 Status Of Completion

5.1 Completed

- Android application for the supervisor.
- Application backend which supports the API's listed in 3.1.3

5.2 To Do

- Algorithms for detecting leaks and predicting water requirements.
- A complaint API using which other third party clients can submit complaints.

6 Test Summary

7 Open Issues

- Currently aggregations can be defined only in a hierarchical manner.

8 Suggested Enhancements

- Push notifications in the android applications using Google Cloud Messaging infrastructure.
- When there are multiple notifications for the same asset, collapse them.
- Leak detection algorithm.
- Water prediction algorithm.
- The location information in the asset view and in the details of a notification must open out into a terrain map which shows the route from the current location to the asset location instead of displaying latitude and longitude
- JSON error responses for certain API's.
- Implement a proper color picker for the pie chart which displays water usage breakup.
- Features listed in 2

9 Installation Notes

- Clone repository `https://github.com/smartwatermanagement/self-healing-water-networks`.
- MySQL setup
  - Install and run MySQL.
  - Create a database and create schema using database dump file `/SWNBackend/DB SQL FIle/swndb.sql`
  - Update `/MonitoringSystem/src/utils/Constants.java` and `/SWNBackend/WebContent/META-INF/context.xml` with the above details.
- InfluxDB setup
  - Install and run InfluxDB.
  - Access InfluxDB terminal through a web ui at `localhost:8083`
  - Create a user account and database.
  - Update `/SensorsDataDAO/src/utils/Constants.java` with the above details.
- Build and run the monitoring system, `/MonitoringSystem`.
- Run **Apache** with the WAR of the application deployed.
- Install the android directly from the IDE used(Android Studio).