

**DESIGN AND IMPLEMENTATION OF A LIGHT  
WEIGHT, SCALABLE AND ASSISTIVE APPLICATION  
PROGRAMMING INTERFACE FOR INTERNET OF  
THINGS**

Ahesh Perera

(168251M)

Degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

June 2018

**DESIGN AND IMPLEMENTATION OF A LIGHT  
WEIGHT, SCALABLE AND ASSISTIVE APPLICATION  
PROGRAMMING INTERFACE FOR INTERNET OF  
THINGS**

Hetti Arachchige Ahesh Suranga Perera

(168251M)

Thesis submitted in partial fulfillment of the requirements for the degree  
Master of Science in Computer Science and Engineering

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

June 2018

## **DECLARATION**

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: .....

Date: .....

Name: H.A.A.S. Perera

The above candidate has carried out research for the Masters Thesis under my supervision.

Name of the supervisor: Dr. Indika Perera

Signature of the supervisor: .....

Date: .....

## Abstract

Cloud computing and Internet of Things (IoT) brings various physical devices which generate and exchange data with the services promoting the integration between the physical world and the computer world into a single common page. Together they have been providing various applications, use cases and services over the past few years, that has made a significant benefit on both industrial applications as well as day to day needs of humans.

On the other side of the coin, programming of the IoT based applications has become very challenging due to the vast knowledge base required in various technical domains, from low-power networking to the embedded operating systems, from low level calculations to the distributed algorithms and so on. It is certain that a well designed, reliable and scalable, easy configurable and high performance Application Programming Interfaces (APIs) are much needed in this paradigm to offer sophisticated services for an IoT cloud. APIs are generally exposed to its consumers as service endpoints to get pre-defined jobs done, and are offering convenient ways for developers to design and implement applications as well as vendors (OEMs) to design and manufacture their devices.

In this research I have mainly focused and discussed about the true challenges, issues and the concerns that we may face when designing and implementing high performance APIs for IoT cloud. I have also elaborated the technical and theoretical limitations come along with the performance issues in such APIs. Most importantly I have tried to design a platform for small start-ups who start developing their IoT based products with a limited knowledge, time, funds and resources so that they can build their products without worrying about the production level challenges in terms of scaling and performance once the business is grown up.

This research will provide a solution for most of the challenges when it comes to IoT cloud in terms of self configurations and elasticity with auto scaling whilst keeping better performance. Considering the massive variety of devices and the resource constraints we have in IoT, an architecture has been proposed for devices to be self-configured to the maximum extent with the API. The proposed solution will have a well designed RESTful API which comes in plug-and-play mode with developer convenience, supporting horizontal scaling as and when needed. In a nut-shell this gives a framework which takes care of all the architectural level challenges and best practices in IoT cloud where the engineering team focuses more on the business and the product.

## **ACKNOWLEDGEMENTS**

I am grateful to Dr. Indika Perera, my supervisor for accepting my research under his supervision and for the guidance, continuous support and the direction given throughout to make this research a success.

My sincere appreciation goes to my family for the support and the motivation given for making this thesis a success.

I would also like to thank my colleagues at work place, Sysco Labs for spending their valuable time with me to discuss about my research and opening gates to discover new areas.

Finally, I wish to thank all the academic and nonacademic staff of Department of Computer Science and Engineering, University of Moratuwa and my colleagues of MSC'16 batch for the support and encouragement provided throughout past 2 years.

# TABLE OF CONTENTS

|  |      |
|--|------|
| DECLARATION  | i    |
| Abstract   | ii   |
| ACKNOWLEDGEMENTS   | iii  |
| TABLE OF CONTENTS  | iv   |
| LIST OF FIGURES  | viii |
| LIST OF TABLES   | x    |
| LIST OF ABBREVIATIONS  | xi   |
| Chapter 1 INTRODUCTION   | 1    |
| 1.1 Background   | 2    |
| 1.2 APIs are Driving the Internet of Things                              | 3    |
| 1.3 Common IoT Challenges  | 4    |
| 1.4 Problem Statement  | 5    |
| 1.5 Motivation   | 6    |
| 1.6 High Level Research Objectives                                       | 8    |
| Chapter 2 LITERATURE REVIEW  | 10   |
| 2.1 IoT based related work   | 11   |
| 2.2 Web Service Protocol for Interoperable IoT Tasking Capability        | 11   |
| 2.2.1 Capabilities of IoT  | 12   |
| 2.2.2 IoT Architecture   | 13   |
| 2.3 Integrated Middleware Framework for Heterogeneous Internet of Things | 14   |
| 2.3.1 Requirements of M2M APIs for IoT Architecture                      | 15   |
| 2.3.2 Convergence of M2M APIs to RESTful Web services                    | 16   |
| 2.4 Web API Management Meets the Internet of Things                      | 17   |
| 2.4.1 Challenges for the Internet of Things and Web APIs                 | 18   |
| 2.4.2 IoTgw - an API Gateway for IoT protocols                           | 18   |
| 2.5 Problems and Limitations when designing a WEB-API of IOT             | 20   |
| 2.5.1 Implementation issues  | 20   |

|           |  |    |
|-----------|--|----|
| 2.6       | A Self-Configuration Architecture for Web-API of IoT | 21 |
| 2.6.1     | Requirements of WEB APIs in IoT                      | 21 |
| 2.6.2     | Related work   | 24 |
| 2.6.2.1   | ThingSpeak   | 25 |
| 2.6.2.2   | NimBits  | 25 |
| 2.6.2.3   | Cosm   | 25 |
| 2.6.2.4   | SensorCloud  | 25 |
| 2.6.2.5   | Evrythng   | 26 |
| 2.6.2.6   | iDigi  | 26 |
| 2.6.2.7   | GroveStreams   | 26 |
| 2.7       | Ad Hoc Networks                                      | 27 |
| 2.7.1     | Data Confidentiality                                 | 28 |
| 2.7.2     | Privacy  | 28 |
| 2.7.3     | Trust  | 29 |
| 2.8       | RESTful Sensor Data Back-end                         | 31 |
| 2.9       | WSO2 IoT Server                                      | 33 |
| 2.9.1     | Architecture   | 34 |
| 2.9.2     | Limitations for the target group                     | 36 |
| Chapter 3 | METHODOLOGY  | 37 |
| 3.1       | Proposed Solution                                    | 38 |
| 3.2       | Components   | 40 |
| 3.2.1     | Cloud API  | 40 |
| 3.2.2     | Front End Dashboard                                  | 40 |
| 3.2.3     | Device / Agent                                       | 40 |
| 3.3       | Cloud API  | 41 |
| 3.4       | Technology stack                                     | 42 |
| 3.4.1     | Node JS  | 42 |
| 3.4.2     | JSON   | 43 |
| 3.4.3     | Mongo DB   | 44 |
| 3.5       | High Level Modular Architecture                      | 44 |
| 3.6       | How does this work?                                  | 46 |

|           |  |    |
|-----------|--|----|
| 3.6.1     | IoT devices registration                       | 46 |
| 3.6.2     | Agents registration                            | 46 |
| 3.7       | Evaluation Plan                                | 48 |
| 3.7.1     | API performance                                | 48 |
| 3.7.2     | Ability to scale                               | 48 |
| 3.7.3     | Accuracy of the configuration scripts          | 49 |
| Chapter 4 | SOLUTION ARCHITECTURE AND IMPLEMENTATION       | 50 |
| 4.1       | Solution Architecture                          | 51 |
| 4.2       | Implementation                                 | 52 |
| 4.2.1     | Cloud API                                      | 53 |
| 4.2.2     | Agent API                                      | 54 |
| 4.3       | How does Scaling work?                         | 55 |
| 4.3.1     | Database layer                                 | 56 |
| 4.3.2     | Configurations Scripts store                   | 56 |
| 4.3.3     | API Service layer                              | 56 |
| 4.3.4     | Congestion Controller / Queue Management Layer | 56 |
| 4.3.5     | Front End Application                          | 57 |
| 4.3.6     | Analytics Engine                               | 58 |
| 4.4       | Sample Results                                 | 58 |
| 4.4.1     | How will Agent send data?                      | 59 |
| 4.4.2     | How will Cloud API receive data?               | 59 |
| 4.4.3     | How will Database save data?                   | 60 |
| 4.5       | Best Practices                                 | 61 |
| Chapter 5 | SYSTEM EVALUATION                              | 63 |
| 5.1       | How was the evaluation done?                   | 64 |
| 5.2       | Tools used for the evaluation                  | 65 |
| 5.2.1     | How K6 works?                                  | 65 |
| 5.2.2     | Sample Evaluation Results for API Performance  | 67 |
| 5.3       | Ability to Scale                               | 70 |



|   |    |
|---|----|
| 5.3.1 What is PM2?  | 70 |
| 5.3.2 Sample Evaluation Results in terms of Scaling ability | 71 |
| Chapter 6 CONCLUSION  | 77 |
| 6.1 Research Contributions                                  | 78 |
| 6.2 Research Limitations                                    | 78 |
| 6.3 Future Work   | 79 |
| REFERENCES  | 80 |

## LIST OF FIGURES

|   |    |
|---|----|
| Figure 2-1 : Architecture of Internet of things.....  | 14 |
| Figure 2-2 : Overall System Architecture.....   | 19 |
| Figure 2-3 : Graphical representation of security challenges in Internet-of-Things.....             | 30 |
| Figure 2-4 : Overview of the first PHP-based RESTful IoT Back-end prototype.....                    | 32 |
| Figure 2-5 : High level System Architecture of WSO2 IoT Server.....                                 | 35 |
| Figure 3-1 : High-level Architecture of the system .....  | 39 |
| Figure 3-2 : Modular Architecture of the API .....  | 45 |
| Figure 3-3 : Registration process .....   | 47 |
| Figure 4-1 : Components diagram of API.....   | 52 |
| Figure 4-2 : Source Structure of Cloud API .....  | 53 |
| Figure 4-3 : Source Structure of Agent API .....  | 54 |
| Figure 4-4 : Components level scaling architecture .....  | 55 |
| Figure 4-5 : LM-35 .....  | 58 |
| Figure 4-6 : Console logs when agent sends data to Cloud API.....                                   | 59 |
| Figure 4-7 : Console logs when Cloud API receives the same data .....                               | 60 |
| Figure 4-8 : Database snapshot of current data set .....  | 61 |
| Figure 5-1 : 10 Virtual users send request per second for 5 seconds.....                            | 66 |
| Figure 5-2 : 1000 Virtual users send request per second during 10 seconds for 4 times.              | 68 |
| Figure 5-3 : Average values for 1000 users , send requests during 10 Seconds for 4 times .....      | 69 |
| Figure 5-4 : PM2 shows a single node process is running in cluster mode .....                       | 71 |
| Figure 5-5 : HTOP shows how the CPU and the memory are utilized for single node process.....        | 71 |
| Figure 5-6 : Average values for 1000 users , send requests in 10 Seconds in a single instance ..... | 72 |
| Figure 5-7 : HTOP shows how the CPU and the memory are utilized for two node processes .....        | 72 |
| Figure 5-8 : PM2 shows that 2 node processes are running in cluster mode.....                       | 72 |
| Figure 5-9 : Average values for 1000 users, send requests during 10 Seconds in 2 instances.....     | 73 |
| Figure 5-10 : HTOP shows how the CPU and the memory are utilized for 4 node processes .....         | 73 |
| Figure 5-11 : PM2 shows that 4 node processes are running in cluster mode.....                      | 73 |
| Figure 5-12 : Average values for 1000 users , send requests during 10 Seconds in 4 instances.....   | 74 |

Figure 5-13 : HTOP shows how the CPU and the memory are utilized for 8 node processes ..... 74

Figure 5-14 : PM2 shows that 8 node processes are running in cluster mode..... 74

Figure 5-15 : Average values for 1000 users, send requests during 10 Seconds in 8 instances..... 75

Figure 5-16 : Sent and Received data variation against number of clusters..... 76

## LIST OF TABLES

|  |    |
|--|----|
| Table 3-1 Initially identified modules | 38 |
| Table 5-1: K6 Built in Matrices        | 67 |

## LIST OF ABBREVIATIONS

| <b>Abbreviation</b> | <b>Description</b>                         |
|---------------------|--|
| OEM                 | Original Equipment Manufacturer            |
| WWW                 | World Wide Web                             |
| IoT                 | Internet of Things                         |
| RFID                | Radio Frequency Identification             |
| ITU                 | International Telecommunication Union      |
| API                 | Application Programming Interfaces         |
| M2M                 | Machine to Machine                         |
| REST                | Representational State Transfer            |
| SDK                 | Software Development Kit                   |
| SLA                 | Service Level Agreement                    |
| HTTP                | Hypertext Transfer Protocol                |
| MQTT                | MQ Telemetry Transport                     |
| CoAP                | Constrained Application Protocol           |
| IOT-OAS             | IoT Open Architecture System               |
| ROM                 | Read Only Memory                           |
| XML                 | Extensible Markup Language                 |
| YAML                | Yet Another Markup Language                |
| JSON                | JavaScript Object Notation                 |
| URI                 | Uniform Resource Identifier                |
| IP                  | Internet Protocol                          |
| IDE                 | Integrated Development Environment         |
| LAN                 | Local Area Network                         |
| URL                 | Universal Resource Locator                 |
| XMPP                | Extensible Messaging and Presence Protocol |