Linnéuniversitetet
Institutionen för medieteknik

Master Thesis

A unified framework for real-time streaming and processing of IoT data

Author: Mohamad Zamam
Supervisor: Juwel Rana
Examiner: Aris Alissandrakis
Date: May 29th 2017
Subject: Social Media and Web Technologies
Level: Master
Course code: 5ME11E

To Hasan, Ghada, Rafeef, Jad and Julie

ABSTRACT

The emergence of the Internet of Things (IoT) is introducing a new era to the realm of computing and technology. The proliferation of sensors and actuators that are embedded in things enables these devices to understand the environments and respond accordingly more than ever before. Additionally, it opens the space to unlimited possibilities for building applications that turn this sensation into big benefits, and within various domains. From smart cities to smart transportation and smart environment and the list is quite long. However, this revolutionary spread of IoT devices and technologies rises big challenges. One major challenge is the diversity in IoT vendors that results in data heterogeneity. This research tackles this problem by developing a data management tool that normalizes IoT data. Another important challenge is the lack of practical IoT technology with low cost and low maintenance. That has often limited large-scale deployments and mainstream adoption. This work utilizes open-source data analytics in one unified IoT framework in order to address this challenge. What is more, billions of connected things are generating unprecedented amounts of data from which intelligence must be derived in real-time. This unified framework processes real-time streams of data from IoT. A questionnaire that involved participants with background knowledge in IoT was conducted in order to collect feedback about the proposed framework. The aspects of the framework were presented to the participants in a form of demonstration video describing the work that has been done. Finally, using the participants' feedback, the contribution of the developed framework to the IoT was discussed and presented.

Keywords*: Internet of Things; Real-time; Data Analytics; IoT Framework; Data Heterogeneity; IoT Data Streaming*

ACKNOWLEDGMENT

First, I would like to thank the Swedish Institute (SI) that gave me the opportunity to study in Sweden. Through the SI Scholarship, I was, fortunately, able to come to Sweden and pursue a Master's Degree. In fact, it is a unique experience that has changed my life forever. I would like also to thank the SI team that was always supportive and assisted me a lot during my studies.

In addition, I would like to thank the teachers at the Department of Social Media and Web Technologies, as well as the administrative staff for their support.

I would like also to thank the participants for their valuable feedback. This feedback was very important for evaluating the different aspects of my research.

I would like to thank Dr. Juwel Rana who supervised me during this thesis. Juwel was always helpful and assisted me to finish this work successfully. Through his advice and guidance, I was able to overcome difficulties and solve problems during the time of my thesis. Thank you a lot!

While these Master's studies approach to the end, I recall the moments with you Abraham Georgiadis. I also recall several projects we have done together. I could say I have learned a lot from you. In fact studying with you was always enjoyable, rather than being a boring duty. You are a true friend. Thanks!

And finally, without you my family, I couldn't have done all of this. You were the love, warmth and endless support. I will always be grateful for your existence in my life, and my success will be always yours. Thank you!

CONTENTS

LIST OF ABBREVIATIONS

ATM	automated teller machine
AWS	Amazon web services
CQL	contextual query language
DBMS	database management system
GPS	global positioning system
GSM	global system for mobile communication
IoT	internet of things
LAN	local area network
LTE	long-term evolution
M2M	Machine to machine
MEM	micro-electromechanical systems
NFC	near field communication
RDD	resilient distributed datasets
RFID	radio frequency identification
ROA	resource oriented architecture
SaaS	software as a service
SOA	service oriented architecture
SWE	sensor web enablement

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

> "*There are a lot of things around the world that are potentially knowable. When we will be able to capture them, to turn them into data, there will be a lot of applications and benefits. But we will discover that what we are capturing now is very close to zero from the available data*."
>
> (Kevin Ashton, 2016)

1.1 Background

The Internet of Things (IoT) is the evolving network of everyday objects that are able to connect and therefore be uniquely recognized, located and controlled, and seamlessly integrated into objects and environments around us anywhere on anytime. By analyzing data from observations, those objects can autonomously interact with other objects as well as people using reduced and sometimes no human intervention. Whether via a Radio-frequency identification (RFID), a sensor, or an actuator an IoT object can sense the surroundings and autonomously respond to conditions or situations. That is essentially contingent on an internetworked global space of software, frameworks, and services acting as a universal neural network that operates these devices and promises to be the future of the current internet.

With these words Nikola Tesla describes the future of Earth, when connectivity will be all around: "*When wireless* is perfectly applied the whole earth will be converted into a huge brain, which in fact it is, all things being particles of a real and rhythmic whole*" (Tesla, 1926). Surprisingly, the idea of connecting devices and computers through networks is not new. In fact, it has been around for decades since the existence of the Internet and computer networks. Many consider the Carnegie Mellon Coke Machine[1] to be the first "connected" device. The university students and staff used this "connectivity" to check the status of Coke availability in this machine. Since that time and day after day, more devices become connected to the Internet, and telecommunications advance in an unprecedented way forming the infrastructure of the modern IoT. Notably, only in 1999 the term "Internet of Things" started life by Kevin Ashton [6]. Since that time the IoT started to be used by the mainstream. Even though the term was coined at that point and started to be used publicly, however, the recent confluence of key technologies and market trends is promising in a new reality for the IoT [39].

A closer look at the IoT ecosystem shows three fundamental elements: sensors; connectivity; and people and processes [32]. Furthermore, the interaction between these entities is creating a new wave of smart applications and services. A smart thermostat can save heating by adapting to usage patterns and turning the temperature down when the user is not at home. An activity tracker can capture heart rate patterns continuously and send observations to the Cloud for medical data analysis. Moreover, using traffic sensors users can identify the real-

[1] https://www.cs.cmu.edu/~coke/history_long.txt

time availability of parking spaces on their phones. The list of IoT applications is quite long however, yet there are many challenges and issues hindering the wide adoption of IoT. The fact that different vendors are bringing devices to the market creates a lot of interoperability and heterogeneity issues. A device that is produced by one vendor utilizes unique specification and data format that vary from an IoT device produced by another vendor. According to [39], a fragmented environment of proprietary IoT technical implementations will inhibit value for users and industry. In addition, the use of generic, open, and widely available standards as technical building blocks for IoT devices and services will support greater user benefits, innovation, and economic opportunity. Accordingly, this research explores and further approaches IoT data heterogeneity through a unified IoT framework.
On the other hand, IoT is the "poster child" of Big Data where billions of connected devices generate and transmit enormous amounts of data from which intelligence should be derived, often in real-time [36]. Furthermore, the volumes and velocity nature of IoT data raise different challenges that should be addressed when designing a unified IoT framework. Low latency, durability, resiliency, and scalability are the utmost demanding needs. Nevertheless, the integration between various IoT resources and Big Data tools in one unified framework has got only a little attention. This research adopts a data-centric approach in designing the proposed IoT framework. At the same time, this framework provides real-time analytics for IoT data streams from smart-city sensors.

1.2 Motivation

It is astonishing how technologies in computing world advance exponentially. Within a decade or so, an emerging field like IoT receiving a lot of hype. However, it is not surprising all the fuss about IoT with the big potential that it promises. The global IoT market reached USD 598.2 Billion in 2015 and the market is expected to reach USD 724.2 Billion by 2023 [34][37]. Furthermore, Gartner, Inc. forecasts that 6.4 billion connected things will be in use worldwide in 2016, up 30 percent from 2015, and will reach 20.8 billion by 2020 [18][34]. Even though the variability in predictions makes these numbers questionable, however, collectively they paint a picture of a significant growth and influence [39]. It is such an enthusiasm for the researcher to be part of this wave that will impact economies, science, and societies. Furthermore, contributing to solving problems within IoT domain will be of a significant value both scientifically, academically and economically.

In addition, with many vendors only supporting their products through proprietary IoT frameworks, it will be a considerable effort developing a unified platform for the variety of IoT devices and services. This research is strongly driven by the need to develop practical IoT technology; as well as proper tools that provide real-time IoT data analytics.

1.3 Research Problem and Questions

The IoT vision provides a large set of opportunities to users, manufacturers, and companies [28]. However, the road is paved with challenges and hurdles, and we have yet to harvest the ultimate benefits from IoT-driven solutions.

According to [11], the phenomenal growth in smarter end-user devices and M2M connections is a clear indicator of the growth of IoT, which is bringing together people, processes, data,

and things to make networked connections more relevant and valuable. Additionally, M2M connections will grow globally from 780 million in 2016 to 3.3 billion by 2021. The IoT devices will have extremely heterogeneous means of connecting to the Internet, often with severe resource constraints [46]. Such a situation will only raise a new challenge to the IoT platforms because they constitute the medium that connects devices to people and processes. So far, many IoT solutions are being deployed in a vertical manner forming standalone systems that have been developed in ad-hoc domains. The recent trend, however, is to evolve towards a globally unified IoT platform [46]. Such a platform should introduce the proper tools that are able to handle the variety of IoT data formats in one place. This requirement introduces the *first* research problem.

Additionally, the current IoT needs for better control, monitoring and management in many areas, and the ongoing research in this field, have originated appearance and creation of multiple systems like smart-city, smart-grid and smart-home [15]. However, the expensive technology, as well as its novelty, have been always an impediment to IoT diffusion to the mainstream. For example, the average sensor message size formatted with RabbitMQ[2] for a TurtleBot[3] is roughly about 5 KB [24]. With an average of 50 messages per second and 10 devices, the cost for hosting this network on IBM Watson IoT platform[4] will be roughly 5,760 USD per month. Similarly, using Microsoft Azure IoT Hub[5], the same network will cost roughly 4440 USD per month. These costs imply another important challenge that bounds the wide adoption of IoT. Moreover, a look at Google search trends reveals a hype of searching for the term “Internet of Things” starting from 2013 (see Figure 1.1). It is only recently people came to be aware of IoT’s big potential. For instance, smart homes have been a dream for decades, but the lack of compelling user experiences and practical technology (e.g., low cost, easy-to-deploy, low maintenance, etc.) has often limited large-scale deployments and mainstream adoption [4]. These challenges impose the need for an affordable IoT technology. In this regard, open-source analytics tools stand as a viable solution to address these challenges. Through utilizing open-source tools and frameworks for an IoT unified platform, the problem of providing data analytics could potentially be solved along with enabling the technology for the public. This requirement introduces the *second* research problem.

[2] https://www.rabbitmq.com/

[3] http://www.turtlebot.com/

[4] https://www.ibm.com/internet-of-things/platform/pricing/

[5] https://azure.microsoft.com/en-us/pricing/details/iot-hub/

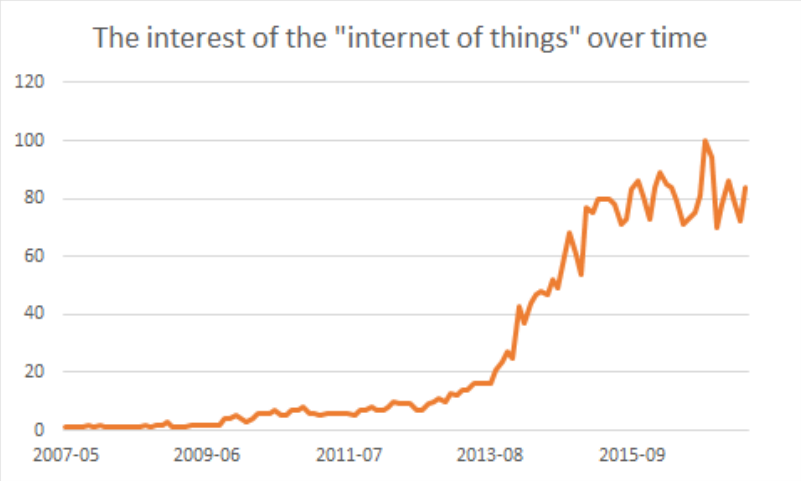


Figure 1.1. The public interest of the term “Internet of Things” according to Google search trend

It is indispensable, however, to make sure that a combination of IoT resources as well as open-source analytics tools will have an impact on the research within IoT domain. Moreover, it is fundamental to assess and value the shift that such a framework can have on the endeavors towards an efficient IoT deployment. Therefore, each single aspect that potentially contributes to IoT research was tested and put to validation.

To summarize, the above-mentioned two research problems can be formed in two main research questions:

- **RQ1**: *How to normalize heterogeneous data that are being generated by IOT devices from different vendors?*
- **RQ2**: *How to use existing open-source real-time analytics tools to develop a generic IOT framework?*

1.3 Thesis Outline

So far Chapter 1 introduced a generic overview of the IoT altogether with reflecting on the research problem and questions, while in Chapter 2 the most recent and related endeavors that tackle similar research problems are presented. Chapter 3 establishes a foundation for this research through providing basic definitions for key aspects such as the IoT framework and the ‘thing’ notion. In Chapter 4 a reflection on how the research problem was approached is presented through the research methodology. Chapter 5 analyses the trend in IoT devices as well as frameworks, along with providing an overview of two known IoT frameworks. The technical outcome of this work is thoroughly presented in Chapter 6 as a prototype of the proposed IoT framework. While in Chapter 7, the different technologies that were utilized are listed and discussed. Chapter 8 presents the results that revealed from this study in the form of a participants’ review. Chapter 9 discusses the results of the participants’ feedback collected in the form of a questionnaire. Chapter 10 addresses the contribution of this thesis to the research in IoT. Finally, Chapter 11 concludes with the challenges encountered, the limitations and the future directions of this research.

2 RELATED WORK

A lot of research has been conducted addressing the different aspects and challenges within IoT. Some of these efforts are taking place at the academia, while the next frameworks and platforms of the IoT are being developed by industrial consortia [14]. Nevertheless, only few research put the focus on solving data formats heterogeneity as well as approaching a unified IoT architecture that comprises advanced data management, analytics, and processing tools at the same time.

Many corporations are entering IoT market by introducing frameworks, software, and applications forming the IoT ecosystem. Within the industry, some IoT frameworks are Cloud-based solutions backed by large publicly traded companies as AWS IOT, Google Cloud IoT, Microsoft Azure IoT Suite and IBM Watson IoT [33]. While other frameworks are public and licensed as open-source software such as Kaa and macchina.io.

In the academia, some efforts addressed IoT frameworks from architectural perspective [20] [21]. While others approached mainly devices heterogeneity and data management [1] [23][45].

In [23], the researchers propose a framework for storing data from various IoT resources regardless of their structure or origin. Besides that, this framework unifies the access to these data through a RESTful API. The storage is a combination of traditional DBMSs and NoSQL databases that retain structured data. In addition, it utilizes the file system to save unstructured data through proposing a naming convention for each file corresponding to the source device and the observation timestamp. According to the researchers, this framework solves a series of challenges that IoT data face: the large volume of data; different data types; rapid data generation; and complicated requirements of data management. The question remains on how scalable is this framework especially due to performance variations between traditional databases and NoSQL databases. Moreover, using files to store data might produce a high latency that could be efficiently avoided through other storage mechanisms.

In [45], the researchers present a concept for an IoT framework the aimed mainly at solving the problem of smart objects' heterogeneity. Through introducing the virtual sensor concept, the researchers proposed virtual representations of IoT devices. With respect to this, applications follow two main concepts in order to use the most relevant connected objects; *cognition* and *proximity*. In this framework context, the *cognition* means applying best optimization techniques to find links between real-world objects (RWOs) and virtual objects (VOs). Furthermore, the *proximity* denotes the establishment of a level of relatedness between any IoT application and the relevant objects that may be used to deliver this application. A case study, which presents how the framework could be useful in a smart city scenario that horizontally spans several application domains was presented. This effort introduced an interesting concept on how to manage the interaction between smart objects from one hand of the framework, as well as interested applications and services on the other hand. However, it did not tackle the problem of identifying a smart object, such as the communication protocol to recognize an object. Additionally, the researchers did not describe how IoT data will be handled in terms of their formats and schemas.

The Global Sensor Networks (GSN) platform is introduced by [1]. The researchers presented

the concept of the virtual sensor as well, which is the key abstraction for their proposed platform. Each virtual sensor is provided with a description that facilitates the detection and the exploitation of this sensor. The virtual sensor can be a stream from a real sensor or from another virtual sensor. Using this approach, the platform allows for a pool of virtual sensors that could be consumed by applications and services. Even though this data-oriented framework allows for integrating a wide spectrum of heterogeneous sensor networks, however, it does not approach efficient data management and processing of sensory data with big volumes. That is mainly because Big Data technologies were at their infancy level and yet to reach their current advances at that point.

The work of [21] presented a unified framework for discovering devices and collecting observations through two kinds of web architectures and based on the Sensor Web Enablement (SWE) specification. A benchmarking of two architectures: Service-Oriented Architecture (SOA) and Resource-Oriented Architecture (ROA) was introduced; both when applied to an IoT framework. The proposed unified middleware is a web browser that enables discovery, exploitation, and management of sensors through the SWE specification. Even though this architecture allows for unified access to heterogeneous hardware platforms, it still lacks addressing the high demands of a unified IoT framework, since it is primarily based on the web browser. Such demands include scalability, performance and efficient data processing and accommodation tools.

The researchers in [20] proposed a conceptual framework integrating ubiquitous sensing devices and IoT applications. This framework is cloud-based at the center of an IoT system architecture. Using this envision different service providers can join the network and offer their data using a storage cloud; analytic tool developers can provide their software tools, and artificial intelligence participants can provide their data mining and machine learning tools. The implementation provides an IoT sensor data analytics SaaS through integrating Microsoft Azure cloud and Aneka. Aneka plays a role of an adapter for different Cloud infrastructures. From an architectural perspective, Cloud-based IoT platforms stand as a viable solution for scalability and efficient computation. However, they still need to address data heterogeneity through providing proper tools that integrate with cloud infrastructures.

Through this research, the proposed framework potentially handles most of IoT data formats while ensuring the scalable and efficient streaming of IoT data. Furthermore, it allows for applying novel data processing and visualization technologies to maximize the benefits of sensory data. Additionally, it provides an interface for applications and services to consume processed IoT data enabling the creation of modern IoT applications such as smart city, smart home, smart health, and smart environment.

In summary, it has been shown from this review the state-of-the-art of IoT frameworks in the market and academia. There are commercial IoT frameworks owned by big companies like Microsoft and IBM, however they are proprietary and they, as stated while demonstrating the research problem, are relatively expensive and not open for public development and research. Furthermore, it has been shown that some frameworks has been introduced in the academia, however still not completely solving IoT data heterogeneity issue and utilizing Big Data tools as will be followed by the proposed framework. The next chapter will establish a foundation for this research through introducing a set of core definitions.

3 DEFINITIONS

The following definitions help in understanding key components of the IoT as well as for this research specifically.

3.1 Thing

A “Thing” refers to any object, thing or device which embraces one or more sensors that harvest data from the surrounding environment and, it does not have to but might act or respond accordingly to these data. That is the simple definition. Whereas according to the Cluster of European research projects on IoT [41]; “Things” are active participants in business, information and social processes where they are enabled to interact and communicate among themselves and with the environment by exchanging data and information sensed about the environment, while reacting autonomously to the real/physical world events and influencing it by running processes that trigger actions and create services with or without direct human intervention. That is the broad definition. However, the two definitions usually refer to the same thing, the “Thing”.

3.2 IoT Framework

The IoT taxonomy in [20] described three basic elements for an IoT ecosystem. The last two elements are: (a) Middleware—on demand storage and computing tools for data analytics and (b) Presentation—novel easy to understand visualization and interpretation tools which can be widely accessed on different platforms. These two elements could be described as an IoT platform. That is referring to the middleware which provides storage and computing tools for IoT data analytics as well as novel easy to understand visualization and interpretation tools. However, that is a semantic-oriented vision of an IoT framework. It addresses the issues of IoT data management. Furthermore, a key capability that is enabled by this architecture is semantic interoperability and integration (i.e., across the sensor data from various sensors) [2]. This approach is adopted within the context of this research while designing the proposed framework.

Having established a foundation through core definitions of this research, the next chapter will present the adopted research methodology for tackling the research problem.

4 METHODOLOGY

In order to answer the three research questions that lead this thesis work, a prototype of an IoT framework was developed. The purpose was to examine how viable is to handle the different data formats coming from IoT in one tool. In addition, it would be possible to utilize open-source data analytics in order to analyze IoT data in real-time and get some insights, potentially. After the exploration of the domain and the development of the prototype, the next step was to have the prototype evaluated to see whether it answers the research questions.

A group of participants with background knowledge in IoT and Big Data were contacted in order to get their opinion regarding the developed prototype. The adopted research methodology was to quantify the participants' opinions about the prototype through conducting a questionnaire. This questionnaire had closed as well as open-ended questions. The aim was to get quantitative as well as qualitative data. Moreover, the questionnaire was an optimal solution that helped out in collecting feedback from the participants quickly with no additional costs that otherwise should be considered if the study participants will be reached out in order to get their feedback.

4.1 Prototyping

According to [35], prototyping is very suitable for gaining experience in new application areas and for supporting incremental or evolutionary software development. In addition, trying out a requirement is possible if a prototype of the system to be developed is established [27]. Following that the IoT is very suitable as a new area of research where prototyping could potentially solve arousing problems.

The developed prototype orchestrates IoT data with open-source processing and analytics tools in order to move IoT-driven deployment to a new paradigm. Through this endeavor, a vision of an IoT framework is proposed which accommodates the efficient handling of heterogeneous IoT data with capabilities of open-source data processing frameworks such as Apache Spark and Apache Kafka in order to maximize the value from IoT. It is worthy to mention that the prototype follows a data-centric approach (i.e., not supporting IoT devices exploration and management, as well as other requirements that should be considered in that sense). However, this feature for supporting devices could be a potential future extension for the proposed framework though. The focus therefore is mainly on enabling IoT data handling, visualization and analytics.

4.2 The Questionnaire

In order to get the participants' opinion about the developed prototype, a questionnaire of 14 questions was created. In addition, a 7-minutes video that demonstrates key aspects of this research is embedded into the questionnaire. Through that, the participants can evaluate the concept of this research in the form of answers to questions, either be closed or open-ended. The open-ended questions provide a free space for participants to enrich their feedback openly.

As seen from the group of questions (see Appendix A), the first goal is to quantify the level of IoT competence among the participants' group has, and therefore to check how credible is

their feedback. Depending on how much competence they hold, it is more likely that additional credence could be given to their answers. It was hoped that by exposing the participants to the demonstration they will gather sufficient knowledge which enables them to validate the different aspects of this research. What was presented through the demo is the research problem that this thesis addresses, how it was tackled, and what value this brings to the IoT. It is worthy to mention, however, that this approach is not the optimal choice in order for that participants to acquire decent knowledge about the framework and within such short period of time. Yet it was a viable choice within the limited time frame of this thesis, since other methods like interviews will require longer time of preparation and conduction.

4.3 Targeted Group

The participants' group is a combination of engineers working in IoT industry, as well as Ph.D. students at Linnaeus University with research interest either in IoT or Big Data. The reason behind involving people working with Big Data is because, as mentioned before, this framework is data-centered and therefore it tackles problems related to Big Data besides IoT. It is, therefore, possible that a participant with background knowledge in Big Data can still evaluate most of the aspects of this research.

Ten participants were contacted in order to invite them for participating in the questionnaire. In addition, they were given a brief idea about the main theme of this research through invitations. They have been told also what was required from them through each invitation.

5 IoT TREND ANALYSIS

In order to answer the research questions and to establish a foundation for developing the prototype, it was necessary to explore the domain and the trending technologies within. The reason for that is because IoT is still at an early level of development. Even though IoT started to bring value to corporations and consumers, however, there is quite an uncertainty about utilized concepts and tools. It was therefore of utmost importance to investigate key issues that touch the implementation of the prototype. Primarily, it was important to perform an overview of IoT devices from manuals of different vendors. That helped to shape an understanding of the variety of specifications and data formats used by each device. Furthermore, an overview of features for two trending IoT platforms was performed; these are Microsoft Azure IoT platform, and ThinkSpeak platform. MS Azure IoT is a commercial platform, while ThinkSpeak is free for usage. The overview was carried out through experimenting both platforms with same tools and data. The goal was to examine strengths and weaknesses of these frameworks, while at the same time, practically evaluating the actual need for a unified open-source platform that meets uncovered aspects in these platforms.

5.1 Trend Analysis of IoT Devices

IoT devices represent the sensing component that provides information about the device itself or the surrounding environment. Despite different purposes from IoT devices, in many forms they share one characteristic; that is transforming natural analog signals into digital ones representing data about different phenomena. According to [8], IoT devices refer to "*Things having identities and virtual personalities operating in smart spaces*" while [22] considers them as "*everyday objects that are readable, recognizable, locatable, addressable, and/or controllable via the Internet*". Though there is no concrete definition of what exactly an IoT device is, since it may describe everything that is potentially able to "connect".

To understand properly what an IoT framework is, as well as the specifications of the diverse IoT devices in the market, it was important to investigate current trends in IoT. This research stresses on establishing a systematic way for describing the IoT devices. That is, by characterizing them according to the main dimensions of the IoT. In order to do that, a matrix of IoT dimensions - IoT devices was created. Through this matrix, various examples of IoT devices along with their commonalities and distinctions are provided. In that sense, different IoT paradigms were reviewed [7][47]. The work in [7] presented the most common dimensions for designing IoT; namely mobility, connectivity, and cost. The list is extended here by combining other characteristics such as the family of an IoT device, the energy it consumes and the application domain and type. In below, these dimensions are briefly derived:

- *Category*: the proliferation of IoT-enabled objects created a demanding need for taxonomizing these objects. Some common categories include wearables, handhelds, electronics and home appliances.
- *Energy consumption*: energy comprises two parts: (1) source; (2) management. According to [16], the four types of power source for a connected object are: harvesting (the energy is gathered from the environment, e.g. solar panels), periodically recharged or replaced, non-replaceable primary source (the power source

determines the lifespan of the object) and remains-powered (the power source is virtually unlimited).

- *Application domain*: this comprises, but not limited to: smart home; smart retail; smart city; smart agriculture; smart environment; smart transportation; personal; military; and smart transportation.
- *Application type*: this describes examples of widely known applications of this device that brought value to business or industry (e.g. road tolls in smart transportation).
- *Connectivity*: this includes four main types [39]: (1) Device-to-Device; (2) Device-to-Cloud; (3) Device-to-Gateway; (4) Back-End Data-Sharing.
- *Mobility*: some objects are manufactured to become mobile (e.g. a pedometer) and others to be fixed (e.g. a thermometer) [16].

Some of IoT objects and devices along with their characteristics in relation to IoT dimensions are described in the Table 5.1.

	Category	Energy	Application Domain	Application Type	Connectivity	Mobility
Apple Watch Series 2	Wearables	Maximum 2 days battery	Personal	Workout, Activity, Maps, etc.	Device-to-Device/ Device-to-Cloud, Wi-Fi, GPS	Mobile
RFID	Sensors	Passive RFIDs with no batteries - energy transmitted by the reader	Smart City, Military, Transportation, Smart Retail	Road tolls, Building access, Inventory, Passports, Payment cards	Device-to-Device/ RFID reader connects through Wi-Fi, LAN, GPS	Usually attached to a mobile device: cellphone, product or card
Smartphone	Handhelds	Varies/ up to 19-20 hours of full usage	Personal	Communication, Connection, Personal information management	Device-to-Device/Device-to-Cloud / Cell, WiFi, Bluetooth, NFC,GPS	Mobile
EnOcean[6]	WSN/ sensors/ actuators	No Batteries/ Consum Light Energy, Movement, Temperature	Smart Home, Industry, Transportation, Logistics	Building automation systems	Wi-Fi	Fixed
Microelectro mechanical System (MEM)	Sensors	Obtained from the context device	Energy, Environment, Transportation, Industry, Health	Accelerometers in cars, MEMS gyroscopes in autonomous helicopters and planes, Bio-ME	Device-to-Device	Fixed (Micro-scale energy harvesting, Inkjet printers) Mobile (cars, planes)
Arduino	Board/ Sensors Assembly	5V either from a battery, USB charging or AC-to-DC adapter	Personal, Academy	Building hobbyists and academic applications	USB, Serial connection	Fixed mostly
GPS receiver	Sensor	Obtained from the context device	Military, Transportation, Industry, Energy	Automobile navigation system, Personal navigation assistant	Device-to-Cloud/ GSM, GPS, LTE, 4G LTE	Mobile

Table 5.1. IoT devices - dimensions matrix

[6] https://www.enocean.com/en/

5.2 Trend analysis of IoT Platforms

IoT platforms evolved recently to meet the rapid growth of the needs and requirements of IoT as a new emerging and promising field. The key driving factors for utilizing IoT platforms include scalability; integration of different hardware; reusability of resources as computing power; cloud storage and data management. Some of the widely known platforms are Google Cloud IoT, AWS IOT, Microsoft Azure IoT Suite, IBM Watson IoT, ThinkSpeak and macchina.io[7]. An overview of two IoT platforms is provided; Microsoft Azure IoT Suite which is a commercial closed-source platform on one hand, and ThinkSpeak which is an open-source free IoT platform on the other hand. The aim of this qualitative comparison is to demonstrate the usability aspect of each platform by applying a usage scenario on each. The example scenario is similarly applied on both platforms in terms of the used data and methodology. However, this overview does not delve in depth to address other aspects like performance, scalability, and adaptability to new sensors and specifications; but only to paint a generic picture of what is an IoT platform and how each IoT vendor is approaching it. The usage scenario is thoroughly described below with more details.

5.3 An Overview of Two IoT Platforms

The experiment scenario follows a tutorial example about Azure Data Analytics for Developers[8]. Each IoT platform in this scenario performs data storage and analysis. The IoT device, in this case, is an ATM machine that is used by customers to withdraw money. For each withdrawal operation, the ATM machine sends a transaction to the cloud in order to do some analytics job and to save the transaction. A virtual set of ATM machines is used which means that transactions are sent by a cluster of machines simultaneously. While sending transactions, some fraudulent withdrawal are simulated. That means two withdrawals from the same card but via two different ATM machines at a near or same time. Technically, that is not possible in real life because a person cannot physically use two different ATMs at the same time. Based on this proposal, and when two withdrawals from one card and through two different transactions are detected in nearly time, then the system has a fraudulent transaction. Accordingly, this fraudulent transaction must be notified to the bank staff in order for them to take the proper action. Figure 5.1 illustrates the experiment scenario, where ThinkSpeak and Microsoft Azure appear to receive IoT device data and perform the data analytics part.

[7] https://www.postscapes.com/internet-of-things-platforms/

[8] Azure Data Analytics for Developers

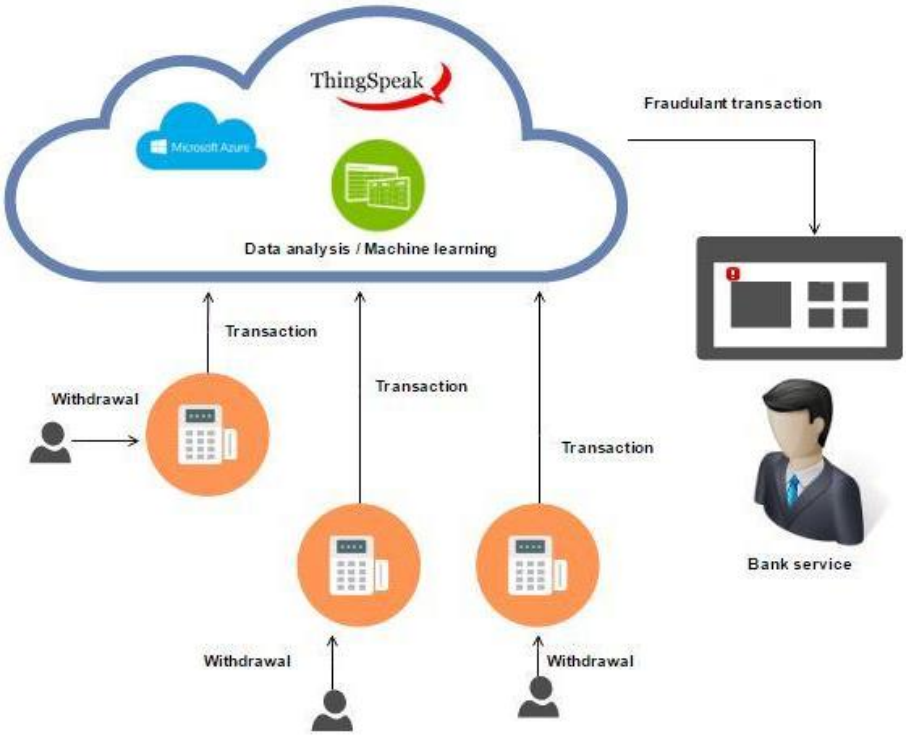


Figure 5.1. Experiment scenario for overviewing MS Azure and ThinkSpeak platforms

In order to simulate an ATM machine, a client web application was developed using Node.js. When executing this application it randomly generates fictitious withdrawals. A withdrawal from an ATM machine looks like the following:

```
{ "transactionId" : 1000, "transactionTime" : "Thu, 10 Sep 2015
14:36:34GMT", "deviceId" : 77996, "cardNumber" : 465881476, "Amount":160,
"EventProcessedUtcTime" : "2015-09-10T14:39:50.3289844Z", "PartitionId" :
3, "EventEnqueuedUtcTime" : "2015-09-10T14:36:34.1740000Z”  }
```

The different fields are generated randomly. A probability of 0.01 is fixed, which corresponds to the likelihood of a fraudulent withdrawal to happen. Each time a withdrawal occurs, this randomly generated number is compared with the probability number. If the randomly generated number is less than the probability number, only then the card number is saved and then a new withdrawal with this number and from a different machine is generated. This scenario was successfully applied on both platforms and data analytics were based on it. In the next two sections, the data analytics part through each platform is described. Furthermore, an emphasis on the usability aspect of each platform is put for both use cases. As a result, four essential questions were investigated for each platform: (I) how could the data be transferred or supplied to the platform? (II) How the data analytics could be triggered? (III) What data format(s) is to provide for each platform? (IV) And what technology is used to perform data analytics? These questions are answered thoroughly for each platform.

5.3.1 Microsoft Azure IoT Framework

MS Azure is a cloud service platform in which developers and IT professionals can build, deploy and manage applications[9]. With different capabilities that Azure provides, IoT applications can be run and hosted on the service. Some of these include creating and maintaining virtual machines, SQL databases, machine learning, stream analytics, etc. Users will have to create an account and login to MS Azure service in order to be able to create IoT applications, and the service is paid but a trial account could be used for testing purposes. Most of the experiment steps can be performed on the `manage.windowsazure.com` portal, and naturally after creating an account. Furthermore, all the instructions for this experiment are based on an MS Azure tutorial and could be followed on Microsoft Virtual Academy website[10].

Figure 5.2 illustrates MS Azure platform architecture. In this scenario, the ATM machines fall in the event producers' layer. Usually, this layer holds entities that produce events, those which meant to be applications or client IoT devices. In the example here they are represented by ATM machines. Event hubs represent a port where applications and IoT devices can subscribe to send data. Those event hubs can be then consumed by stream analytics jobs in order to process and analyze data. The output from a data analytics job can be directed to a service bus where different entities can subscribe: IoT devices and actuators to respond accordingly, or dashboards to visualize results for end users.

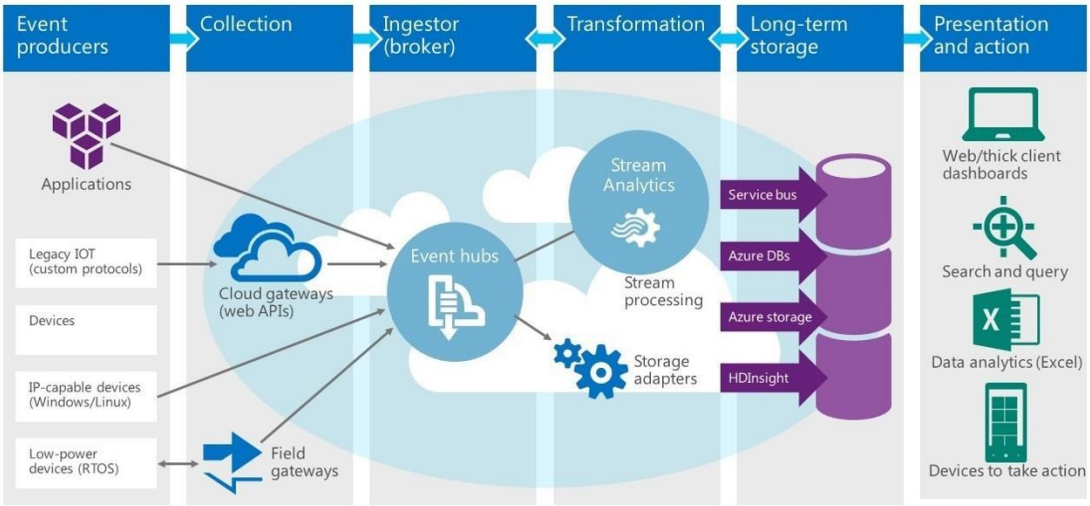


Figure 5.2. MS Azure IoT platform by J. Prosise[11]

After applying the above-mentioned scenario on Azure IoT platform, it was possible to answer the proposed questions. Those questions assisted in establishing a foundation for

[9] https://azure.microsoft.com/en-in/overview/what-is-azure/

[10] https://mva.microsoft.com/en-US/training-courses/Hands-On-with-Azure-Machine-Learning-16638?l=2oXJxvJrC_506218965

[11] From "Azure Data Analytics for Developers", by J. Prosise, Microsoft Virtual Academy (2016). Available at https://mva.microsoft.com/en-US/training-courses/azure-data-analytics-for-developers-16878?l=s0xrMjP9C_406218965

understanding what an IoT platform is and how it can be utilized:

How could data be sent to MS Azure platform?

The Azure platform provides an option to publish a web service for a stream analytics job. As a result, this web service can be accessed from any application by issuing simple HTTP requests. Having said this, any client application that could issue HTTP calls can use this Model As A Service (MaaS) and then provide data in different ways. One way to do that is to send data via real-time streaming.

Through creating an event hub on Azure platform, a URL for the service and a public key were obtained. Those then were used to create a distributed access signature, which enabled the proposed client to start triggering events and sending them to Azure event hub. Usually, events could be any time-based data obtained from a device. In this example, an event is a withdrawal from an ATM machine, while other examples could be a sensor's measurement or an observation from a wearable device.

How could data analytics be triggered on MS Azure platform?

Data analytics could be triggered by receiving any IoT event through an HTTP request. As mentioned above, events are received by Azure event hub. Moreover, Azure analytics job provides the ability to build queries against received data. In addition, it provides a windowing functionality such as the sliding window over a time series of data. This enables building queries such as: return the count of transactions each 5 minutes window.

What data formats does MS Azure platform accept?

Different data formats can be used to send data to MS Azure platform. However, the preferable data formats are JSON and tabular data. The client that is sending a data stream can post the data through an HTTP request as JSON format.

What technology is used to process and analyze data?

Using drag and drop, a data analyst can provide data sets and models in order to build a data analytics, e.g. a predictive model. The Azure platform provides models and libraries from different data analytics languages like R and Python. It also supports the ability to implement any customized script that a data analyst might have written with the above-mentioned programming languages. However, in the proposed scenario a normal SQL-like query is used to analyze the received data and detect fraudulent transactions. The query looks like the following:

```
SELECT W1.CardNumber as [CardNumber],
W1.DevicelD as [ATM1], W2.DevicelD as [ATM2],
W1.TransactionTime as [Timel], W2.TransactionTime as [Time2]
FROM Withdrawals W1 TIMESTAMP BY TransactionTime
JOIN Withdrawals W2 TIMESTAMP BY TransactionTime
ON W1.CardNumber = W2.CardNumber AND DATEDIFF(ss, W1, W2) BETWEEN 0 and 60
WHERE W1.DevicelD != W2.DevicelD
```

The query above detects any repeated usage of a card from two distinct ATMs and then triggers some action to the output data analytics hub in order to take the proper action.

5.3.2 ThingSpeak IoT Platform

ThingSpeak is an IoT platform that enables developers and IT professionals from collecting and storing sensory data in the cloud and develop IoT applications. The platform supports data analysis and visualization using MATLAB and then acting on the data[12]. The typical ThingSpeak workflow provides the following list:

- Create a channel and collect data
- Analyze and visualize the data
- Act on the data using several Apps

Figure 5.3 depicts a general architecture for the ThinkSpeak platform. As seen, the channel element is the part where ThinkSpeak receives data from any IoT source being a device, and application or a sensor. This client can subscribe to the channel and start sending data as a stream. If the channel is private, then only in that case the client should provide the channel private key when sending requests. The ThinkSpeak React component represents the part where analytics jobs can be triggered and scheduled. For example, this component can run a MATLAB code every fixed time interval. The result of a MATLAB analysis then can be provided to a ThingHTTP where different actions could be taken such as sending a tweet to a specific destination, triggering another ThinkSpeak App, or sending analysis result to a subscribed client.

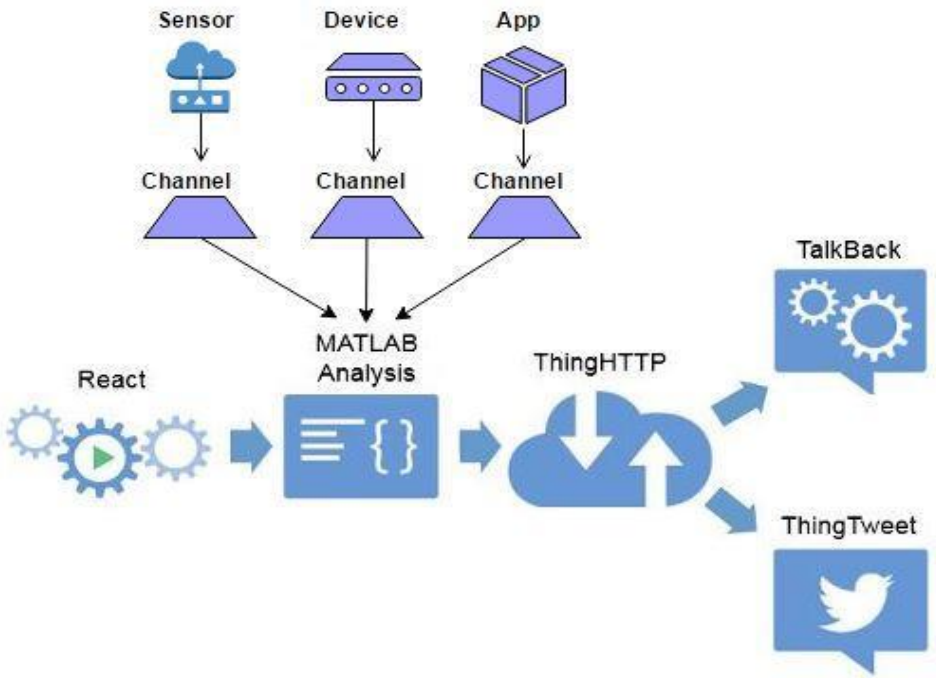


Figure 5.3. ThinkSpeak architecture

The above-mentioned scenario that was applied on MS Azure platform was similarly applied on the ThinkSpeak platform (i.e. same client application with simulated transactions). As a result, the same group of questions that were answered above will be answered for the

[12] https://se.mathworks.com/help/thingspeak/

ThinkSpeak platform:

How could data be sent to ThinkSpeak platform?

ThinkSpeak provides a functionality for creating channels. A channel could be public or private based on its owner's preference. Moreover, a channel can serve as a port for sending data to the cloud. Depending on channel's configuration, any device, website or service can access the channel in order to send data. A client can use normal HTTP requests to send data to ThinkSpeak channels. Furthermore, when sending data to ThinkSpeak a standard duration of 15 seconds should separate each request from the following. Once data is received by a ThinkSpeak channel, multiple actions could be triggered such as data analytics and visualizations.

How data analytics could be triggered on ThinkSpeak platform?

ThinkSpeak React could be used to trigger data analytics. This is like a scheduling application where users can select what action they want to schedule and on which basis. For instance, a data analytics code written with MATLAB could retrieve the highest temperature among the last 100 measurements of an IoT data stream. Furthermore, this code could be scheduled to run each 1 hour by ThinkSpeak React.

What data formats does ThinkSpeak platform accept?

Generally, IoT data could be tabular time-based series when sending to ThinkSpeak platform. JSON format is commonly used to send data to ThinkSpeak channels. In addition, a normal HTTP request is used to send data from any client such as an IoT device, a website or an application.

What technology is used to process and analyze data?

MATLAB code is primarily used to process received data. Algorithms, models and statistical methods could be applied on parts or full datasets. These technologies can be utilized to extract information and insights from channel's data and take an action accordingly. An example action could be taken when a sensor's observation exceeds a predefined threshold.

5.3.3 Reflecting On the Overview

The purpose from this overview was to show usability and implementation aspects from different platforms' perspective. This step contributed to understanding strengths and weaknesses of each platform. As a result, it was possible to understand how to combine best implementation aspects of each platform in order to establish a concept for a new unified IoT framework. Furthermore, a starting point was to enable customization of the new proposed framework and by that encourage developers' community to collaborate and improve the framework. For this reason, open source data analytics tools appeared to be an ideal solution. First of all, Big Data community from developers and engineers is quite familiar with these technologies which make it easy for them to adapt to a new framework that utilizes these tools. Furthermore, these tools are proven technologies and tested over a couple of years which makes them stable enough to handle the special requirements of IoT data, or per say, Big Data from IoT.

In summary, it has been revealed from this overview that a unified IoT framework should

meet the following requirements:

1. It should allow networks of devices to subscribe while ensuring scalability when hosting an enormous number of resources.
2. It should support real-time streaming of IoT data since the value of these data is maximized if obtained in real-time. Therefore, enabling practitioners to take proper actions immediately.
3. It should support resources' heterogeneity, thus enabling different data formats to be received, normalized and processed.
4. And last but not least, it should enable data analytics and visualization in an efficient way. Therefore, enable getting actionable insights from IoT data.

Through the presented overview above, MS Azure and ThinkSpeak both serve as the streaming and data analytics platforms for IoT. Even though there are several IoT platforms like these two, however the approach in this thesis is to develop a generic open-source platform that serve the abovementioned purposes, while at the same time, solve data heterogeneity problem. Based on this argument and the criteria that revealed from the above overview, there is a need for adopting different technologies that could integrate with each other to form this 'unified IoT framework'.

The first two requirements could be summarized by two principles; *subscription* and *streaming*. Concretely, a publish/subscribe system could ensure handling these two features. Such system enables IoT devices to subscribe and stream data for analytics. It should as well be an open-source system in order for it to achieve open-source requirement for the proposed framework. There are couple of systems that provide these two mentioned features: stream processing and open-sourcing. Apache Kafka, Apache Flink and Apache Storm are examples of these systems. In that sense, Apache Kafka was selected based on several aspects that are clarified later in Chapter 7.

On the other hand, Apache Kafka cannot handle different data formats coming from IoT; that is the third requirement. For this reason, it was necessary to add an extension tool that achieve this requirement, which is one of the core contributions of this research. The tool is developed with Java programming language due to platforms-independency of Java as well as the different supported connectors that it provides for many data analytics and streaming tools like Apache Kafka and Apache Spark. Chapter 6 gives more detailed overview about this developed tool.

The fourth requirement could be summarized by two principles; *data analytics* and *data visualization*. Data analytics require a set of operations like transformations, aggregations and MapReduce. Apache Spark was selected for this purpose for its features that are clarified more in depth in Chapter 7. Furthermore, in order to visualize data it was necessary to find a tool that provide an extension for Spark in Java and at the same time, offers rich visualizations. The combination of Elasticsearch with Kibana dashboard was an optimal choice for this purpose due to several reasons illustrated in Chapter 7 as well. The next chapter expands on the implementation aspects of the prototype using the abovementioned technologies and how they integrate together, however, it is necessary first to explain that a dataset is needed in order to test an IoT framework. It is also necessary to explain the motivation behind selecting a specific dataset, in this thesis, the CityPulse project dataset. That will be presented at the introduction of Chapter 6.

6 PROTOTYPE IMPLEMENTATION

As discussed in Chapter 5, there was a selection of two IoT frameworks for overviewing the principle of an IoT framework through a practical example. That example demonstrated a set of ATM machines that utilizes these two frameworks through a developed application that serves as a client sending IoT data. In addition, the dataset that was used is a simulated one for testing and was generated by the application itself. Nevertheless, when it comes to developing an IoT framework, it is essential to test it against real IoT data. The reason behind that is because of the data volumes, diversity and velocity of real IoT data are essential factors that cannot be obtained when using generated datasets. These factors are necessary to show the validity of the developed framework when used later in real life deployments of IoT networks of devices. What was done here to do that is using the abovementioned application in the prototype to simulate sending IoT data. However, this time a real dataset from the City of Aarhus in Denmark was used along with the application instead of the generated one.

In summarize, the developed client application in the overview scenario in Chapter 5 was extended in the prototype development to handle the new data set (i.e., to simulate streaming the dataset, which posed new requirements such as storing data and retrieving them using MongoDB). Additionally, a data handling tool was added to address data heterogeneity, that is, another component in the developed prototype. Finally, the open source tools; Apache Kafka, Apache Spark, Elasticsearch and Kibana represent the open-source data analytics tools and the final component of the developed prototype. The following paragraphs will dig in depth into the implementation aspects; how the data set was selected, cleaned and stored, and how the different tools were integrated together in order to deliver the final prototype, the unified IoT framework.

6.1 Observing the City Of Aarhus in Denmark

One essential reason for selecting the smart city applications for testing IoT frameworks is because of the diverse and rich IoT resources that exist in urban areas. Examples of these resources include smart buildings, vehicles and facilities. Another reason for experimenting with data from smart cities is the availability. For example, several projects can be found online that handle applications for big cities as the Array of Things project[13] in the City of Chicago in the USA, and the CityPulse project[14] in the City of Aarhus in Denmark. Moreover, in order for the stakeholders of these projects to encourage collaboration and creativity, they engage the public into these project through crowd-sourcing. The data, as well as the collective resources and information from these projects, are opened to the public contribution. According to the Array of Things project team, "*Because the data will be published openly and without charge, it will also support the development of innovative applications*" [44]. Therefore, the project executes partnerships with academic institutions, companies and corporations. In addition, full data catalogue from the City of Chicago is publicly available[15]. The catalogue includes data from buildings, facilities, transportation and

[13] https://arrayofthings.github.io/

[14] http://www.ict-citypulse.eu/page/

[15] https://data.cityofchicago.org/

community.

The developed prototype is tested through applying on a dataset collected from the City of Aarhus in Denmark. The basic characteristics of this dataset will be further discussed below.

6.2 The Citypulse Project Dataset

The aim of this project is to provide smart city applications by providing data analytics and IoT data streams processing. Several papers were written as well as tools created within the context of this project and utilizing this open data set. The work in [3] [9] [26] [42] provides examples of conducted research on these data. The part of this dataset that covers two months - August and September 2014 was retrieved from the online catalogue. The reason behind the interest in this part is mainly because it comprises concurrent observations from four different phenomena: traffic, pollution, weather and parking. The selection of this dataset enables data from different phenomena to be correlated when studied and the influence of each phenomenon on the other could be examined and might lead to interesting results. Table 6.1 illustrates the sub-datasets of the CityPulse dataset collection over the year 2014. As mentioned before, the interest is mainly in the duration between August and September 2014 due to the data availability from the four types during this period. Each one of the four datasets will be briefly described below.

Description	Duration											
	2014											
	1	2	3	4	5	6	7	8	9	10	11	12
Road Traffic Data		Road Traffic Dataset-1						Road Traffic Dataset-2		Road Traffic Dataset-3		
Pollution Data								Pollution Dataset-1				
Weather Data		Weather Dataset-1						Weather Dataset-2				
Parking Data					Parking Dataset-1							

Table 6.1. City of Aarhus datasets collection. Note: reprinted from CityPulse Database Collection, by the researcher, retrieved from http://iot.ee.surrey.ac.uk:8080/datasets.html.

6.2.1 Traffic Sub-dataset

This sub-dataset represents a set of real observations collected by a group of entities that monitor the traffic status at different locations in the city. Each entity (observation point) is one or more sensors counting vehicles at a specific place. The sensors are distributed all over the city either at landmarks or intersections of major streets. The collection of observation

points comprises 449 points around the city. Within a duration of two months, each point generates and sends about 17000 traffic observations - approximately 1 observation each 5 minutes. Table 6.2 shows the formatting of each traffic observation as well as an example of a real observation.

status	avg-Measured-Time	avgSpeed	extID	median-Measured-Time	TIMESTAMP	vehicle-Count	_id	REPORT_ID
OK	74	50	668	74	2014-08-01T07:50:00	5	20746220	158324

Table 6.2. A traffic observation from the City of Aarhus

6.2.2 Pollution Sub-dataset

The pollution sub-dataset represents a collection of measurements that is manually generated in order to complement the traffic dataset. The dataset simulates one sensor for each of the traffic sensors mentioned above, and at the exact location of the traffic sensor [12]. The source of this dataset provides more information on how it was generated[16]. Furthermore, the collection consists of 449 points that generate pollution observations. Table 6.3 illustrates the formatting of each pollution observation as well as an example of one generated observation.

ozone	particullate_matter	carbon_monoxide	sulfure_dioxide	nitrogen_dioxide	longitude	latitude	timestamp
101	94	49	44	87	10.10499	56.23172	2014-08-01 00:05:00

Table 6.3. A pollution observation from the City of Aarhus

6.2.3 Weather Sub-dataset

The weather sub-dataset provides a collection of observations for the city climate, though not in a specific location but all over the city. Each observation is a [timestamp, value] pair. The six main elements that are observed: Dew point in degrees Celsius; humidity (percentage); pressure in millibars (mBar); temperature in degrees Celsius; wind direction in degrees; and wind speed in kilometers per hour (kph). Each observation point that monitors one of these elements generates about 70 weather observations per day. Table 6.4 shows three sequenced observations of the dew point in the City of Aarhus.

[16] http://iot.ee.surrey.ac.uk:8080/datasets/pollution/readme.txt

Observation 1	"2014-08-01T01:00:00": "13"
Observation 2	"2014-08-01T13:00:00": "15"
Observation 3	"2014-08-01T01:20:00": "13.0"

Table 6.4. Three sequenced observations for the dew point from the City of Aarhus

6.2.4 Parking Sub-dataset

The parking sub-dataset is a data stream representing parking occupation data from the city of Aarhus. There is a total of 8 parking lots providing information over a period of 6 months (55.264 data points in total) [12]. Each observation point at a parking lot monitors the lot occupation and sends observations to the IoT hub, every 30 minutes approximately. The observations provide useful information for the city tenants about parking availability. Moreover, and within the context of the CityPulse project, the dataset is utilized for building an application that suggests available nearby parking in real-time. Table 6.5 shows an observation for parking occupancy from the City of Aarhus.

vehiclecount	updatetime	_id	totalspaces	garagecode	streamtime
0	09:04.1	1	65	NORREPORT	11/3/2014 16:18

Table 6.5. A parking occupancy observation from the City of Aarhus

6.3 Processing IoT Data: Creating Data Streams from the Static Dataset

6.3.1 The Motivation behind Streaming IoT Data

Even though batch processing systems like Apache Hadoop have been optimal for Big Data processing recently, however, new demands are arising and should be considered. For example, in the case of data coming from IoT, these demands like real-time queries and data streams cannot be met efficiently by batch-based frameworks like Hadoop [40]. In fact, it is now indispensable for data management systems and Cloud computing to support data streaming. The need for an alternative to the traditional way of exchanging data between the client and the server is behind the emergence of messaging and streaming technologies. Furthermore, streaming data reduces data processing costs, especially when dealing with enormous amounts of events generated by many resources like the IoT. In addition, it enables applications and processes to react in real-time to events rather than waiting a long time for a large batch job or a transaction in a traditional DBMS. For instance, the pollution dataset that is mentioned earlier in the City of Aarhus dataset will take hours only to be stored in a NoSQL database like MongoDB. Additionally, when streaming these data the latency of an analytics job applied on this stream is significantly improved compared to a traditional batch job. An analytics job that aims at calculating the average of pollution measurements will have very low latency if applied on a streaming window function rather than applying on a batch job in a traditional database. The above-mentioned reasons motivate the transformation of the

example dataset from a static one into a stream. Having said that, storing this dataset in MongoDB is primarily for simulating the streaming process of IoT events as if they happen in real-time.

6.3.2 Processing Datasets

Each sub-dataset from the City of Aarhus dataset was scanned in order to be saved in MongoDB. The purpose as mentioned before is to retrieve these data and stream them as they were originally generated. Four datasets were scanned, processed and saved: traffic; pollution; weather and parking. The selected period was from the 1st of August until the 30th of September, 2014. Additionally, because these datasets are time series data, each time point that happens to have an event was treated as an index on a timeline over the above-mentioned two months. It means that each timestamp from an observation was indexed first, and then those events that share this timestamp were grouped by it. As a result, the events count at each time point varies depending on how many observations were recorded by the sensors of the four observed phenomena. However, a typical time point on the timeline has at least one observation in order to exist and thereby to be recorded in the database. The most useful time point is the one that has simultaneous observations from every sensor in the city (i.e., the one that gives maximum insights though correlating observations). Figure 6.1 illustrates the timestamp points of the City of Aarhus dataset represented as a timeline. As seen from the figure some timestamps have events from the four phenomena that the dataset has, while others might have one event only. Particularly, since each stream type is produced by a group of sensors that are sending observations from different locations in the city, then each timestamp will have at least one set of observations from all points of a specific phenomenon (i.e., all observations from the 449 traffic sensors at specific timestamp for instance).

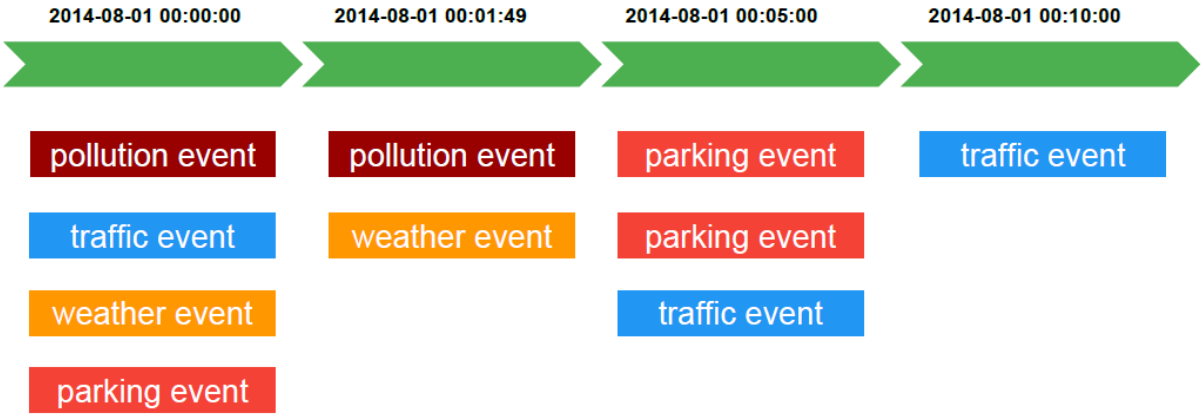


Figure 6.1. The timestamp points of the City of Aarhus dataset represented as a timeline with events

6.3.3 Database Schema

MongoDB is used to store and retrieve the events for streaming. A document that corresponds to a time point might have one or more observations as subdocuments. An application is used to parse each file in the example dataset and analyze the timeline in order to index observations. It is worthy to mention here that such a process consumes a lot of resources and has a very high cost in a way that requires hours of processing when executing on a local machine. The high cost of I/O operations in traditional databases is one reason for

processing IoT data as streams. An optimal way is by applying queries on data streams in near real-time or real-time.

Figure 6.2 shows the MongoDB scheme of the time points' collection. It also shows the structure of each sub document that represents an IoT observation. After storing the dataset, a collection of 21778 documents were created. Each document corresponded to a time point, and the average was one observation every 3 minutes. The average size of each document is 120kb yielded a total dataset size of 2.5GB over 2 months. Such big volumes of data represent one challenge in dealing with data coming from IoT devices.

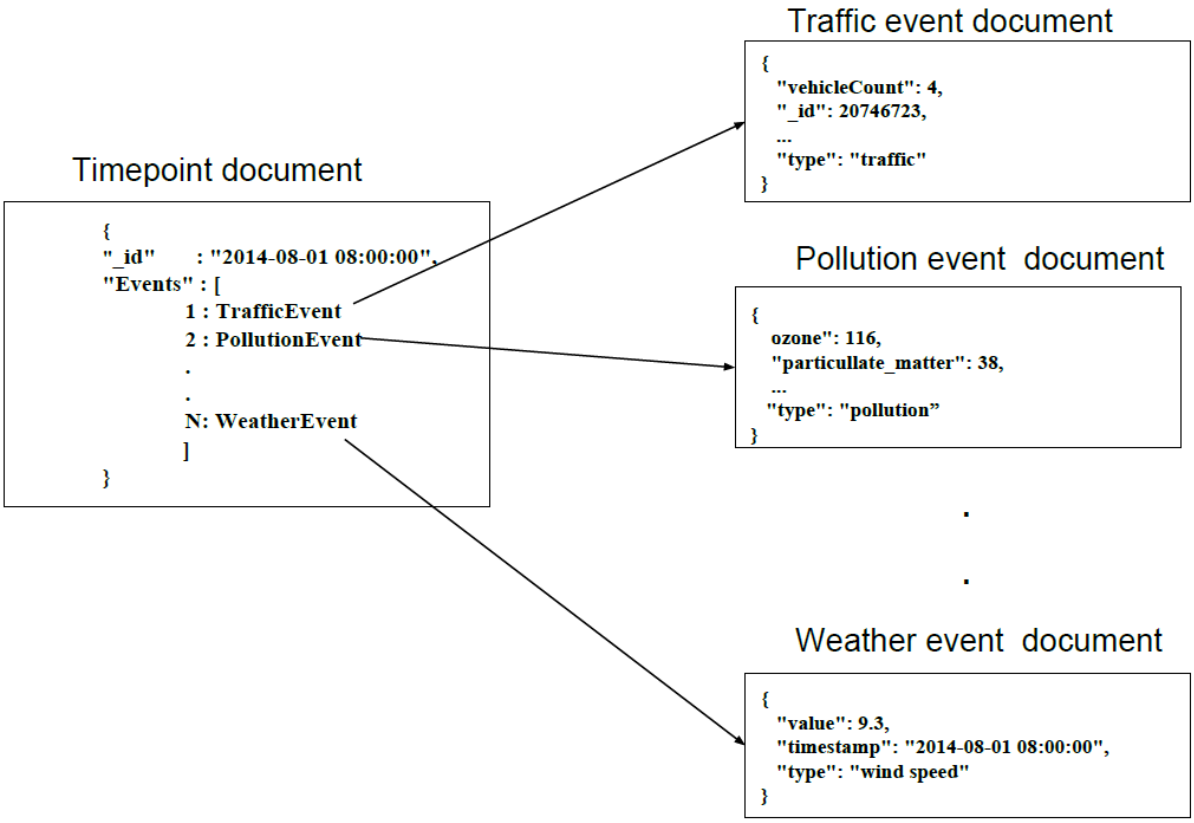


Figure 6.2. MongoDB schema of the time points collection with events in each point

6.4 Framework Architecture

In order to simulate the data streaming from IoT devices, a streaming simulator is one developed entity in the proposed architecture. This component is equivalent to an IoT deployment of smart devices that monitor environments and send data to the handling tool.

From a high level perspective, and taking data management complexity into consideration, three essential requirements for an IoT system architecture should be considered. First, in order to turn data into useful information and to ensure interoperability among different applications, it is necessary to provide adequate and standardized formats, models and semantic description of data content (meta-data), using well defined languages and formats [28]. In the proposed architecture this is accomplished through the data handling tool. Second, a middleware that assures data storage and analytics is required [20]. In order to address this requirement, open-source data management tools are used such as Apache Kafka, Apache Spark, and Cassandra database. Third, a presentation layer should be considered.

That is, a novel easy to understand visualization and interpretation tools which can be widely accessed on different platforms [20]. In order to achieve this requirement, the proposed architecture utilizes Elasticsearch and Kibana.

Figure 6.3 dissects the architecture of different components that integrate together constituting the proposed IoT framework. These components communicate and exchange data in a variety of formats. The streaming simulator receives data from MongoDB as JSON format. Then each sub-dataset is formatted into a known IoT format and streamed to the data handling tool. This latter in turn listens to any coming IoT events with a predefined list of known IoT formats. Finally, each event is serialized and sent to Apache Kafka.

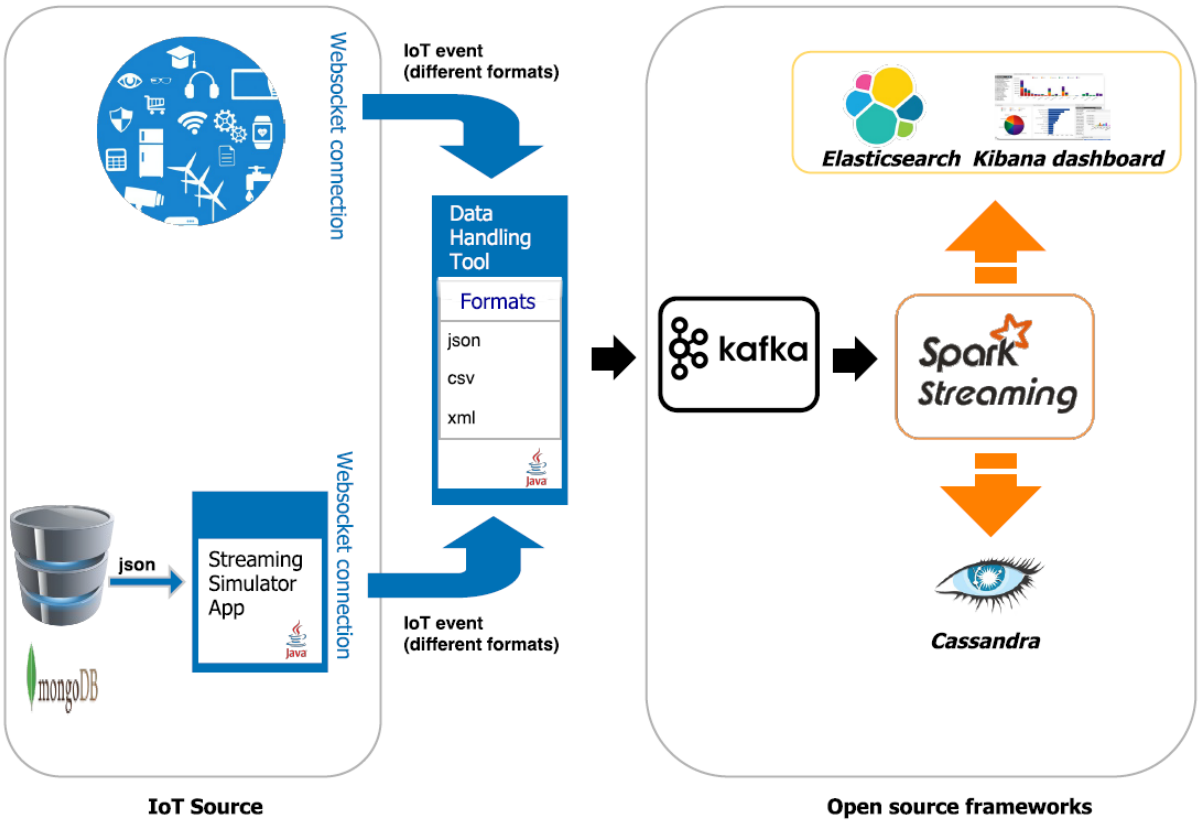


Figure 6.3. The IoT framework architecture

Kafka holds received messages within topics. Each sub-dataset is published to a certain topic in Kafka. Consequently, these topics could be consumed by Spark Streaming in order to apply necessary transformations, machine learning, and analytics on streamed data.

Spark pushes processed streams to Elasticsearch which is mainly used to apply real-time queries on data streams and also to build Kibana visualizations on top. However, Elasticsearch is not meant to retain data for a long-term storage and retrieval. Instead, Cassandra database is used for this purpose.

6.4.1 The Streaming Simulator

The streaming simulator is a Java application that fetches IoT observations from MongoDB and sends them to Kafka. Figure 6.4 depicts the process of simulating IoT events streaming. In order to perform this process, a time pointer slides on the time range that bounds the

example dataset (i.e., between “2014:08:01 00:00:00” and “2014:09:30 59:59:59”). Each sliding step on the timeline is one second since the time component of every timestamp is formatted as “HH:mm:ss”. While the time pointer progresses, another thread fetches events ahead from MongoDB every five minutes interval. These fetched events are the ones with timestamps between current time pointer and five minutes ahead. By doing that, it is guaranteed that the events queue is always having events to pull from. Furthermore, when the event on top of the queue matches the time pointer, the event is then sent to Kafka. However, the event is formatted with an arbitrary format that the data handling tool accepts. The idea behind this approach is to simulate IoT objects sending data to the data handling tool in different formats. While in real life each object will instead send its own format that the vendor will be providing. Each object, as well, should include the format in the message in order for its message to be realized by the data handling tool.

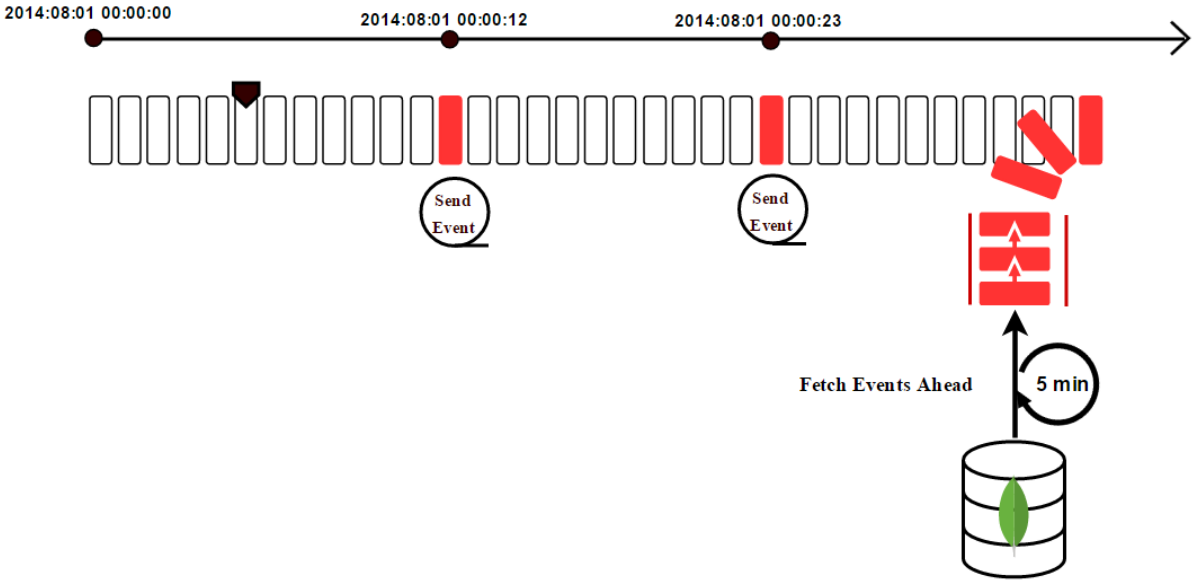


Figure 6.4. The process of simulating IoT events streaming

The communication between the client (either a real sensor or the streaming simulator) and the server (the data handling tool) is established through a web socket connection. The security of this connection could be handled by following an OAuth authorization methodology, however, it is not implemented in this prototype.

6.4.2 The IoT Data Handling Tool

The list of accepted data formats by this tool is already defined in a configuration file. Therefore, any new IoT format that might come to the industry by a new vendor can be added to this list. However, the tool should know in fact about the schema of each observation in order to be able to parse the received observation. The schema means in this context the structure of an observation and the data type of each field in the observation. For example, one possible specification for the weather observation in the City of Aarhus dataset might follow the SesnsorML format (see Appendix B for an example). This format follows The Observations and Measurements XML Implementation standard [29]. Other examples of

complex observations and sensors' standards with entity specification and ontology can be found elsewhere [13][30][38]. These specifications are helpful for discovering IoT objects automatically and additionally for collecting observations. Indeed, the issue of managing IoT devices by the proposed framework is out of this research scope and it is not intended to be fulfilled here. Instead, the main focus is on received data formats considering that observations are already realized and turned into raw formatted data (i.e., extracted through their schemes).

The data handling tool extracts observations based on the format as well as the observation's schema. The result is an object that is serialized and sent to Apache Kafka. Kafka holds logs of data as partitions per topic. Furthermore, for each data stream of the example data set, the data handling tool automatically creates a topic in Kafka. Thus, four topics initially are created when running the tool: traffic; pollution; weather and parking. Moreover, for each data stream, a Kafka producer is created and this producer is responsible for serializing observations and sending them to Kafka as JSON formatted data. The resulted topics are able to be consumed by different applications that have connectors with Kafka, thus the possibilities are unlimited when consuming IoT data from this tool. Having said that, not only the Spark application can consume IoT topics from Kafka, but any other application that has a connector to Kafka and interested in IoT data, since Kafka provides this scalability feature. As a result, all data will be finally formatted in JSON when written to Kafka topics and therefore unified for processing.

6.4.3 The Consumer Application: The Data Analytics Layer

The primary goal of using Spark as a consumer of IoT data is to process data and make them ready for utilization. It is meant by utilization to create visualizations; get insights; and trigger commands to things or actuators in semi real-time. The data processing comprises different techniques such as machine learning, streaming, aggregations and transformations. The main features that have been used through Spark are MapReduce, streaming, aggregations and transformations. In order to address the powerful features of Spark when it comes to processing Big Data from IoT, a thorough example of using data streaming and transformations is provided. This example is derived from the City of Aarhus dataset. Two data streams were aggregated and processed in order to produce a third data stream that contains mainly geopoints with some observations for each point. Through Kafka-Spark streaming library, the topics from Kafka are streamed as Resilient Distributed Datasets (RDDs) to Spark. Using Spark MapReduce in this driven example, two data streams - pollution and traffic are mapped to a third stream. The latter holds events that share timestamp as a key with both observations from the pollution data stream and the traffic data stream as a value. Table 6.6 shows an example RDD from the resulted stream.

```
{
      "timestamp": "2014-08-01 08:00:00",
      "ozone": 116,
      "particullate_matter": 38,
      "carbon_monoxide": 65,
      "sulfure_dioxide": 27,
      "nitrogen_dioxide": 35,
      "longitude": 10.104986076057457,
      "latitude": 56.23172069428216,
      "avgMeasuredTime": 61,
      "avgSpeed": 60,
      "extID": 668,
      "medianMeasuredTime": 61,
      "vehicle_count": 4,
      "_id": 20746723,
      "REPORT_ID": 158324
}
```


Table 6.6. An RDD from the mapped pollution-traffic stream into a new stream

The next step in this example is to reduce the above-mentioned stream into geopoints with the corresponding observations, while at the same time updating these observations for each geopoint. This is achieved through the State[17] function of Spark. For each geopoint, the stored state is always the latest observation at that point. Therefore, this state is updated every time a new observation is received from Kafka topic. While there might be a tiny delay until the observation arrives from the sensor, processed, sent to Kafka, and finally arrived to Spark consumer; nevertheless this totally happens in nearly real-time. At the end, the resulted stream holds the 449 geopoints with updated state about the pollution as well as the traffic observations at each point. Table 6.7 illustrates the state of the 449 geopoints at the timestamp [2014-08-01 08:00:00].

geopoint	location	timestamp	carbon_monoxide	. . .	vehicle_count
geopoint1	[56.23172069428216, 10.104986076057457]	2014-08-01 08:00:00	65	. . .	4
geopoint2	[56.22579478256016, 10.116589665412903]	2014-08-01 08:00:00	49	. . .	5
		.....			
geopoint449	[56.22579478256016, 10.116589665412903]	2014-08-01 08:00:00	75	. . .	0

Table 6.7. The state of the 449 geopoints at the timestamp [2014-08-01 08:00:00]

The whole state of the above table will change at the next time point, thus changing the value of every field (i.e., a column in the table). The stream that produces this table is saved to Elasticsearch as an index. Accordingly, this index is updated while observations are being

[17] https://spark.apache.org/docs/1.6.2/api/java/org/apache/spark/streaming/StateSpec.html

streamed. The index is used then to build Kibana visualizations based on streams' updated data. It is worthy to mention that this table is treated as an RDD object by Spark. With respect to this, Spark engine holds objects of this type in memory rather than the file system, besides distributing the computing overhead between cluster nodes. This means a very low latency which solves many problems in real-time based analytics by achieving an immediate response to unforeseen changes in data.

6.4.4 Data Analytics Dashboard

Through using mappings, each processed IoT data stream from Spark application corresponds to an Elasticsearch index. This means that the index is always updated with stream data from Spark. Furthermore, using Kibana dashboard a data analyst could apply live queries on data streams as well as observe changes in data through the Discover tab of Kibana dashboard. Figure 6.5 shows weather data after indexing in Elasticsearch through the Discover function of Kibana dashboard.

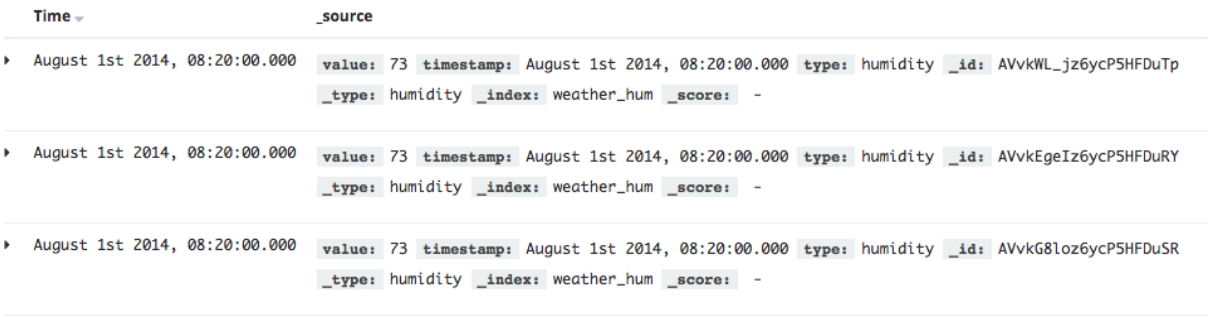


Figure 6.5. Indexed stream of weather observations into Elasticsearch

Table 6.8 shows an example of retrieving all traffic points where the surrounding level of carbon monoxide in the atmosphere is between 70 and 100. These queries can be very useful in data analytics in order to extract useful information from live streams. Even more, analysts can save these queries and create visualizations based on them.

```
GET traffic_points/points/_search
{
  "query": {
        "range" : {
            "carbon_monoxide" : {
                "gte" : 70,
                "lte" : 100
            }
        }
    }
}
```

Table 6.8. An example query applied on geopoints stream in Elasticsearch

Kibana enables building different dashboards for analytics at the same time. An example data analytics dashboard that was built comprises four visualizations of sensors' observations from the City of Aarhus. These observations represent traffic, pollution, humidity and dew point. Figure 6.6 shows a real-time view of the variation over time for traffic congestion and

carbon monoxide levels. As seen from these charts, there is some correlation between the observed level of traffic at a specific geographic location and the rate of pollution at that location. This could be useful to suggest healthiest routes for pedestrians in the city, as well as the healthiest times for walking.

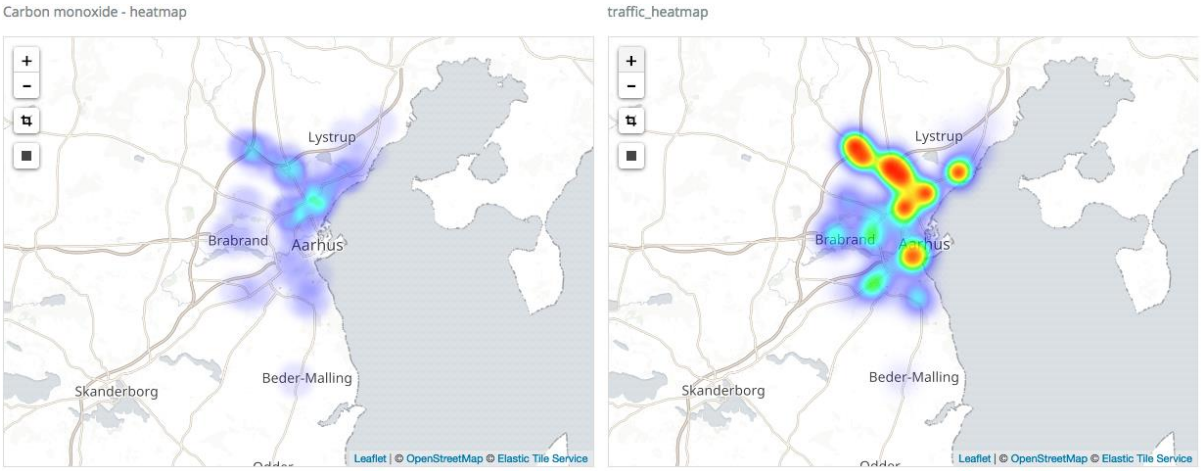


Figure 6.6. Kibana dashboard - The variation over time in carbon monoxide and traffic levels

Another two visualizations represent humidity and dew point levels in the city. This would allow for providing micro-climate forecasts if efficient machine learning algorithms are applied to data streams. Figure 6.7 depicts two metric charts for real-time observations of humidity and dew points levels, as well as statistical values for each phenomenon like the average, minimum and maximum observation.

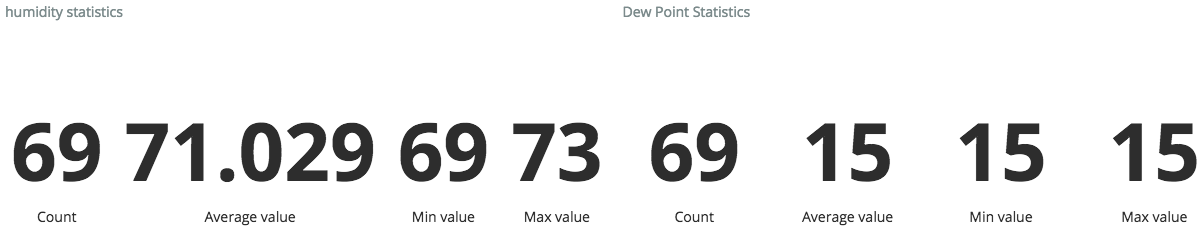


Figure 6.7. Kibana metric charts for real-time observations of humidity and dew points

As been illustrated before in this Chapter, a real dataset with different formats was used to feed data to the prototype. This dataset has real observations that were formatted with JSON, XML, CSV and TSV. When the data is received by the data handling tool, it is serialized and sent to Kafka topics in one format which is JSON. This means that whatever the received IoT format is, the output will always be formatted in JSON. The concept of using the data handling tool was to normalize heterogeneous IoT data; that is the first research problem in this thesis. Furthermore, by utilizing the open-source analytics tools it was demonstrated how these different tools integrate together to address the second research problem; how to develop a unified IoT framework using open-source analytics tools. However, Chapter 8 will expand more on how the research questions were answered by the implementation of the prototype as well as the feedback from the participants group. However, Chapter 7 will give a detailed overview of the different technologies that were applied in the prototype development.

7 TECHNOLOGY

Brief tips for practitioners and researchers are provided on how to setup each used open-source tool on MS Windows platform (see Appendix C). The main reason behind that is due to some technical challenges (see Chapter 11) that were encountered at the beginning of this prototype development. Even though the documentation for each tool is well-written, however, there was always a missing point that delayed the installation of one tool or another, thus delaying the development process. In order to facilitate the road for those who will be interested in this research, some guidelines for installation and integration are presented for each tool. A good example that illustrates installation tips of the different used tools can be found in [10].

7.1 Apache Spark

The IoT as a network of millions of devices communicating and collecting observations produces big volumes of data streamed via networks to the Cloud. The complex requirements of these Big Data streams as building interactive queries should be taken into account when designing IoT frameworks. In that sense, in-memory computing stands out as a notable solution that can handle both real-time and stream requirements. Spark is a good example for this case which supports in-memory computing using RDDs, real-time and interactive querying using Shark, and stream processing using fast micro-batching [40].

In addition, in the cluster mode, the load is balanced between nodes so producing efficient computations and data processing. The reason for using Spark in the proposed IoT framework is due to the variety of data processing components that Spark provides, as well as their thorough documentation and big community. Some of these components include data streaming, transformations and MapReduce. Some other technologies that are powerful for data processing include machine learning and graphs, though they are not used within this framework. The installation tips could be found in Appendix B.

7.2 Apache Kafka

The publish/subscribe (pub/sub) paradigm is designed to deliver events from a publishing source to interested clients in an asynchronous way. The pub/sub systems are very interesting solutions because of their decoupling properties, which means that interacting parties do not need to know each other [31]. A pub/sub system that supports data streaming enables the parallel and asynchronous communication between different entities and through that, increasing scalability, flexibility, and reliability of data management systems. Apache Kafka is an efficient messaging system that abstracts away the details of files and gives a cleaner abstraction of log or event data as a stream of messages. This allows for lower-latency processing and easier support for multiple data sources and distributed data consumption [5]. In the proposed framework, Apache Kafka is used for saving IoT streams as topics. Each topic is a set of partitions that are sequences of immutable records. This means that IoT events are saved in the chronological order of their arrival to Kafka. It is important therefore to maintain the original timestamp of each observation in order to identify the real sequence of observations. Through using Kafka, multiple applications can consume IoT topics independently. The Spark application in the proposed framework is just one consumer of

multiple that could benefit from streamed IoT data. Additionally, processed IoT data could be also pushed again to Kafka and consumed again. That is, like creating pipelines of topics inside Kafka.

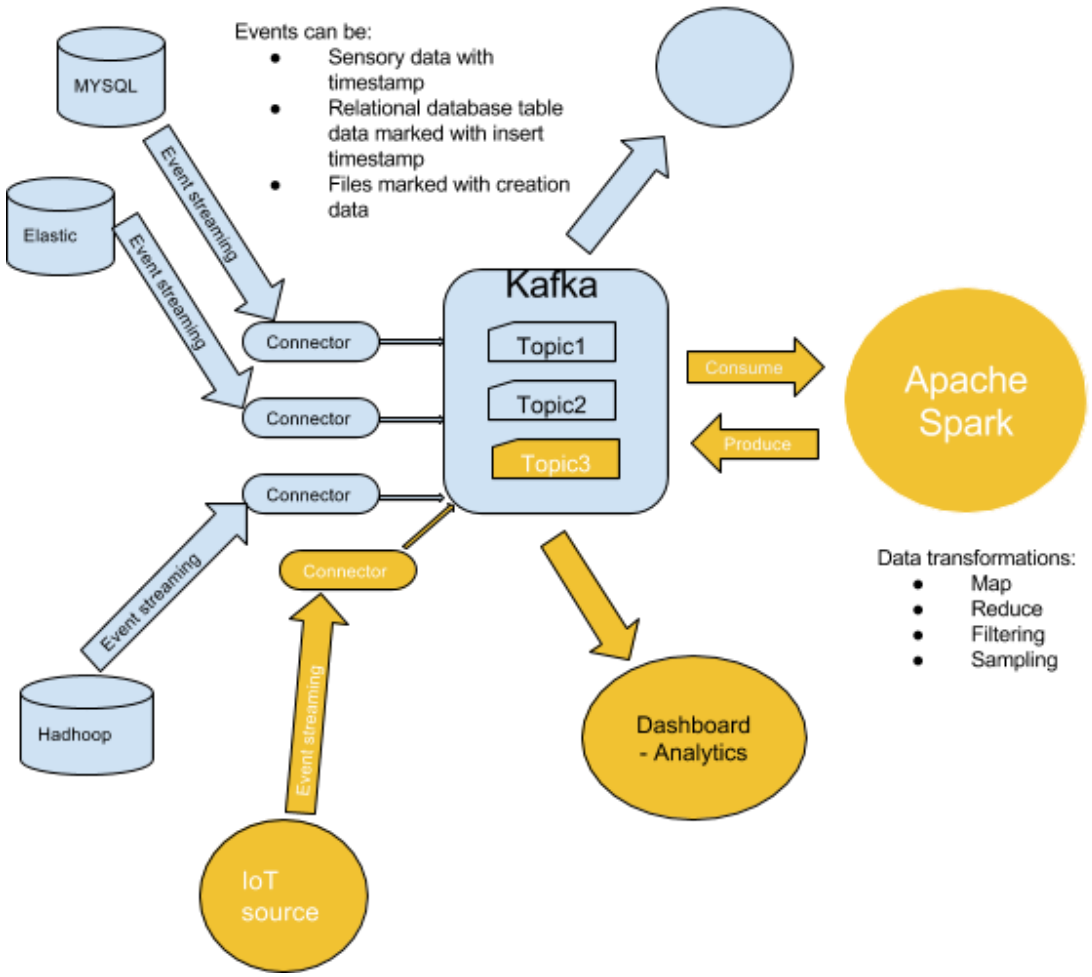


Figure 7.1. The Spark-Kafka anatomy as part of a complete data management solution

Figure 7.1 depicts how Spark and Kafka integrate in order to process and stream data in a data management solution. Spark is the producer and consumer in this example, while one topic in Kafka results from Spark processing and streaming of another topic. For instance, in the example of the City of Aarhus dataset, one topic could be every observation from a parking lot sensor. While another topic might hold every fully-occupied parking lot in the city.

7.3 Elasticsearch

Elasticsearch is a highly scalable open-source full-text search and analytics engine. It allows for storing, searching, and analyzing big volumes of data quickly and in near real-time [19]. The main reason behind using Elasticsearch is primarily to benefit from its Kibana dashboard. Elasticsearch provides a flexible and easy to use query language for exploring data. Furthermore, queries can be translated to charts and visualizations. For example, from a stream of pollution observations, this can be narrowed down to all observations that exceed a threshold. Even more, this resulted query could also be narrowed down to get all those that

match the first condition but fall within a specific geographic location in the city. All of that with a simple JSON-like query language. Elasticsearch uses *indices* to group documents. The document is called a *type*. For instance, in the City of Aarhus dataset example, an index named IoT is created which contains a type called "pollution". This type holds all processed pollution observations. In order to visualize different streamed data to Elasticsearch, Kibana dashboard is used. However, some data mappings should be considered before visualizing data streams. For example, Elasticsearch should recognizes a field that contains the pair (latitude, longitude) as a geolocation point through mappings. Similarly, that applies to a field that is formatted like "yyyy-MM-dd HH:mm:ss" as a timestamp. The timestamp field is needed when building aggregations on data streams such as windowing functions. In addition, the geopoint field is needed when building Tile Maps with Kibana dashboard.

7.4 Kibana

One big challenge for visualizing IoT data is that they are geo-related and are sparsely distributed. To cope with this challenge, a framework based on Internet GIS is required [20]. Kibana dashboard is used for this purpose. Kibana provides a high-level query language of data streams from Elasticsearch, along with the most recent and novel visualization technologies. Kibana is a window for Elasticsearch that enables visual exploration and real-time analysis of data stored there. It allows for combining different types of visualizations in one or multiple dashboards. Using this feature, different IoT resources could be monitored in one place. Furthermore, Kibana enables building different dashboards for a variety of purposes. One dashboard could deal with maps that display geographic areas overlaid with circles keyed to the data. While another dashboard could deal with data tables and metrics that show statistics of data as the count, average, min and max values. One limitation with Kibana, however, is that it enables only visualizing aggregated data but not the actual values. For instance, it is not possible to plot the variation of a specific series of sensor measurements over time. Instead, these observations should be visualized as a window of the average measurements each fixed window time (i.e., an average of observations each, e.g., 5 min window). Sometimes this is not expressive enough to show the actual change of an observed phenomenon over time.

7.5 Apache Cassandra

Apache Cassandra is a scalable fault-tolerant database system that is mainly suitable for large scale datasets. It can be run on commodity hardware as well as on the Cloud and supports replicating data across multiple datacenters. Cassandra is an optimal database solution for storing time-series datasets like IoT datasets. The main reason for that is because it retains each row with a corresponding timestamp which is an ideal approach for time-series data. This results in a fast indexing mechanism and efficient queries against the database. In the proposed framework, Cassandra is utilized for retaining IoT data for later usage where revising the history of a specific phenomenon might be needed in the future. Additionally, Cassandra uses CQL query language. It is similar to SQL, that is, easy to understand and to apply against data. Moreover, Cassandra supports drivers for many languages (Java driver is used in the proposed framework).

7.6 MongoDB

MongoDB is an open-source, document database designed for ease of development and scaling. Using MongoDB reduces a lot of complexity when it comes to creating schemas, storing and retrieving data. Besides that, the connectors for programming languages are pretty easy to follow and use. That makes MongoDB very suitable for quick implementation and experimenting with data. In the example of the City of Aarhus dataset, the whole dataset is ingested in MongoDB in order to be streamed. After testing MongoDB with the proposed framework, it showed a very low latency in retrieving data which was vital for simulating the IOT observations as in real-time. As mentioned earlier, roughly 21778 documents were created from the ingested dataset, where MongoDB still ensuring a high performance at that level.

7.7 Java

For all developed modules within the proposed framework, Java programming language was used for implementation. The motivation behind using Java is due to its support for many data processing frameworks and tools. Furthermore, and based on researcher's personal experience, Java is more straightforward and easier to implement compared to Python and Scala. For instance, using Intellij IDE reduces a lot of the overhead related to assembling dependencies, compiling and running Java programs.

This chapter has demonstrated different technologies that were applied in the implementation of the prototype. The next Chapter presents the findings of this study.

8 RESULTS

The proposed framework was applied on a smart-city scenario in order to test how IoT data will be streamed, and furthermore, how the different integrated components function in one unified framework. It was possible through this implementation to stream real IoT data to the framework. The purpose was to mimic real devices sending observations of different phenomena from the City of Aarhus. As discussed above in Chapter 6, the framework successfully showed real-time streams of data through its analytics dashboard. Furthermore, stakeholders and practitioners with practical knowledge within the streamed data could possibly infer useful insights from visualized data. However, this was not investigated due to the limited feasible time of this thesis. It will be interesting though to involve data analytics in a more in-depth user study where those analysts can use the prototype in practice.

In addition, it is important as discussed before to apply the framework on an IoT scenario such as smart-city applications. Practically, this requires a real or simulated dataset in order to be fulfilled. Having said that, researchers are compelled to select open data from the Internet for the mentioned purpose, thanks to their availability. However, this data belongs to different corporations, institutions and organizations that reaching out their users is not feasible sometimes. An example from this research is the CityPulse Project that provides the data sample which this study utilizes. It was not practical due to the time frame getting people from the project or the City of Aarhus in order to conduct a user study. Instead, an alternative solution was to get participants' review through demonstrating the framework to them and getting their feedback via a questionnaire.

As mentioned before, in order to explain the concepts of this research to the participants, a 7-minutes video was provided into the questionnaire. This video described the problems being tackled such as data heterogeneity and vendor-driven platforms and how these problems were approached. In addition, it demonstrated IoT data being streamed through the different tools until they are processed and visualized. As a result, after watching the video a participant can get an adequate knowledge about the overall content of this research and would be accurately able to give her feedback.

In total, 7 participants gave their feedback through answering the questionnaire. Below the results are thoroughly presented.

8.1 Results from the Questionnaire

The goal of the first question was to make sure that the participant has watched the video before proceeding to answer the rest of the questions. Therefore, if the participant answers with yes she will be allowed to continue answering other questions, otherwise, she will be forwarded to the end of the questionnaire.

It was essential as well to quantify the capacity of IoT knowledge among the participants. Even though they were selected and contacted on the basis that they are participants in the domain, however, it is likely that they are more knowledgeable about their competence. Furthermore, because some of them were data participants it was important to approximate their ratio compared to the whole group. Through this, it could be clarified how reliable was the feedback. As seen from Figure 8.1 the level of IoT competence within the participants' group is generally moderate (4 participants), while some reported a competent level (1

participant). However, some reported a low level of competence (2 participant).

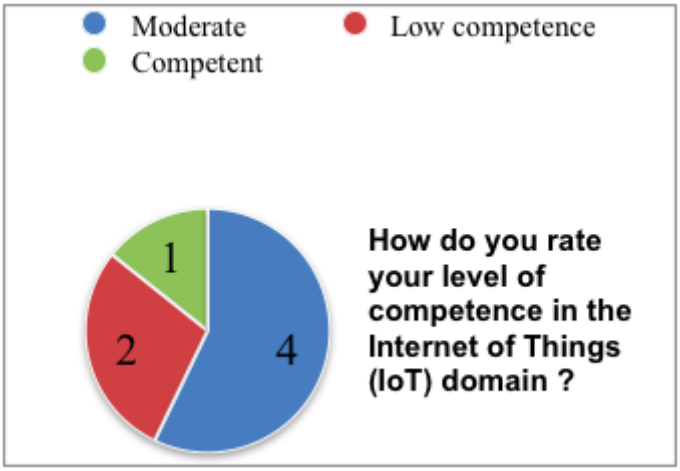


Figure 8.1. The level of IoT competence among the participants

Another goal from the questionnaire was to see at which level the participants will be convinced about the validity of this research. As seen from Figure 8.2 some were neither convinced nor unconvinced (1 participant), while others were somewhat convinced (2 participants), however, the majority reported that they were very convinced (4 participants).

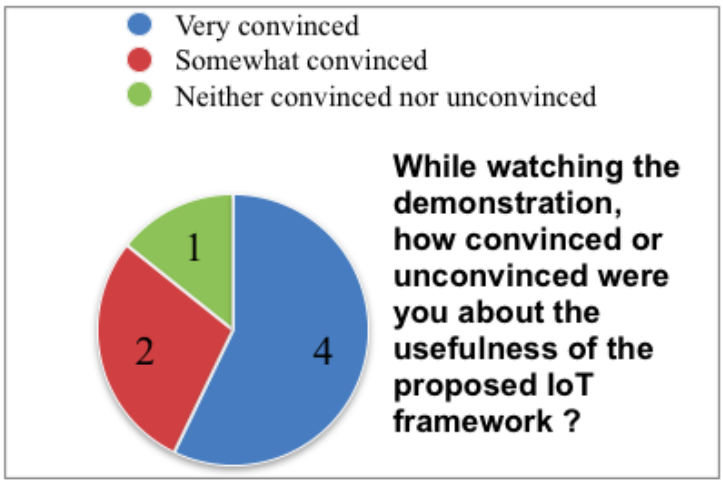


Figure 8.2. The level of conviction among the participants about the proposed framework

Another aspect that was validated is the value brought by the proposed framework compared to commercial frameworks since it is open-sourced. Concretely, the value proposition relies on reducing costs for corporations and consumers by using this open-source framework. As seen from Figure 8.3 the majority of participants (5 participants) reported that they somewhat agree on this value. While 1 participant reported that he strongly agree on this value. However, 1 participant reported that he neither agree nor disagree on this value.

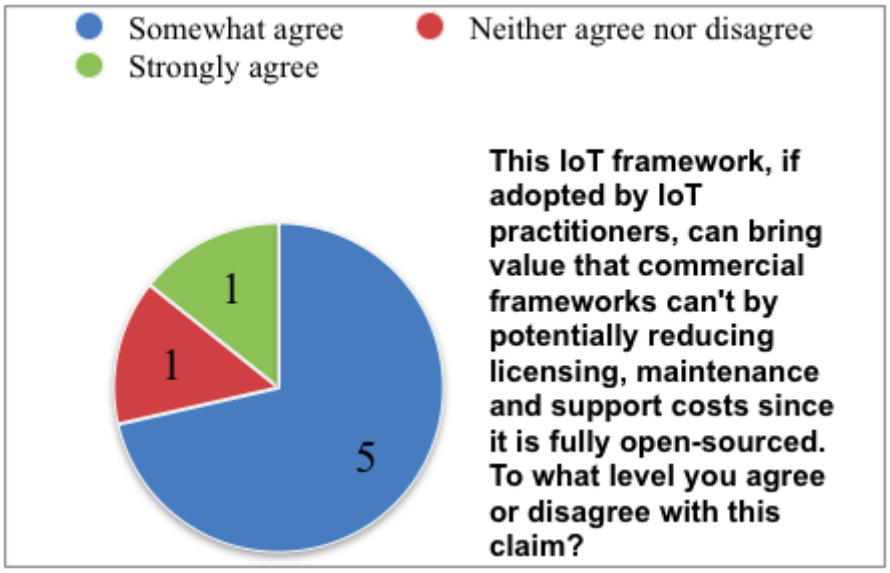


Figure 8.3. The participants' level of agreement on the value of open-sourcing the proposed framework

With respect to solving IoT data heterogeneity problem, and enabling vendors to customize the framework based on their new formats, the participants expressed a positive attitude. As seen from Figure 8.4 the majority (5 participants) reported an acceptable level of agreement on the value brought through solving data heterogeneity problem. Furthermore, some (2 participants) showed strong agreement on that aspect.

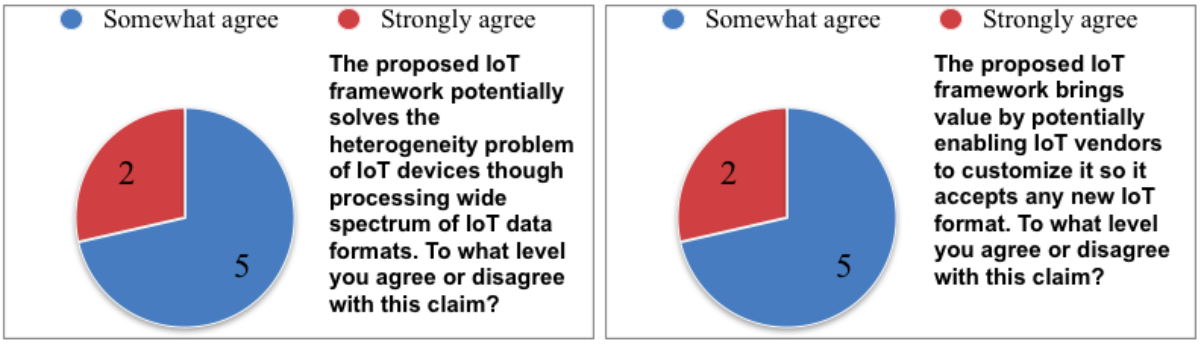


Figure 8.4. The participants' level of agreement on the concept of handling IoT data heterogeneity

The different factors that potentially give credence to the proposed framework are presented to the participants. As seen from Figure 8.5, they reported a high level of credence regarding two factors: open-sourcing and customizability. While for other factors as the free updates and the community support, they reported a moderate level of importance. However, GNU Licensing received a low level of importance in the participants' opinion.

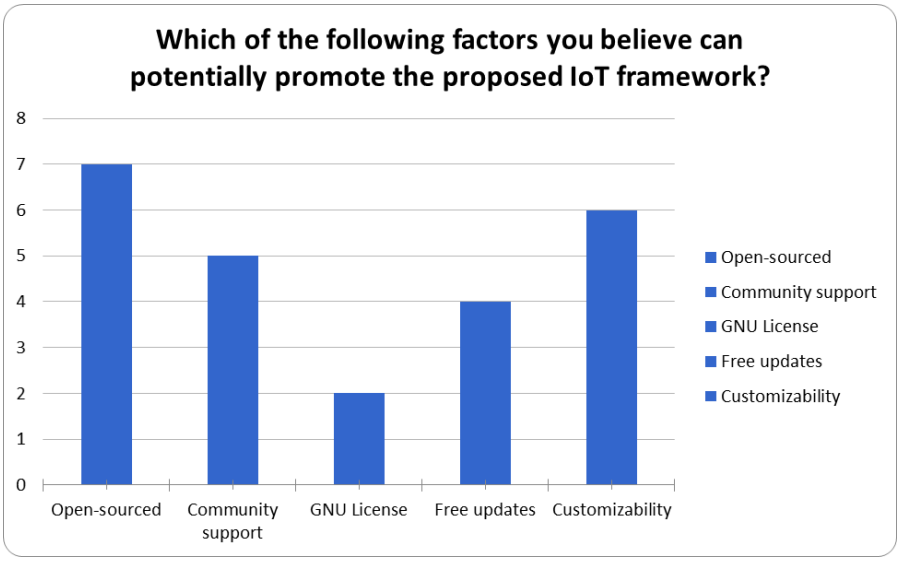


Figure 8.5. The participants' level of credence to the different factors that could promote the proposed framework

As seen from Figure 8.6, 5 participants reported that they didn't know about any other tool that simulates IoT data streaming. However, some pointed out that they knew about two other tools, namely Apache Apex and Kafka.

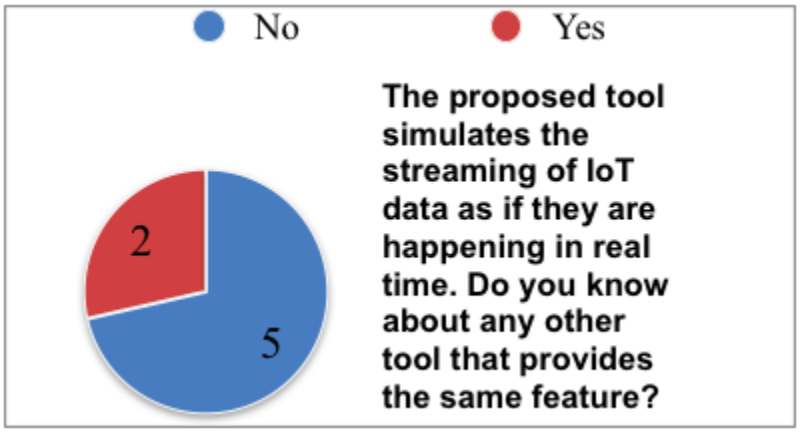


Figure 8.6. The participants' familiarity about potential tools that simulate IoT data streaming

With respect to the benefit of streaming historical IoT data, the majority of participants (6 participants) found this concept to be useful (see Figure 8.7). Nevertheless, some (1 participant) reported a moderate opinion as it might or might not be of that importance.

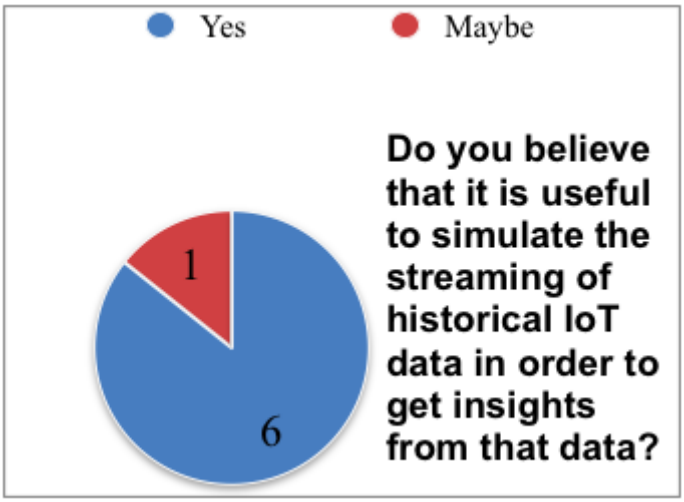


Figure 8.7. The participants' opinion regarding the usefulness of streaming historical IoT data

Regarding the open-source tools for data analytics, the participants' opinions were slightly varied. As seen from Figure 8.8 Apache Kafka has received high level of recommendation; while 4 participants highly recommended it, 2 participants normally recommended it and 1 participant expressed a moderate level. Furthermore, Apache Spark received a high level of recommendation as well. Three participants highly recommended Spark, while 3 participants normally recommended and 1 participant reported moderate recommendation. Furthermore, Elasticsearch and Kibana got an acceptable level of recommendation. Three participants reported high level of recommendation for both, however, 3 participants showed uncertainty about recommending both tools.

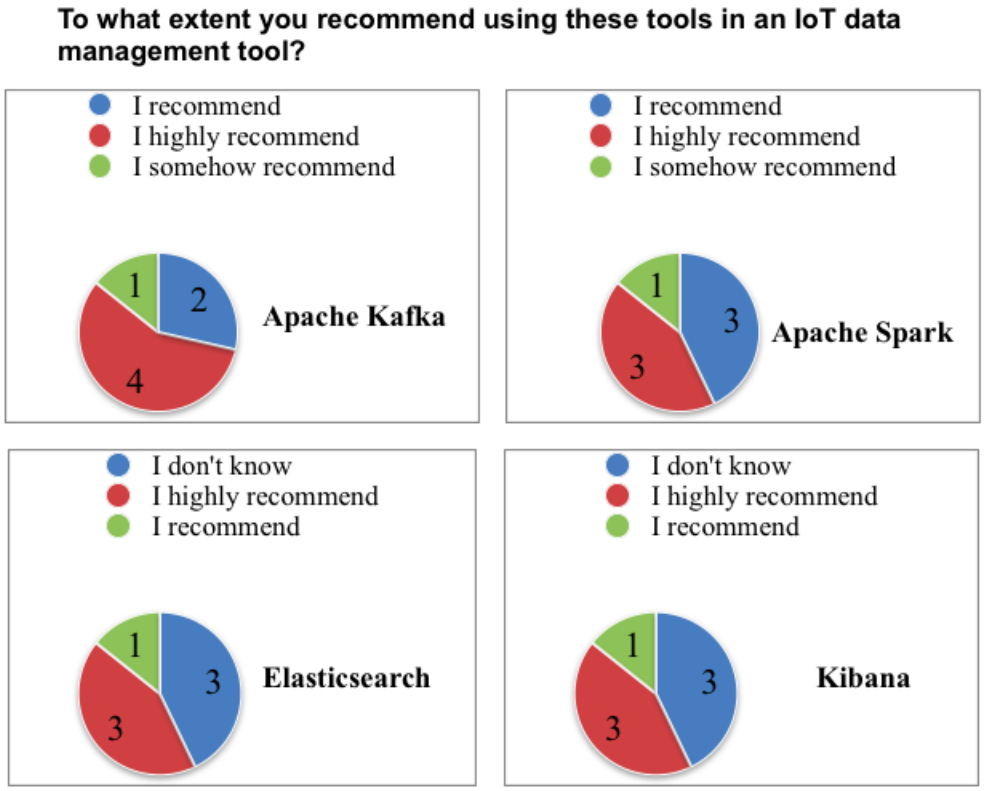


Figure 8.8. The participants' recommendation regarding open-source data analytics tools

In order to quantify how the value could be measured from an IoT framework, different factors were introduced to the participants. As seen from Figure 8.9, "Assuring different modules interoperability" and "Customizability" have received the highest rates in participants' opinion (i.e. 5 out of 7 marked them). In addition, "Solving data heterogeneity problem", "Robustness" and "Reducing deployment, maintenance and support costs" have received acceptable rates of value (4 out of 7 participants selected them). However, "The flexibility in building customer applications on top" as well as "Forecasting" have received a relatively low rate of value in participants' opinion (2 out of 7 selected them). There was one response that suggested other factors, however, did not specify concretely what those factors could be.

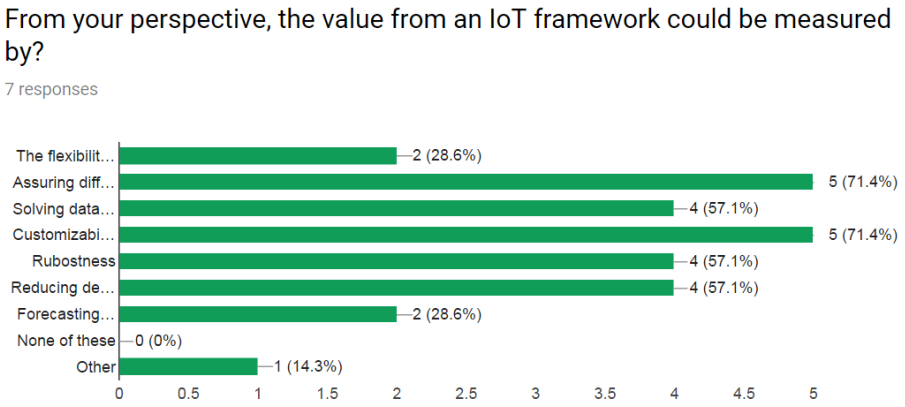


Figure 8.9. The participants' selection of different factor that measure value from an IoT framework

In order to quantify the novel contributions of this research, an open-ended question was introduced to the participants. The answers gave some useful insights about the proposed framework that are worthy to mention. First of all, the participants indicated that one novel contribution is "*providing a solution which integrates the existing tools/libraries and is easy to deploy and maintain*". Next, they reported that "*interoperability between IoT modules as well as integration*" are another two important contributions. In addition, they stated that "*Processing/supporting of different data types, real-time visualization, real-time processing of data*" stands as a contribution. Furthermore, they mentioned that "*Solving data heterogeneity problem*" is on the list. However, they pointed out that "*It is important to compare the development solution against other commercial and academic tools to have a clear understanding of the novelties*".

Additionally, in order to derive some potential additions and improvements to the proposed framework, another open-ended question was introduced to the participants. The participants suggested adding a "*High-quality documentation and installers/scripts*". Another suggestion was to "*Demonstrate more examples of by whom and how the results can be used*". And finally, a viable addition in their opinion might be to "*present more deeply the deployment architecture of the framework and the benefits of it.*" as well emphasizing on "*comparison against commercial and academic tools is necessary*".

With respect to commercializing the proposed framework, the participants reported various attitudes. As seen from Figure 8.10 more than fifty percent reported a positive attitude, namely (28.6%) said they definitely recommend it; while (28.6%) reported that it is very likely for them to recommend. On the other hand, (28.6%) reported uncertainty about

commercializing the framework. Finally, (14.3%) reported that it is not likely at all to recommend it.

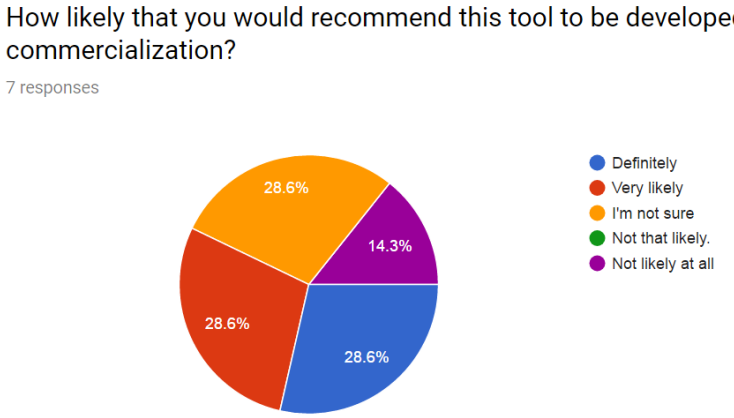


Figure 8.10. The participants' recommendation regarding the framework commercialization

Altogether, the results from the questionnaire formed a supporting opinion for the validity of different aspects of this research. The next chapter will put these results together with the research problem in order to draw a full picture of how these results contributed to answer the research questions and thereby solve the main research problem of this thesis.

9 DISCUSSION

With a moderate level of competence among the participants, and with respect to the novelty of IoT, it is possible that this group can reliably assess an IoT framework since they mostly should have encountered or worked on similar platforms during their career. Therefore, they are probably experienced about known issues and problems, as well as potential solutions. Accordingly, the feedback from the participants' group is most likely to be valuable. Furthermore, the different aspects, concepts and technologies as well as the architecture of this framework were presented through the demonstration video. That is, trying comprehensively to explain this work for them in a way that makes their feedback as reliable as possible.

Generally speaking, the participants were either very convinced with the prototype or fairly convinced (see Figure 8.2). In either case, it is likely a positive indicator that implies the usefulness of this endeavor. Yet the proposed framework needs more exploration and enhancement and maybe another future user study would further enhance the finding of this research. However, based on the collected data and the participants' review the research questions can be answered.

RQ1: *How to normalize heterogeneous data that are being generated by IOT devices from different vendors?*

Through developing the matrix of IoT devices-dimensions that is shown in Table 5.1, the most dominant and recent devices in the IoT market were analyzed and introduced. The specifications of each device were also presented in this matrix. Those devices are produced by different vendors with specific format for each device. The most known formats thus were collected and provided to the data handling tool. Even though this approach doesn't cover all possible IoT data formats in the market, it still considers the most used ones. Through examining public IoT data repositories such as UCI Machine Learning Repository, as well as devices' brochures from different IoT vendors, those formats were found to be the most used when it comes to sending data from devices to the cloud. Those include XML, RDF and CSV to name few. Other formats were also collected and provided to the data handling tool. This tool as mentioned before in Chapter 6 can handle any observation that is formatted with one of this formats and the output will be JSON formatted. Through this approach, almost any platform or application could read and use the outputted data. What is more, if a new IoT format was missed or if an IoT vendor come up with his own format, it could be still added to the data handling tool. Through providing a straightforward documentation and tips, IoT vendors can customize the data handling tool to accept their new formats. While the rest open-source tools will handle the rest of the overhead related to data analytics, storage and visualization.

To summarize, the problem of normalizing heterogeneous IoT devices was tackled by proposing and implementing two steps. First, collecting trending IoT data formats and developing a tool that is capable of receiving IoT observations that are formatted using these trending formats and turning them into standard JSON format. Second, by enabling vendors from customizing the data handling tool, the unsupported, missed or newly presented formats could still be added to the predefined list of formats of the data handling tool. Having that in

mind, it could be concluded therefore that heterogeneous IoT data are normalized by the proposed data handling tool, and that forms the answer for the first research question.

RQ2: *How to use existing open-source real-time analytics tools to develop a generic IOT framework?*

As mention before, the aim of this research is more at a data-centric framework rather than a control-centric where devices are discovered and managed. For this reason, the focus was to integrate efficient data processing and management tools in order to develop a unified solution for IoT data processing after normalization. Through exploring the trending IoT platforms it was possible to shape a comprehensive understanding of how they function and what technologies they utilize, wherein two examples were examined; namely MS Azure and ThinkSpeak. As a result, it was derived that a sequence of activities should be explored and implemented; a data streaming pipeline; an analytics hub and a data presentation layer.

Regarding data streaming and pipelining, a streaming tool that integrates with Kafka was proposed and developed. In that sense, the majority of the participants reported the novelty of this tool (see Figure 8.6). Even though some participants suggested other tools like Apache Apex and Kafka, however, Apache Apex was not explored to check its relevance to this framework. Furthermore, Apache Kafka is more likely to be used as a data pipelining at the first place rather than a streaming tool. Therefore, it was utilized within the developed framework on that basis. On the other hand, most of the participants agreed regarding the novelty as well as the usefulness of streaming archived IoT data (see Figure 8.7). A use case dataset was utilized from the City of Aarhus in Denmark in order to apply the developed framework on a practical example from the real world. Altogether, the participants recommended using Apache Kafka (see Figure 8.8) and the streaming tool in order to stream static data and that is likely to meet the first requirement of the data streaming pipeline.

With respect to the analytics hub, Apache Spark was utilized for applying basic data processing techniques such as MapReduce, transformations and aggregations on streamed IoT data. According to the participants' recommendation (see Figure 8.8), Apache Spark is an efficient data analytics tool that is open source and has a considerable community behind. For those reasons, Spark could be an optimal solution that meets the requirement of an analytics hub for an IoT framework.

Additionally, the requirement of the visualization component was met by the combination of Elasticsearch and Kibana. Both tools were fairly recommended by the participants as well (see Figure 8.8). It might be the participants' unfamiliarity with these tools behind their moderate recommendation though. However, they successfully covered the visualization part of streamed data. Even more, they provide a high-level query language for real-time data which is a quite important feature that is needed for data analytics.

On the basis of this discussion, the integration of the above-mentioned tools results in the generic IoT framework; the data handling tool with Kafka as the data pipeline; Apache Spark as the analytics hub; and Elasticsearch and Kibana dashboard as the presentation layer. Consequently, this integration yields to answering the second research question on building a generic IoT framework based on open-source analytics tools.

This chapter introduced a general discussion on the results obtained from the participants'

review, along with answering the first two research questions. The next chapter summarizes a more detailed overview of the contribution of this research.

10 CONTRIBUTION

During the development of the framework, it was essential to consider creating a value when using a tool or proposing a concept within the context of this research. The reason behind is because, at the end, any novel framework should solve a problem and that in turn practically corresponds to a form of revenue, either being community-driven or economy-oriented. In other words, when deploying an IoT framework it should somehow reduce costs, increase revenues and save resources for stakeholders. Having said that, the development of this framework was primarily committed to this standard and driven by creating value. The list below illustrates the most relevant contributions to the IoT that are believed to add value by the developed framework.

10.1 Solving IoT Data Homogeneity

The presented smart-city scenario showed different IoT data formats. The parking observations are tabular data; while pollution data is comma separated; the traffic data is formatted as JSON, and finally the weather data is formatted in XML. The list of IoT formats is quite long, however, and based on investigating different IoT platform within the scope of this research, no tool is so far capable of handling heterogeneous data from the variety of IoT devices. One value that is brought by this IoT framework is supporting the majority of IoT formats in one tool. The output is always homogeneous data formatted in JSON; that is a standard format which is understandable by almost any platform.

10.2 Customizability for IoT Vendors

From a vendor's point of view, it will be efficient to customize the data handling tool in order to add any new vendor's data format and thus, avoiding extra costs related to licensing or buying access from commercial frameworks. In addition, vendors are likely to find it useful adding a plugin to this tool especially when this will make their IoT modules available to a wide spectrum of customers using the framework. In other words, it could be a mutual benefit between the vendors and the consumers, however, it might not be beneficial to proprietary corporations and alliances though.

10.3 Open-sourced

From a corporation's point of view, a comprehensive IoT deployment will be only dependent on open-source tools that could substitute other commercial frameworks. Having said this, a corporation that adopts such a generic framework will not be obliged to dedicate a budget for maintenance, technical support, and installation; but rather having its own deployment and also making the maximum benefit from community support for an open-source framework. From a customer's point of view, there will not be an obligation to use vendor-driven technologies, and accordingly avoiding 'vendor-lock' that results from proprietary IoT frameworks. In other words, when buying an IoT module a customer should not worry about compatibility issues since heterogeneous modules' integration is already addressed by the proposed data handling tool.

10.4 Wide Range of Supported IoT Applications

When IoT data will be available as a form of processed streams of sensory observations, the

possibilities for developing applications on top are unbounded. Through providing APIs for data, applications can subscribe to consume these data and turn them into services. Taking the smart city scenario within this research, developers can create mobile applications that help users to find nearest available parking. They could as well provide services that suggest healthiest and unhealthiest walking times and routes through the city. Furthermore, sensors monitoring air, soil and water pollution can be efficiently utilized by organizations and governments to take actions for protecting environments and resources. In addition, through measurements of micro-climate in different areas of the city, residents can get up-to-date, high-resolution "block-by-block" weather and climate information [44]. Even more, sensors and actuators inside buildings can create an intelligent sensing environments that help gather real-time data for the patients, monitor their vital signs and identify abnormal situations that need medical attention [43].

10.5 A Backbone for IoT Ecosystem

IoT systems create value by combining sensor capabilities with back-end and front-end systems that turn raw data into information services of value [17]. By providing a data pipeline from sensors in a field until these data are turned into actionable insights, the proposed IoT framework serves as the backbone of an IoT system.

So far, this chapter presented the value brought by the proposed solution to the IoT industrial and academic community. The next chapter will conclude with the challenges, limitations and future directions that derive from this research.

11 CONCLUSION

11.1 Challenges

With respect to developing the data handling tool, it was fundamental to investigate the variety of IoT formats in the market. In order to do that, it was important to look at IoT industry manuals and vendors' brochures. However, it turned to be quite cumbersome to find clear information about specifications of IoT modules; that is, the source where a module's data format could be obtained.

Additionally, in order to validate the technical outcome of this research, it was essential to apply it on a dataset from IoT. On that sense, it was quite challenging to find a relevant source of IoT open data in order to use for this purpose. At the first place, real data were needed for this task in order to make the results as realistic as possible. Having said this, automatic or simulated IoT data might still work, however, it will not give real insights as data from real world might do. Furthermore, it was important to find a dataset that meets two conditions; first it should represent different measured phenomena; and second, those phenomena should have been measured concurrently when they were recorded. The reason behind that is because the intention was to correlate the effects of these phenomena on each other and on their context as well, where they are all demonstrated in one scenario. Accordingly, an open dataset was obtained from the CityPulse project which has observations from traffic and climate that were recorded simultaneously. However, there were some decent IoT repositories that helped a lot in providing IoT data for testing. Kaggle Datasets[18] and UCI Machine Learning Repository[19] to name few good resources.

Finally, another challenge was working with Big Data tools that consume a lot of computing resources. For example, running Kafka brokers besides Spark engine along with and IDE for development is much more than an ordinary local machine can stand. It is even somehow odd to run them locally since they are already designed to run either on the Cloud or on powerful machines. However, it is indispensable to use them locally because of testing and development purposes, nevertheless, they are primarily intended to run on a cluster of computing nodes after deployment.

11.2 Limitations

First and foremost, it is necessary to conduct further in-depth user studies that investigate the usability of the developed framework from the end-users' point of view. That is, a study in which actual users that will benefit from the technical outcome of this research are included. With respect to the smart-city scenario, those users could be staff from municipalities or from observing facilities of climate, traffic and environment, as this tool will be primarily oriented to assist those people in their work. It is also interesting to include data analysts in order to evaluate the possibility for getting insights from analyzed IoT data. However, such a study was not possible because of the limited feasible time within the scope of this thesis, as well as the difficulties related to reaching out to such people as mentioned before.

[18] https://www.kaggle.com/datasets

[19] https://archive.ics.uci.edu/ml/datasets.html

The findings of this study suggest that using trending IoT data formats in the data handling tool, and enabling vendors from customizing this tool to add new formats solves IoT data heterogeneity problem. This approaches could have some limitations since not all possible IoT formats were examined thoroughly. Some of these formats might be straightforwardly processed and might result unexpected outcomes. Though the approach of this research is to enabling customizability through open-sourcing the framework. That could enable vendors from adding their formats and furthermore solving any issues that might rise.

The findings of the study suggest as well that using open-source tools like Apache Spark and Apache Kafka in a unified IoT framework is a viable solution to develop a unified IoT framework. Through the results, it was revealed that such technologies are recommended to be utilized in IoT frameworks. Nevertheless, this suggestion might not be generalizable to other IoT frameworks. This finding is limited until it is compared with the market popularity of these tools in order to examine the generalizability of these results.

Another limitation within the proposed framework is the lack for testing the framework on a distributed cluster of computing nodes as well as the Cloud; that is the structure which ensures scalability when running the framework against a big number of devices. Even though this framework was tested within the scenario of a smart-city, however, the number of devices that were simulated is somehow limited. In real scenarios, the framework is required to operate hundreds of thousands of devices that will be sending large volumes of data at a high speed. As a consequence, integrating the framework on Cloud environments is a necessary requirement.

Even though the orientation of the proposed framework is towards a data-centric approach in tackling IoT data problems, however, it might still be required to provide a function for device management. It will be important for IoT devices to be able to blend in the framework and start sending data seamlessly and using as little as possible of human intervention. Nevertheless, the functionality of device management is not implemented in the proposed framework, the issue that might raise some challenges as additional tools might be needed to identify and control devices.

11.3 Future Work

First of all, the current implementation of the data handling tool is fully dependent on sending the data format from the IoT source (i.e. the device sends the format along with its observations). Practically, this could be improved by automatically detecting the data format from every observation that is being received. By doing that, vendors will not be required to identify their new formats to the tool but rather only sending their data. However, that will be tied to sending the schema along with the data in order for the tool to be able to recognize the structure of the observation. In that sense, some formats like SensorML could be utilized to give a semantic representation for observations.

Since security is a big concern in IoT domain, it would be interesting to further explore how to enhance the security aspect of the proposed framework. For example, it would be possible to secure communication between clients represented by IoT devices and the server represented by the data handling tool. On option would be to use OAuth authentication where the client sends authentication credentials to the server. Next, the server sends a token to the

client. This token is used then by every request in order to send data to the server.

Finally, as the participants pointed out it could be interesting to compare the proposed framework with other commercial and academic tools. That comparison could approach the implementation aspects as well as the architecture in order to address the strengths and weaknesses of this implemented prototype and improve it. Furthermore, more application scenarios could be investigated in order to identify by whom and how the results of this work will be used. In addition, a fully-fledged version of the framework could be supported with a high-quality documentation that thoroughly describes installation tips as well as usage instructions.

REFERENCES

[1] Aberer, K., Hauswirth, M., & Salehi, A. (2007). Infrastructure for Data Processing in Large-Scale Interconnected Sensor Networks. In *2007 International Conference on Mobile Data Management* (pp. 198–205). https://doi.org/10.1109/MDM.2007.36

[2] Aggarwal, C. C., Ashish, N., & Sheth, A. (2013). The Internet of Things: A Survey from the Data-Centric Perspective. In C. C. Aggarwal (Ed.), *Managing and Mining Sensor Data* (pp. 383–428). Springer US. https://doi.org/10.1007/978-1-4614-6309-2_12

[3] Ali, M. I., Gao, F., & Mileo, A. (2015). CityBench: A Configurable Benchmark to Evaluate RSP Engines Using Smart City Datasets. In *Lecture Notes in Computer Science* (pp. 374–389). https://doi.org/10.1007/978-3-319-25010-6_25

[4] Alur, R., Berger, E., Drobnis, A. W., Fix, L., Fu, K., Hager, G. D., … Zorn, B. (2016, April 11). *Systems Computing Challenges in the Internet of Things*. *arXiv [cs.CY]*. Retrieved from http://arxiv.org/abs/1604.02980

[5] Apache Kafka. Retrieved April 27, 2017, from http://kafka.apache.org/documentation

[6] Ashton, K. (2009). That 'internet of things' thing. RFiD Journal, 22(7), 97-114.

[7] Bandyopadhyay, D., & Sen, J. (2011). Internet of Things: Applications and Challenges in Technology and Standardization. *Wireless Personal Communications*, *58*(1), 49–69. https://doi.org/10.1007/s11277-011-0288-5

[8] Bassi, A., & Horn, G. (2008). Internet of Things in 2020: A Roadmap for the Future. *European Commission: Information Society and Media*.

[9] Bischof, S., Karapantelakis, A., Nechifor, C.-S., Amit P. Sheth, W. S. U.-M. C., Mileo, A., Barnaghi, P., & Authors. (2014). Semantic Modelling of Smart City Data. Retrieved from http://corescholar.libraries.wright.edu/knoesis/572/

[10] Chourey, S., & Profile, V. Getting Started with Spark on Windows 7 (64 bit). Retrieved April 27, 2017, from http://letstalkspark.blogspot.com/2016/02/getting-started-with-spark-on-window-64.html

[11] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper. Retrieved May 1, 2017, from http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html

[12] CityPulse Smart City Datasets - Datasets. Retrieved April 22, 2017, from http://iot.ee.surrey.ac.uk:8080/datasets.html#pollution

[13] Compton, M., Henson, C., Lefort, L., Neuhaus, H., & Sheth, A. (2009). A Survey of the Semantic Specification of Sensors. In *Proceedings of the 2Nd International Conference on Semantic Sensor Networks - Volume 522* (pp. 17–32). Aachen, Germany, Germany: CEUR-WS.org. Retrieved from http://dl.acm.org/citation.cfm?id=2889933.2889935

[14] Derhamy, H., Eliasson, J., Delsing, J., & Priller, P. (2015). A survey of commercial frameworks for the Internet of Things. In *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)* (pp. 1–8). https://doi.org/10.1109/ETFA.2015.7301661

[15] Díaz, M., Martín, C., & Rubio, B. (2016/5). State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing. *Journal of Network and Computer Applications*, *67*, 99–117. Retrieved from http://www.sciencedirect.com/science/article/pii/S108480451600028X

[16] Dorsemaine, B., Gaulier, J.-P., Wary, J.-P., Kheir, N., & Urien, P. (2015). Internet of Things: a definition & taxonomy. In *Next Generation Mobile Applications, Services and Technologies, 2015 9th International Conference on* (pp. 72–77). IEEE. Retrieved from http://ieeexplore.ieee.org/abstract/document/7373221/

[17] Explaining the Internet of Things Ecosystem and Taxonomy. (2017). Retrieved 16 March 2017, from https://451research.com/images/Marketing/IoT/IoT_Taxonomy_12.1.15.pdf

[18] Gartner Symposium/ITxpo IoT Forecast. (2017). Gartner.com. Retrieved 14 February 2017, from https://www.gartner.com/newsroom/id/3165317

[19] Getting Started | Elasticsearch Reference [5.3] | Elastic. Retrieved April 28, 2017, from https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started.html

[20] Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013/9). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generations Computer Systems: FGCS*, *29*(7), 1645–1660. Retrieved from http://www.sciencedirect.com/science/article/pii/S0167739X13000241

[21] Hong, Y. (2012). A Resource-Oriented Middleware Framework for Heterogeneous Internet of Things. In *2012 International Conference on Cloud and Service Computing* (pp. 12–16). https://doi.org/10.1109/CSC.2012.10

[22] Intelligence, S. C. B. (2010). Disruptive Technologies Global Trends 2025. *Energy Storage Materials*.

[23] Jiang, L., Xu, L. D., Cai, H., Jiang, Z., Bu, F., & Xu, B. (2014). An IoT-Oriented Data Storage Framework in Cloud Computing Platform. *IEEE Transactions on Industrial Informatics*, *10*(2), 1443–1451. https://doi.org/10.1109/TII.2014.2306384

[24] Kamburugamuve, S., Christiansen, L., & Fox, G. (2015). A Framework for Real Time Processing of Sensor Data in the Cloud. *Journal of Sensors*, *2015*. https://doi.org/10.1155/2015/468047

[25] Kaplinsky, R., & Morris, M. (2001). *A handbook for value chain research* (Vol. 113). IDRC Ottawa. Retrieved from http://www.prism.uct.ac.za/Papers/VchNov01.pdf

[26] Kolozali, S., Bermudez-Edo, M., Puschmann, D., Ganz, F., & Barnaghi, P. (2014). A Knowledge-Based Approach for Real-Time IoT Data Stream Annotation and Processing. In

2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) (pp. 215–222). https://doi.org/10.1109/iThings.2014.39

[27] Methods for Software Prototyping. (2017). Retrieved 2 May 2017, from http://sce.uhcl.edu/helm/REQ_ENG_WEB/My-Files/mod4/Software_Prototyping.pdf

[28] Miorandi, D., Sicari, S., De Pellegrini, F., & Chlamtac, I. (2012/9). Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, *10*(7), 1497–1516. Retrieved from http://www.sciencedirect.com/science/article/pii/S1570870512000674

[29] Observations and Measurements | OGC. Retrieved April 25, 2017, from http://www.opengeospatial.org/standards/om

[30] Page, K. R., De Roure, D. C., Martinez, K., Sadler, J. D., & Kit, O. Y. (2009). Linked Sensor Data: RESTfully Serving RDF and GML. In *Proceedings of the 2Nd International Conference on Semantic Sensor Networks - Volume 522* (pp. 49–63). Aachen, Germany, Germany: CEUR-WS.org. Retrieved from http://dl.acm.org/citation.cfm?id=2889933.2889937

[31] Paridel, K., Bainomugisha, E., Vanrompay, Y., Berbers, Y., & De Meuter, W. (2010). Middleware for the internet of things, design goals and challenges. *Electronic Communications of the EASST*, *28*. Retrieved from https://www.researchgate.net/profile/Koosha_Paridel/publication/220054384_Middleware_for_the_Internet_of_Things_Design_Goals_and_Challenges/links/00b4951a5efb7ddc35000000.pdf

[32] Postscapes.com. Internet of Things Infographic | What Is The “Internet of Things”? . Retrieved 3 May 2017, from https://www.postscapes.com/what-exactly-is-the-internet-of-things-infographic/

[33] Postscapes.com. IoT Cloud Platform Landscape | 2017 Vendor List. Retrieved 30 April 2017, from https://www.postscapes.com/internet-of-things-platforms/

[34] Postscapes.com. IoT Market | 2017 Forecast Size and Growth Projections by Country, Year, Industry Market Verticals, and Analysts. (2017). Retrieved 14 February 2017, from http://www.postscapes.com/internet-of-things-market-size/

[35] Ralhan, P. (2000). Web.njit.edu. Retrieved 2 May 2017, from https://web.njit.edu/~turoff/coursenotes/CIS732/samplepro/prototyping.doc

[36] Rebooting the IT Revolution - National Science Foundation. (n.d.). Retrieved from https://www.nsf.gov/crssprgm/nano/reports/2015-0901_RITR%20WEB%20version%20FINAL_39p.pdf

[37] Research Nester (2017). Internet of Things (IoT) Market: Global Demand, Growth Analysis & Opportunity Outlook 2023. Retrieved from http://www.researchnester.com/reports/internet-of-things-iot-market-global-demand-growth-analysis-opportunity-outlook-2023/216

[38] Rodríguez, A., McGrath, R., Liu, Y., & Myers, J. (2009). Semantic Management of Streaming Data. In *Proceedings of the 2Nd International Conference on Semantic Sensor Networks - Volume 522* (pp. 80–95). Aachen, Germany, Germany: CEUR-WS.org. Retrieved from http://dl.acm.org/citation.cfm?id=2889933.2889939

[39] Rose, K., Eldridge, S., & Chapin, L. (2015). The internet of things: An overview. *The Internet Society (ISOC)*, 1–50. Retrieved from https://pdfs.semanticscholar.org/6d12/bda69e8fcbbf1e9a10471b54e57b15cb07f6.pdf

[40] Shahrivari, S. (2014). Beyond Batch Processing: Towards Real-Time and Streaming Big Data. *Computers*, *3*(4), 117–129. https://doi.org/10.3390/computers3040117

[41] Sundmaeker, H., Guillemin, P., Friess, P., & Woelfflé, S. (2010). Vision and challenges for realising the Internet of Things. *Cluster of European Research Projects on the Internet of Things, European Commision*. Retrieved from http://www.robvankranenburg.com/sites/default/files/Rob%20van%20Kranenburg/Clusterbook%202009_0.pdf

[42] Tönjes, R., Barnaghi, P., Ali, M., Mileo, A., Hauswirth, M., Ganz, F., ... & Puiu, D. (2014). Real time iot stream processing and large-scale data analytics for smart city applications. *In poster session, European Conference on Networks and Communications*.

[43] Tragos, E. Z., Foti, M., Surligas, M., Lambropoulos, G., Pournaras, S., Papadakis, S., & Angelakis, V. (2015). An IoT based intelligent building management system for ambient assisted living. In *2015 IEEE International Conference on Communication Workshop (ICCW)* (pp. 246–252). https://doi.org/10.1109/ICCW.2015.7247186

[44] Urban Center for Computation and Data. Array Of Things. Retrieved 13 April 2017, from https://arrayofthings.github.io

[45] Vlacheas, P., Giaffreda, R., Stavroulaki, V., Kelaidonis, D., Foteinos, V., Poulios, G., … Moessner, K. (2013). Enabling smart cities through a cognitive management framework for the internet of things. *IEEE Communications Magazine*, *51*(6), 102–111. https://doi.org/10.1109/MCOM.2013.6525602

[46] Zhang, Y., Raychadhuri, D., Grieco, L. A., Baccelli, E., Burke, J., Ravindran, R., … Schelén, O. (2015). Requirements and Challenges for IoT over ICN. Retrieved from http://www.diva-portal.org/smash/record.jsf?pid=diva2:1043578

[47] Zorzi, M., Gluhak, A., Lange, S., & Bassi, A. (2010). From today's INTRAnet of things to a future INTERnet of things: a wireless- and mobility-related view. *IEEE Wireless Communications*, *17*(6), 44–51. https://doi.org/10.1109/MWC.2010.5675777

Appendix A
The participants' questionnaire

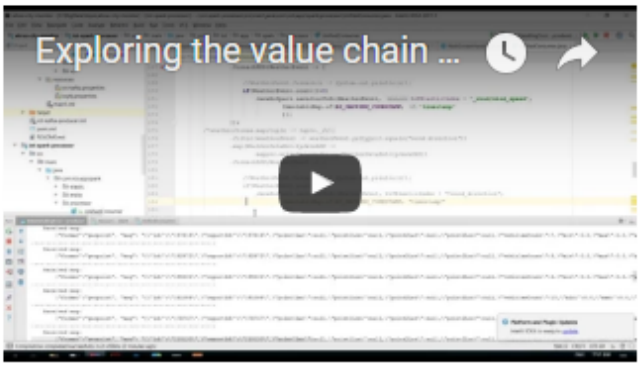
Exploring the value chain of IoT in real time analytics paradigm

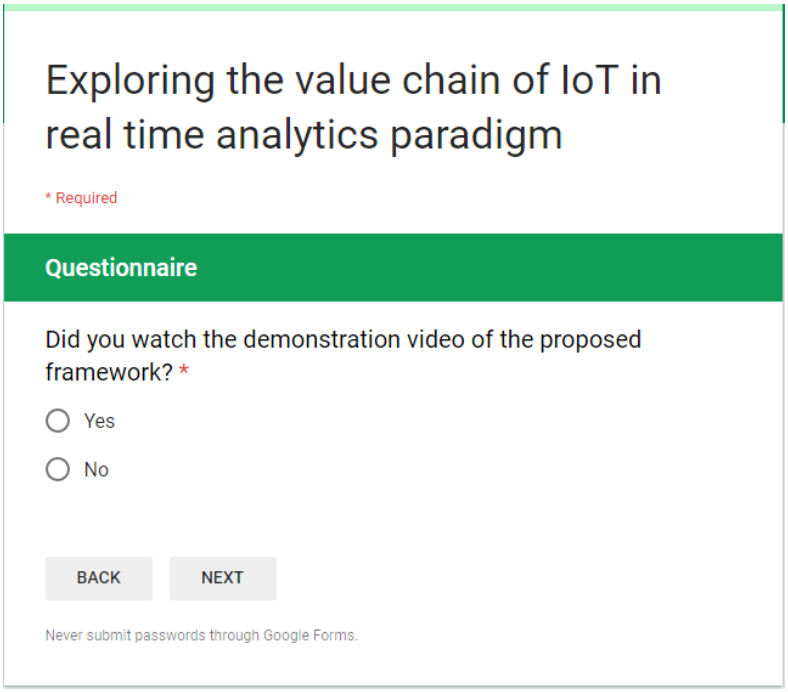
This questionnaire is conducted as part of my master thesis work. My master program is Social Media and Web Technologies, Department of Media Technology at Linnaeus University. Please take your time to answer the following group of questions as careful and as honest as possible. Your answers and feedback is of a high value to this effort and will greatly contribute to improve the results as well as the outcome of this research. Please peer in mind that your responses will be kept and processed anonymously.

Student: Mohamad Zamam
Supervisor: Dr. Juwel Rana

Demonstration Video

Please check the following video that demonstrates key aspects of my research outputted as a from of a prototype. The value proposition here focuses on the IoT data processing and management in order to maximize the value brought to industry. A data handling and streaming tool is developed along with using other open-source platforms in order to create a generic IoT platform. The different tools and technologies that were utilized to develop the prototype are presented in the video as well. Please watch the video first and move to answer the questionnaire next.





Exploring the value chain of IoT in real time analytics paradigm

* Required

Questionnaire

Did you watch the demonstration video of the proposed framework? *

- Yes
- No

BACK NEXT

Never submit passwords through Google Forms.

Exploring the value chain of IoT in real time analytics paradigm

* Required

How do you rate your level of competence in the Internet of Things (IoT) domain ? *

- Outstanding
- Competent
- Moderate
- Low competence
- No competence

While watching the demonstration, how convinced or unconvinced were you about the usefulness of the proposed IoT framework ? *

- Very convinced
- Somewhat convinced
- Neither convinced nor unconvinced
- Somewhat unconvinced
- Very unconvinced

This IoT framework, if adopted by IoT practitioners, can bring value that commercial frameworks can't by potentially reducing licensing, maintenance and support costs since it is fully open-sourced. To what level you agree or disagree with this claim? *

- ○ Strongly agree
- ○ Somewhat agree
- ○ Neither agree nor disagree
- ○ Somewhat disagree
- ○ Strongly disagree

The proposed IoT framework potentially solves the heterogeneity problem of IoT devices though processing wide spectrum of IoT data formats. To what level you agree or disagree with this claim? *

- ○ Strongly agree
- ○ Somewhat agree
- ○ Neither agree nor disagree
- ○ Somewhat disagree
- ○ Strongly disagree

Which of the following factors you believe can potentially promote the proposed IoT framework? *

- ☐ Open-sourced
- ☐ Community support
- ☐ GNU License
- ☐ Free updates
- ☐ Customizability
- ☐ None
- ☐ Other: _____

* Required

Do you believe that it is useful to simulate the streaming of historical IoT data in order to get insights from that data? *

- ◯ Yes
- ◯ No
- ◯ Maybe

To what extent you recommend using these tools in an IoT data management tool? *

	I don't recommend	I somehow recommend	I don't know	I recommend	I highly recommend
Apache Kafka	◯	◯	◯	◯	◯
Apache Spark	◯	◯	◯	◯	◯
Elasticsearch	◯	◯	◯	◯	◯
Kibana	◯	◯	◯	◯	◯

The proposed tool simulates the streaming of IoT data as if they are happening in real time. Do you know about any other tool that provides the same feature? *

- ◯ Yes
- ◯ No

BACK NEXT

* Required

What streaming simulation tool you are familiar with? *

Your answer

BACK NEXT

How likely that you would recommend this tool to be developed for commercialization?

- ◯ Definitely
- ◯ Very likely
- ◯ I'm not sure
- ◯ Not that likely.
- ◯ Not likely at all

BACK NEXT

From your perspective, the value from an IoT framework could be measured by?

- [ ] The flexibility in building customer applications on top
- [ ] Assuring different modules interoperability
- [ ] Solving data heterogeneity problem
- [ ] Customizability
- [ ] Rubostness
- [ ] Reducing deployment, maintenance and support costs
- [ ] Forecasting and prediction
- [ ] None of these
- [ ] Other: _____

In your opinion, what is the most useful contribution (if any) that could be obtained from this proposed tool?

Your answer

What would you recommend as an addition or a change to this tool?

Your answer

APPENDIX B

Observation Schema for the Weather Element of the City Of Aarhus Dataset

```
<?xml version="1.0" encoding="UTF-8"?>
<om:OM_Observation xmlns:om="http://www.opengis.net/om/2.0"
                                    xmlns:gml="http://www.opengis.net/gml/3.2"
                                    xmlns:swe="http://www.opengis.net/swe/2.0"
                                    xmlns:xlink="http://www.w3.org/1999/xlink"

xmlns:xsi="http://www.w3.org/2001/XMLSchema:instance"
                                    gml:id="WeatherObservation"
                                    xsi:schemaLocation="http://www.opengis.net/om/2.0
http://schemas.opengis.net/om/2.0/observation.xsd http://www.opengis.net/swe/2.0
http://schemas.opengis.net/sweCommon/2.0/swe.xsd">
  <gml:description>Complex Weather Observation</gml:description>
  <om:type
xlink:href="http://www.opengis.net/def/observationType/OGC:OM/2.0/OM_ComplexObse
rvation" />
  <om:phenomenonTime>
    <gml:TimeInstant gml:id="ot1t">
      <gml:timePosition>2014:08:01T00:00:00</gml:timePosition>
    </gml:TimeInstant>
  </om:phenomenonTime>
  <om:resultTime xlink:href="#ot1t" />
  <om:procedure xlink:href="http://www.example.org/register/process/weatherStation3" />
  <om:observedProperty xlink:href="http://sweet.jpl.nasa.gov/2.0/atmo.owl#Weather" />
  <om:featureOfInterest xlink:href="http://www.ga.gov.au/bin/gazd01?rec=293604"
xlink:role="urn:cgi:featureType:SEEGRID:framework:locality" />
  <om:result xsi:type="swe:DataRecordPropertyType">
    <swe:DataRecord definition="record_weather.xml">
      <swe:field name="Temperature">
        <swe:Quantity>
          <swe:uom xlink:href="Cel" />
          <swe:value>35.1</swe:value>
        </swe:Quantity>
      </swe:field>
      <swe:field name="WindSpeed">
        <swe:Quantity>
          <swe:uom xlink:href="km.h:1" />
          <swe:value>6.5</swe:value>
        </swe:Quantity>
      </swe:field>
      <swe:field name="WindDirection">
        <swe:Quantity>
          <swe:uom xlink:href="deg" />
          <swe:value>085.0</swe:value>
        </swe:Quantity>
      </swe:field>
      <swe:field name="Humidity">
```

```
        <swe:Quantity>
          <swe:uom xlink:href="percent" />
          <swe:value>32.</swe:value>
        </swe:Quantity>
      </swe:field>
    </swe:DataRecord>
  </om:result>
</om:OM_Observation>
```

APPENDIX C

Installation Tips for Used Open-source Tools and Frameworks

C.1. Spark Installation

The used version is 2.1.0. It can be downloaded from Apache Spark download page:

http://spark.apache.org/downloads.html

Note: When using the Scala API for apache, these following steps become necessary. Otherwise when using Java API, as in the proposed framework, the Spark artifact can be downloaded as .jar file and included in Java classpath, or included in the dependency file of the project. For an instance, the following dependency can be added to the pom.xml when using Intellij IDE:

```
groupId:                    org.apache.spark
artifactId:                 spark-core_2.11
version: 2.1.0
```

All Spark artifacts are hosted in Maven Central and can be downloaded from there. Note, as well, that it is important to be careful about the version of the included dependency, since it is essential for the integration with other components (such as Spark Streaming component).

Follow these steps when using Spark with the Scala API:

- Download and install Java (downloaded tool: Java SE Development Kit 8u131)
- Make sure `JAVA_HOME` is defined in Environment variables and pointing to Java home directory.
- Download & Install Anaconda Python 3.5+. (Extract to `Path\to\Anaconda3` or any folder)
- Download winutils.exe (Put in `Path\to\Hadoop\bin`). This is for Win 64-bit. The downloaded version could be obtained from this link:

  https://github.com/steveloughran/winutils/tree/master/hadoop-2.7.1/bin
- Create Environment Variables :

  SPARK_HOME : Path\to\Spark
  HADOOP_HOME : Path\to\Hadoop
- Add to Environment variable PATH at the end:

  `%SPARK_HOME%\bin;%HADOOP_HOME%\bin;%JAVA_HOME%\bin;`
- Create folder Path\to\tmp\hive
- On Command prompt in Admin Mode : (One time only) navigate to `Path\to\Hadoop\winutils.exe chmod -R 777 Path\to\tmp\hive`
- On command prompt in Administrator mode start Spark using :

  `pyspark --packages com.databricks:spark-csv_2.11:1.3.0`
- You should see Welcome to Spark version [Spark version] using Python 3.5.1 message.

C.2. Kafka Installation

C.2.1 Zookeeper

Instructions from https://dzone.com/articles/running-apache-kafka-on-windows-os

Download from http://apache.mirrors.spacedump.net/zookeeper/ version: zookeeper-3.4.9

Goto Zookeeper config directory: `Path\to\Zookeeper\conf`

- Rename file "zoo_sample.cfg" to "zoo.cfg"
- Open zoo.cfg in any text editor
- Find & edit `dataDir=/tmp/zookeeper` to `:\Zookeeper\data`
- Create Environment Variable:

```
ZOOKEEPER_HOME = Path\to\Zookeeper
```

- Edit System Variable named "Path" add ;%ZOOKEEPER_HOME%\bin;
- Run Zookeeper by opening a new cmd and type `zkserver`.

C.2.1 Kafka

- Go to Kafka config directory. Path\to\Kafka\config
- Edit file "server.properties"
- Find & edit line "`log.dirs=/tmp/kafka-logs`" to "`log.dir= Path\to\tmp\kafka-logs`".
- If Zookeeper is running on some other machine or cluster we can edit "zookeeper.connect:2181" to our custom IP and port. For this work we are using same machine so no need to change. Also Kafka port & broker.id are configurable in this file. Leave other settings as it is.
- Kafka will run on default port 9092 & connect to zookeeper's default port which is 2181.

C.2.3 Running Kafka Server

Important: Ensure that Zookeeper instance is up and running before starting a Kafka server.

- Go to Kafka installation directory Path\to\Kafka\
- Open a command prompt here by pressing Shift + right click and choose"Open command window here" option)
- Type `> .\bin\windows\kafka-server-start.bat .\config\server.properties` and press Enter.
- If everything went fine, now Kafka should be up and running.

C.2.4 Kafka Topics:

Create and consuming a topic

- In the command prompt, navigate to `Kafka` home directory. `Path\to\Kafka`
- Create a topic through running the command `> bin\windows\kafka-topics.bat --create --zookeeper localhost:2181 --replication-`

```
factor 1 --partitions 1 --topic test
```

- Start messaging through a producer:
  Run the command `>bin\windows\kafka-console-producer.bat --broker-list localhost:9092 --topic test`
- Start consuming messages through creating a consumer > `bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic test --from-beginning`