

**Submission of Thesis to Norma Smurfit Library, National College of Ireland**

Student name: James Doherty Student number: x13103776

School: School of Computing Course: Cloud Computing

Degree to be awarded: MSc.

Title of Thesis: \_\_\_\_\_

Optimisation of Internet Bandwidth and Application Latency using the

Edge Cloud and Machine Learning of IOT data

One hard bound copy of your thesis will be lodged in the Norma Smurfit Library and will be available for consultation. The electronic copy will be accessible in TRAP (<http://trap.ncirl.ie/>), the National College of Ireland's Institutional Repository. In accordance with normal academic library practice all theses lodged in the National College of Ireland Institutional Repository (TRAP) are made available on open access.

I agree to a hard bound copy of my thesis being available for consultation in the library. I also agree to an electronic copy of my thesis being made publicly available on the National College of Ireland's Institutional Repository TRAP.

Signature of Candidate: *James Doherty*

For completion by the School:

The aforementioned thesis was received by \_\_\_\_\_ Date: \_\_\_\_\_

This signed form must be appended to all hard bound and electronic copies of your thesis submitted to your school

# Submission of Thesis and Dissertation

National College of Ireland  
Research Students Declaration Form  
(Thesis/Author Declaration Form)

Name: James Doherty

Student Number: x13103776


Degree for which thesis is submitted: MSc. in Cloud Computing

## Material submitted for award

- (a) I declare that the work has been composed by myself.
- (b) I declare that all verbatim extracts contained in the thesis have been distinguished by quotation marks and the sources of information specifically acknowledged.
- (c) My thesis will be included in electronic format in the College Institutional Repository TRAP (thesis reports and projects)
- (d) **Either** \*I declare that no material contained in the thesis has been used in any other submission for an academic award.  
**Or** \*I declare that the following material contained in the thesis formed part of a submission for the award of

---

*(State the award and the awarding body and list the material below)*

Signature of research student: 

Date: 20/09/2015

OPTIMISATION OF INTERNET BANDWIDTH  
AND APPLICATION LATENCY USING THE  
EDGE CLOUD AND MACHINE LEARNING OF  
IOT DATA

AUTHOR JAMES DOHERTY



SUBMITTED AS PART OF THE REQUIREMENTS FOR THE DEGREE  
OF MSc IN CLOUD COMPUTING  
AT THE SCHOOL OF COMPUTING,  
NATIONAL COLLEGE OF IRELAND  
DUBLIN, IRELAND.

September 2015

Supervisor Adriana Chis

# Abstract

After Cloud computing the Internet of Things (IOT) has been heralded as the next step in the evolution of the Internet. A central concern is the integration of the IOT and the Cloud, which faces significant challenges. Gartner, the worlds leading information technology research and advisory company, predicts that by 2020 there will be over 26 billion connected devices exchanging data and information over the Internet. With this significant growth in data will come challenges such as increased network load and application latency.

Nielsen's Law of Internet Bandwidth indicates that connection speeds grow by only 50% per year. Therefore it becomes apparent that an approach is required to address future limitations. To overcome some of these issues recent developments in this field have seen applications moving to the data, to create the Edge Cloud. In effect it switches the current method around, which, in the centralised data centre model of Cloud Computing, brings the data to the application.

This paper proposes the use of an Intelligent Agent on the Edge Cloud to deliver workload optimisation between the Cloud and the Edge Cloud through Machine Learning. These Agents utilise Artificial Intelligence to map future events based on past analysis of IOT data. While recent research has illustrated the need for Smart Gateways to pre-process and trim data, this paper proposes to implement a means of learning from past actions taken which will be hosted on the Edge Cloud. It is proposed that, over time, improvements will be seen in the usage patterns of scalable Cloud resources, as well as a reduction in the network load and improvements in application latency.

# Acknowledgement

This dissertation work Optimisation of Internet Bandwidth and Application Latency using the Edge Cloud and Machine Learning of IOT data for Masters in Cloud computing is accomplished at National College of Ireland, Cloud competency Centre. I am grateful to my supervisor Dr. Adriana Chis. I would like to thank her for her guidance, efforts and discussions during the course work. Every meeting helped me to achieve the desired results in the research work.

# Declaration

All the work submitted as a part of this research work is solely implemented and edited by me. All the sources of information and knowledge are referenced properly inside the document.

Signed ..... Date .....

Mr. James Doherty

# Contents

<b>Abstract</b>	<b>ii</b>
<b>Acknowledgement</b>	<b>iii</b>
<b>Declaration</b>	<b>iv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Document Outline . . . . .	2
<b>2 Background</b>	<b>4</b>
2.1 Internet Of Things . . . . .	4
2.2 Data Transmission to the Internet of Things . . . . .	5
2.3 Cloud Computing . . . . .	7
2.4 Edge Cloud . . . . .	8
2.5 Machine Learning . . . . .	9
<b>3 Design</b>	<b>10</b>
3.1 Design Overview . . . . .	10
3.2 Motivation . . . . .	10
3.3 Current Solution . . . . .	11
3.4 Proposed Solution . . . . .	12
3.4.1 Data Source - Internet Of Things . . . . .	13
3.4.2 Predictive Analytics . . . . .	15
3.4.3 Computation and Storage . . . . .	15
3.5 Use Case . . . . .	15
<b>4 Implementation</b>	<b>18</b>
4.1 Data Source - Internet Of Things . . . . .	18
4.1.1 Contiki-os . . . . .	18
4.1.2 Cooja - The Contiki Network Simulator . . . . .	19
4.1.3 Reverse Proxy . . . . .	21

4.2	Predictive Analytics . . . . .	22
4.2.1	Azure ML Web Service . . . . .	22
4.3	Cloud Resources . . . . .	24
4.3.1	Data Retrieval . . . . .	24
4.3.2	Data Processing . . . . .	25
<b>5</b>	<b>Evaluation</b>	<b>26</b>
5.1	Test Case Outline . . . . .	26
5.2	Test Environments Setup . . . . .	27
5.2.1	Edge Cloud setup . . . . .	27
5.2.2	Conventional Cloud setup . . . . .	28
5.3	Results Analysis . . . . .	28
5.3.1	Analysis of Bandwidth Usage . . . . .	28
5.3.2	Analysis of Request/Response Times . . . . .	30
<b>6</b>	<b>Conclusion</b>	<b>33</b>
6.1	Limitations of Proposed Approach . . . . .	33
6.1.1	Data Retrieval Technique . . . . .	33
6.2	Future work . . . . .	34
6.2.1	Implement a local ML resource . . . . .	34
6.2.2	Use of Hardware . . . . .	34
6.2.3	Cost Analysis . . . . .	34
<b>A</b>	<b>Code Listing</b>	<b>37</b>



# List of Figures

2.1	Internet of Things layers Aazam and Huh (2014)	4
3.1	Current Centralised Cloud using Predictive Analytics	11
3.2	Centralised Cloud and ML resources	12
3.3	Proposed Edge Cloud using Predictive Analytics	12
3.4	Edge Cloud - resources co-located near the Data Source	13
3.5	Reverse Proxy and Boarder Router in IOT Network	14
3.6	Wind Farm use case for Edge Cloud	17
4.1	Cooja Simulator running 20 sensor nodes and 1 boarder router node	20
4.2	Boarder Router web page showing Nodes and Routes	21
4.3	Sensor Node showing Json String	21
4.4	ML Training Experiment	23
4.5	ML Web Service	24
5.1	Results of ML for each type of Node	29
5.2	Normal Distribution for ML request/response times on Edge Cloud	31
5.3	Normal Distribution for ML request/response times on Conventional Cloud	32
5.4	Normal Distribution for IOT request/response times on Edge Cloud	32
5.5	Normal Distribution for IOT request/response times on Conventional Cloud	32

# Listings

A.1 Python code to utilise ML Web Service . . . . .	37
A.2 Javascript NodeJs Reverse Proxy Server . . . . .	38

# Chapter 1

## Introduction

The Internet of Things (IOT), a phrase introduced by [Ashton \(2009\)](#), is a scenario where uniquely identified everyday objects are connected together via the Internet. However, according to [Fantacci, Pecorella, Viti, and Carlini \(2014\)](#) the IOT faces a major problem from fragmentation and interoperability. A common set of standards as well as interoperability among systems will unlock the true potential of the IOT. There are a number of large companies, such as Google, Intel and Microsoft, attempting to provide a set of such standards. Developed by Google, the "Thread"<sup>1</sup> protocol is one example of such a standard, however, Microsoft have formed another interest group called the "AllSeen Alliance"<sup>2</sup> which have another vision. While these challenges exist throughout the layers and technologies there is no indication of the IOT growth slowing. With 8 billion devices reported in 2012, and predictions of 26 billions connected devices by 2020, there will be a dramatic increase in data traversing the Internet, in the multitude of different formats and standards. While different bodies throughout the globe attempt to draw up standards that will help with many of the issues, the existing technology stack needs to adapt to allow, as much as possible, unhindered growth in the area. Gartner predicts that the volume of data produced by the IOT will pose significant challenges for the existing data centres. The increased inbound data from autonomous globally distributed IOT sensor networks to a single location for processing will cause issues. Future predictions are that current data centre bandwidth capacities, designed to handle human generated volumes of data, will not be able to accommodate the load, neither technologically nor economically. While bandwidth capacities of centralised data centres struggle to keep up with demand alternative methods of managing and processing the IOT data must be found. One suggestion is changing the manor

---

<sup>1</sup><http://www.threadgroup.org>

<sup>2</sup><https://allseenalliance.org/>

in which we handle the vast amounts of data generated. [Chang, Hari, Mukherjee, and Lakshman \(2014\)](#) introduce the Edge Cloud, a new model for cloud computing. This model supplements the traditional data centre with nodes placed at the edge of the network to service the IOT data locally, reducing the need for presenting raw data to the data centre.

When combined Moores Law, where computer-processing power is said to double every two years, and Kryder's Law, where storage continues to become cheaper, the Edge Cloud becomes a practical solution. These observed laws allow for greater computation and storage on Edge devices, reducing centralised processing requests towards the Cloud. This also introduces opportunities to ensure additional resources are available only when required by optimising the integration between the Cloud and the Edge Cloud. Recent research has highlighted the possible use of Agents to delegate the workload between the Edge Cloud and the centralised Cloud. These Agents call on resources in the Cloud when certain triggers or messages on the local Edge Cloud are received. While improvement can be seen with this delegation there is scope for further improvement by using none static triggering methods that evolve over time. Techniques like Microsoft predictive analytics can be adapted to introduce Machine Learning at the Edge to provide intelligent integration between the Edge Cloud and the Centralised Cloud.

In this dissertation I propose that if IOT applications are deployed on an Edge Cloud and used in conjunction with centralised Cloud resources, improvements in the application latency and utilisation of Internet bandwidth will be seen when measured against the solitary use of centralised Cloud resources. The main contributions of this thesis are the following:

- A prototype Cloud application that can be deployed on resource-limited infrastructure situated near an IOT sensor network or Centralised Cloud infrastructure. This application will gather and processes sensor data in real time and also will use Machine Learning resources as an efficient approach to identifying when to send data for further analysis.
- An empirical evaluation which shows that our solution improves application latency and efficiently utilises Internet bandwidth.

## 1.1 Document Outline

In order to understand the overall topic at hand this paper will introduce the reader to a number of relevant subjects, giving some background.

In [chapter 2](#) we discuss the existing state of the art in the field. The areas include an introduction and review of the IOT concept in [section 2.1](#). In order to transmit the large amounts of data from the IOT across the Internet it is important to understand the current technologies that are used. In [section 2.2](#) we discuss the different options that are currently in use and also proposed methods of data transmission between machines over the Internet. In [section 2.3](#) we introduce the Cloud to the user and outline the limitation for use of the current Cloud infrastructure with the IOT. In [section 2.4](#) we discuss a new paradigm called the Edge Cloud where we move the processing and storage near the data. We identify the benefits to the user with this method. In [section 2.5](#) we discuss the current use of Machine Learning and Artificial Intelligence in the Cloud and identify the best methods of bringing this technology to the Edge Cloud. In [chapter 3](#) we discuss the current design and outline the design decisions for the prototype application. Also in this chapter we outline a use case for the prototype application. In [chapter 4](#) we outline the technologies used in the implementing the design for the use case. Finally, [chapter 5](#) identifies the test case and the setup for the prototype. The results obtained are reviewed and evaluated against the hypothesis.

## Chapter 2

# Background

### 2.1 Internet Of Things

[Ashton \(2009\)](#) predicted that future computing would depend more on machine-generated data, captured by computer-enabled objects or things. As a result the term Internet of Things (IOT) was introduced. Remote objects are identified or discovered. After perceiving surrounding data they interact with servers over the Internet. The amount of this machine-generated data has already surpassed that generated by humans [Walker, Blodgett, Suftin, and Kunicki \(2013\)](#). Machine-generated is data that was produced by machines only, without human intervention [Slezak and Kowalski \(2013\)](#). [Aazam and Huh \(2014\)](#) describe the Internet of Things by breaking it down into five layers. These layers include the Perception layer, the Network layer, the Middleware layer, the Application layer and Business layer as illustrated below in [Figure 2.1](#).

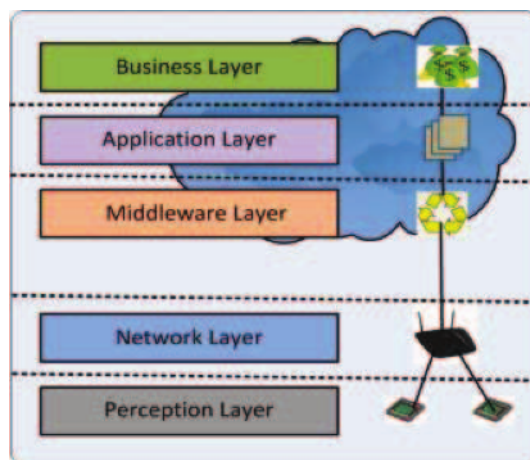


Figure 2.1: Internet of Things layers [Aazam and Huh \(2014\)](#)

The purpose of the Perception layer is to identify the object and perceive the data from the surrounding environment. As described by [Aazam and Huh \(2014\)](#) the machine data is collected on this layer by sensors including bar code labels, Radio-frequency identification (RFID) tags, Global Positioning System (GPS), and cameras along with many other sensor options. The Network layer collects the data from the lower layer, the Perception layer, and then sends to the Internet. This layer includes one interface connected to the sensor network and another to the Internet. The Middleware layer receives data from the Network layer, acting as a service manager and storing data. It also performs information processing before it passes the output to the Application layer. After receiving the information the Application layer presents the final set of data. Data is presented in different formats depending upon the goals of the system. Variables include the type of devices in the Perception layer, the way the information has been processed by the Middleware layer and also the needs of users. The final layer, the Business layer, is about offering a service to make money from the data obtained. A meaningful service is created to turn the data into knowledge that can be offered to businesses and users.

In order for the IOT to turn raw data into information, knowledge and wisdom, as per the DIKW pyramid [Frické \(2009\)](#), the data must be processed. Due to the limited resources on IOT sensor nodes scalable Cloud Computing resources currently perform this processing. The volume of data generated by the IOT networks, which then needs to be transmitted via the Internet, raises challenges around the data formats that can be used. The next section will discuss the Data Transmission methods for the IOT.

## 2.2 Data Transmission to the Internet of Things

As identified by [Bormann, Castellani, and Shelby \(2012\)](#) there are three main technologies of the web. They consist of Hypertext Markup Language (HTML), Hypertext Transfer Protocol (HTTP) and Uniform Resource Identifier (URI). [Bormann, Castellani, and Shelby \(2012\)](#) demonstrate how IOTs use only two of these technologies for machine-to-machine (M2M) communication. While the existing methods use the three main technologies of the current Internet further for human communication research has identified a more suitable approaches for M2M communication [Bormann, Castellani, and Shelby \(2012\)](#), [Aazam and Huh \(2014\)](#) These approaches replace HTML with special data formats based around Extensible Markup Language (XML).

HTML is designed for humans, specifically to display data rather than for applications to access the information automatically [Fu and Liu \(2004\)](#). The extra overhead required to make readable page for humans increases the size of the transmitted HTML. XML

was designed to describe the data and not to display the data like HTML. It is a means of carrying data in a software and hardware independent manner.

Efficient XML Interchange (EXI), the compact binary representation of XML, is one alternative to HTML for M2M communication. [Peintner, Kosch, and Heuer \(2009\)](#) show the difficulty in handling XML due to the processing overhead and the verbosity associated with its use. [Peintner, Kosch, and Heuer \(2009\)](#) demonstrates that the optimised representation obtained from EXI is valuable for certain datatypes. [Moritz, Golatowski, Lerche, and Timmermann \(2013\)](#) illustrate the messaging scenarios for the IOT, utilising EXI to obtain the best compression rates. Three scenarios are outlined, from the resource intensive XML/SOAP-over-HTTP option, moving to an improved option of using XML/SOAP-over-CoAP and finally the optimised EXI/SOAP-over-CoAP option.

Another example is Java-Script Object Notation (JSON), a lightweight data-interchange format. JSON syntax is a subset of object notation syntax. JSON data is written as name/value pairs and is completely language independent with other programming languages [Soliman, Abiodun, Hamouda, Zhou, and Lung \(2013\)](#). [Wehner, Piberger, and Göhringer \(2014\)](#) advocate the use of the JSON format in the IOT as it avoids overhead as its flexibility and a light-weight structure. [Soliman, Abiodun, Hamouda, Zhou, and Lung \(2013\)](#) also support the use of JSON as it is faster and easier to deconstruct, which is another requirement on resource constraint nodes. It also allows for the reduction of traffic load and bandwidth usage when compared to full-fledged XML.

[Aazam and Huh \(2014\)](#) also identify the issue of unnecessary data being sent to and from nodes of the IOT. With only one Kilobyte of unnecessary information from each of the 26 billion IOT nodes predicted by Gartner, (2013) by the year 2020 there would be over 27 Terabytes of irrelevant traffic generated. In this scenario, [Aazam and Huh \(2014\)](#) suggest the device is stopped from transmitting information or a gateway device is connected between the Cloud and the IOT. It is proposed that this gateway device do some processing before sending the information to the Cloud, such as organising the information, deciding what type and when to send the gathered data to the internet. The benefit of this type of hierarchy is the better utilisation of network and also cloud infrastructure resources.

The volume of data transmitted on the Internet from each IOT network will reduce when more efficient protocols are developed for use on the Network layer. However, it has been suggested that the current network infrastructure will fall short of the requirements in the near future if the IOT growth predictions are proven correct. The



next section will discuss the current standing of Cloud Computing and how the IOT currently processes and stores the information generated.

## 2.3 Cloud Computing

[Kalagiakos and Karampelas \(2011\)](#) describe Cloud Computing as a technology that allows a consumer to utilise a providers hardware and software on demand while connected to the Internet. [Al-Anzi, Salman, Jacob, and Soni \(2014\)](#) suggests the cloud enables a convenient, on demand and scalable network access to a shared pool of configurable computing resources. Such a resource pool has been acknowledged by [Zhou, Leppanen, Harjula, Ylianttila, Ojala, Yu, Jin, and Yang \(2013\)](#) as a requirement for the IOT. Resource limitations on the IOT devices mean challenges exist in computational and storage performance. With this, [Zhou, Leppanen, Harjula, Ylianttila, Ojala, Yu, Jin, and Yang \(2013\)](#) introduce the term Cloud-based Internet of Things. This sees Cloud Computing providers introducing services for the IOT. [Zhou, Leppanen, Harjula, Ylianttila, Ojala, Yu, Jin, and Yang \(2013\)](#) highlight a number of these services, from an open source data logging service provided by Nimbits , to the Application Programming Interface (API) provided by ThingSpeak. This API is used to store and retrieve data from devices via HTTP over the Internet. These different services show the potential for Cloud Computing in the IOT domain. A standardised approach to the architecture and protocols used will allow developers across the IOT field to develop applications and services that can be utilized on exiting Cloud Computing infrastructure [Zhou, Leppanen, Harjula, Ylianttila, Ojala, Yu, Jin, and Yang \(2013\)](#).

The data centre based Cloud Computing model discussed in this section provides scalable storage and computational power in a centralised location. However, it has been suggested by Gartner that this architecture does not fulfil the needs of the IOT with regards latency-sensitive applications and high bandwidth usage. Gartner suggests that existing technologies described in this section can be used on IOT network by connecting and integrating existing resources. This approach would exacerbate the issue of network load. A proposed solution to this problem has been suggested called the Edge Cloud. In the next section we will discuss the Edge Cloud paradigm, using resources near the end user in the IOT. The end user may be for example a set of IOT sensors, actuators or a perhaps human user.

## 2.4 Edge Cloud

As the computational power of mobile devices grows, in line with Moores Law [Schaller \(1997\)](#), the ability of devices to compute data natively, at the edges of the Internet, grows. These devices also use the Cloud as a backend. Kryder's Law [Walter \(2005\)](#) suggests the continued growth of storage on these devices. This means that the applications that are hosted on these mobile devices can now perform, to an extent, the processing of the data. With this a reduction in the application latency and bandwidth usage to the data centre can be experienced. Another advantage is the potential improvement in the QoS. Any network outages experienced in the Cloud are not felt locally. The processing of data can continue while the backend core infrastructure is unavailable.

Cisco have proposed the term Fog computing [Bonomi, Milito, Zhu, and Addepalli \(2012\)](#), meaning the Cloud but nearer the ground or nearer to the end user. Cisco suggested that the fog, like the Cloud, provides data, compute, storage, and application services to the end-user however, it is not centralised, rather it is distributed geographically, nearer the user reducing the impact on real-time IOT applications, such as industrial automation. Not all the raw data is transported to the centralised Cloud, only the important pieces of information. In order to determine what is important there needs to be a means of analysing the raw data nearer the source. Again this helps reduce bandwidth and network load.

While Fog computing can provide a degree of computation to the IOT network there may still be a need for further process power to help with large computational tasks that cannot be performed on the Edge nodes. For this the scalable resources in the centralised Cloud are still required. This means that a method of integrating the Edge Cloud with the Cloud is needed. A challenge exists in deciding when to request the resources on the cloud. It is possible to call on further resources only when reaching the limits of the edge nodes or perhaps when a message is received to trigger the resource request. However, as [Ajila and Bankole \(2013\)](#) suggests there is a possibility to learn from past data as to the optimal time to call on the extra resources provided by the cloud. As the example from [Ajila and Bankole \(2013\)](#) indicates, where the Cloud itself creates further resource requests, Machine Learning is a viable method for this process.

## 2.5 Machine Learning

Microsoft introduced Machine Learning (ML) predictive analytics to its Cloud Platform called Azure in 2015. But what is ML? ML provides computers with the ability to learn without being programmed with the explicit code required to understand the task at hand. It is a type of artificial intelligence (AI) focused on allowing programs to adapt and grow over time with the continued exposure to data. The end goal is to allow the machine to improve its own performance [Li, Gibson, Ho, Zhou, Kim, Buhisi, Brown, and Gerber \(2013\)](#) The machine learns from the training data.

A number of ML uses and techniques have been identified in recent research as suitable candidates for use in Cloud Computing. From supervised machine learning algorithms used in Cloud pricing models as suggested by [Ajila and Bankole \(2013\)](#) to other suggestions from [Nikraves, Ajila, and Lung \(2014\)](#) who propose using ML to predict the required provisioning of resources on Cloud infrastructure. The research by [Nikraves, Ajila, and Lung \(2014\)](#) further suggests the best method to be Support Vector Machine (SVM) in providing the best prediction model for required resources. [Simjanoska, Ristov, and Gusev](#) illustrates that the current proactive auto-scaling approach is used to overcome cost versus performance trade-offs in the cloud. [Ajila and Bankole \(2013\)](#) also suggest that SVM is a more accurate method there is also a indication that Neural Networks (NN) can also perform well in the regards the scaling decisions for the Cloud. [Ajila and Bankole \(2013\)](#) findings that NN and SVM are the most effective means to predict future system characteristics is also supported by the research from [Simjanoska, Ristov, and Gusev](#) who also found SVM to be an effective method of ML in the Cloud. While these ML techniques provide an effective means of resource scaling in the Cloud.

With the research from [Simjanoska, Ristov, and Gusev](#), [Ajila and Bankole \(2013\)](#), and [Nikraves, Ajila, and Lung \(2014\)](#) highlighting the use cases for ML in the Cloud, and also articles from Microsoft Research regarding Machine Learning in the Azure Cloud it is clear that advantages of Machine Learning are becoming evident in relation to computation of enormous data on centralised Cloud infrastructure. However, a ML technique can also be adapted for use in the Edge Cloud. Future research will need to be done to identify the suitability on the resource constrained Edge Clouds.

# Chapter 3

## Design

### 3.1 Design Overview

The applications will save metrics while retrieving and processing data from an IOT network of devices. The results from the Edge Cloud will then be compared against baseline results obtained by running the application on a conventional Cloud infrastructure located in a centralised datacenter. The problem this prototype application addresses has been identified and analysed in the literature review. This chapter will discuss the design decisions, technologies to be used in the prototype and their roles in the overall solution. In [section 3.5](#) we shall then discuss the use-case to be implemented in order to demonstrate how the solution would improve on the current conventional Cloud implementation.

### 3.2 Motivation

As identified in the research chapter of this document there has been a number of articles suggesting the Edge Cloud as a means of improving internet bandwidth usage in M2M and IOT applications. However there has been no research conducted to compare and analyse the actual outcome of the different scenarios. The aim of this prototype application is to investigate the suggested benefits in using decentralised Cloud resources located near the data in regards to the bandwidth usage and data latency. The prototype application consists of three main sections. The IOT component which acts as a source of data, the Cloud component which stores and performs computation on said data and the Predictive Analytics component which predicts the likely outcome based on past events. This prediction which can then be used by the compute resource

to decide on what actions are to be taken. In this section, following a brief discussion on the current architecture, the design decisions of each of the components will be discussed in more detail.

### 3.3 Current Solution

In order to compare the results of using the Edge Cloud the application will first be run on the conventional Cloud architecture. The current architecture is outlined in [Figure 3.1](#) As can be seen the data generated in the IOT Sensor Network must traverse the Internet to reach the centralised Data Centre for processing, storage and forwarding to the ML component for analysis. The results from the Predictive Analytics are then fed back to the Data Centre and must again traverse the Internet to return to the IOT Sensor Network or any other designated end point.

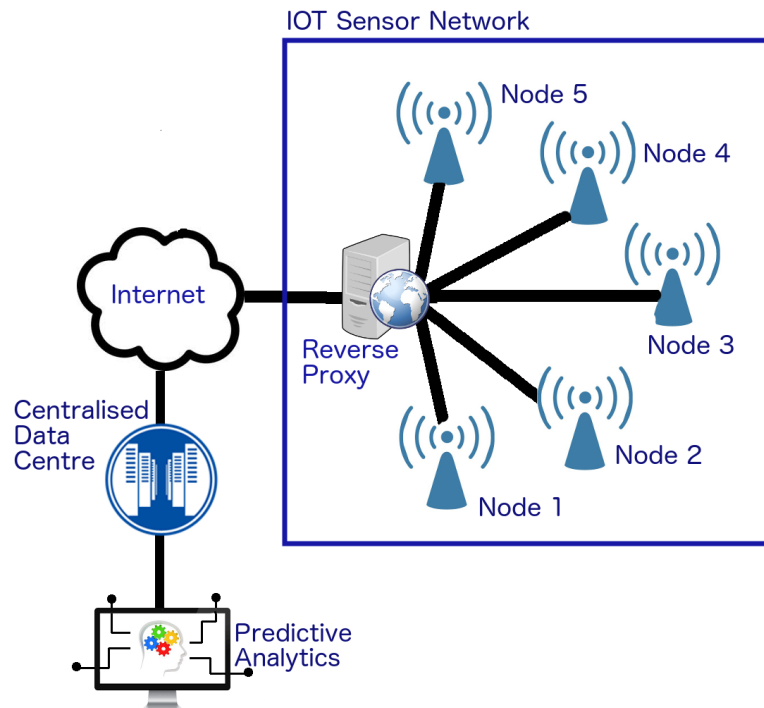


Figure 3.1: Current Centralised Cloud using Predictive Analytics

This current implementation of the distributed resources can span the globe. For example the Azure ML web service currently only resides in the US South Central data centre. As can be seen from the [Figure 3.2](#) the requirement for Internet bandwidth is apparent.

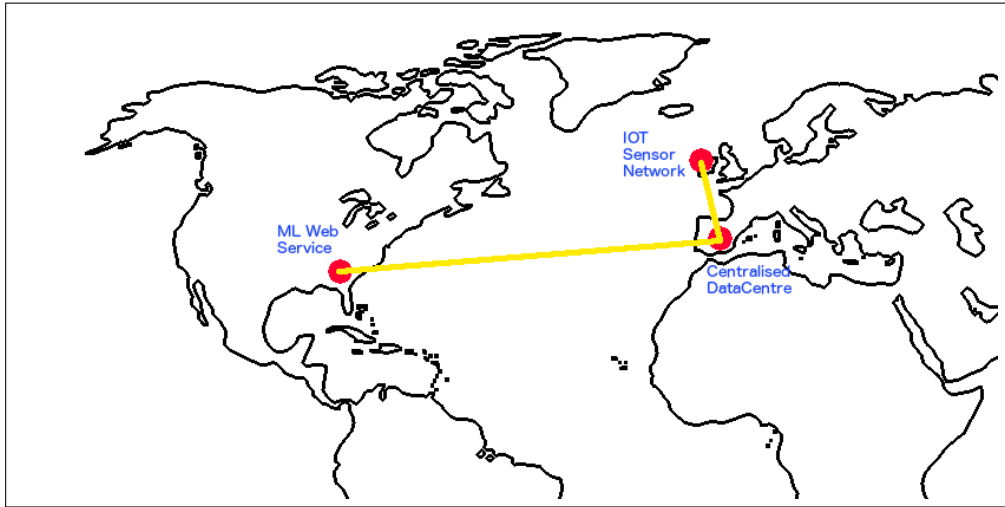


Figure 3.2: Centralised Cloud and ML resources

### 3.4 Proposed Solution

Figure 3.3 outlines the proposed design of the application using the Edge Cloud and co-located Predictive Analytics module to improve on data latency and Internet bandwidth usage.

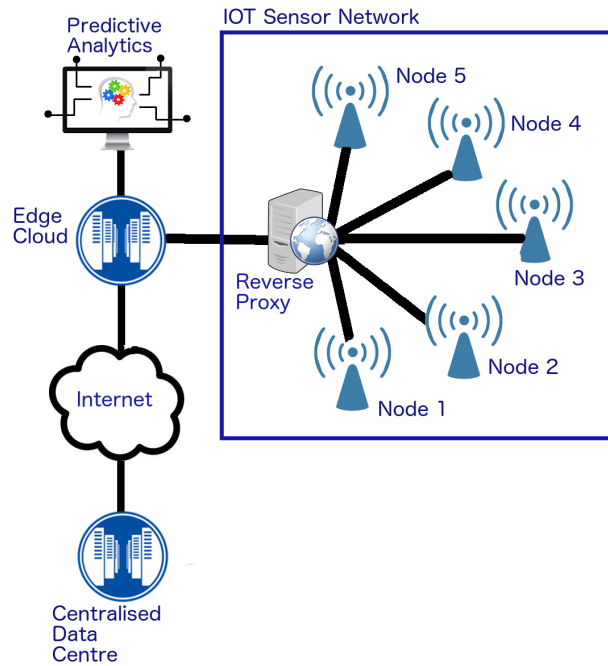


Figure 3.3: Proposed Edge Cloud using Predictive Analytics

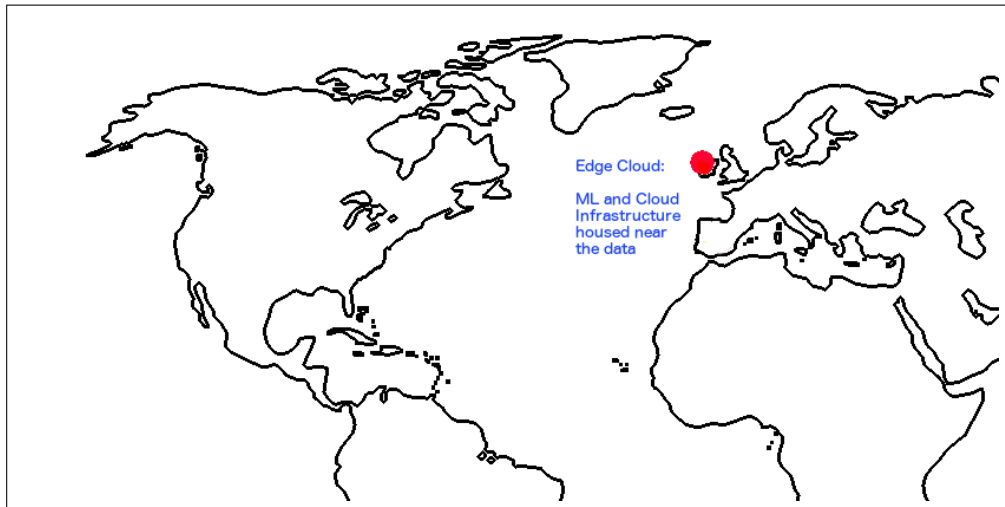


Figure 3.4: Edge Cloud - resources co-located near the Data Source

As per [Figure 3.4](#) the data does not need to travel to the centralised Cloud in order to be processed. In the following sections we will discuss each of the components of this proposed design in more detail.

### 3.4.1 Data Source - Internet Of Things

As discussed in the research section of this document the Internet Of Things will become the source of large amounts of computer generated or M2M data. In this application the data generated by each of the Nodes in the IOT will be a Json string. As has been identified in the research section of this document Json syntax has been highlighted as a suitable lightweight data-interchange format for the IOT due to its flexibility and light-weight structure. This method has been selected over alternatives such as XML as it is faster and easier to deconstruct on power restraint devices. Each node will generate sensor data in this Json format which will then be sent to the origin of the request. In testing this application the data generated will be controlled by number of requests that have been received by each node. There will be a threshold at which point the data produced will change. With this change the expected results from the Machine Learning will adjust to produce the required result. The computation resource will process the results and take any action that may be required. The possible actions will be to either save the data locally or to send the data to the conventional cloud over the internet for further analysis. The amount of data to be sent over the internet will be logged for comparison between the tests using the Edge Cloud and the tests using only the Conventional Cloud.

IPv6 has been selected for this application as a suitable protocol to identify the IOT devices. The IPv6 protocol will allow for the required growth in the number of connected devices predicted to join the internet in the coming years. The length of an IPv6 address is 128 bits, compared with 32 bits in IPv4. The address space therefore has  $2^{128}$  or approximately  $3.4 \times 10^{38}$  addresses versus the  $4.3 \times 10^9$  addresses in IPv4.

The IOT design consists of the border router utilising the IPv6 Routing Protocol for Low-Power (RPL) and Lossy Networks [Winter \(2012\)](#) technology. The border routers, or gateways, are routers that can be found at the edge of a network. They allow one network, such as our simulated IOT network, to connect to another network, such as the internet. It also identifies the routes to the nodes.

The final aspect in this design of the architecture of the IOT network is the means of connecting from the Cloud to the devices on a simulated network. Each device will have its designated IPv6 address, however this address will be behind a firewall on the VM hosting the simulation. In order to access the required node using its IPv6 address the host VM must forward the request to the correct node. This will be done using a Reverse Proxy Server running on the VM hosting the IOT simulation. Note that when this architecture is used with real world devices the IPv6 address will allow for the direct connectivity between the device and the internet. There will be no reverse proxy required.

The proposed design of the IOT sensor network is shown in [Figure 3.5](#).

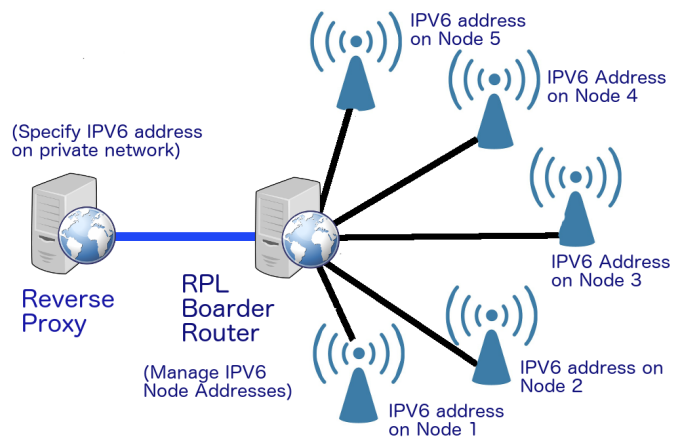


Figure 3.5: Reverse Proxy and Boarder Router in IOT Network



### 3.4.2 Predictive Analytics

Another congestion point in the processing of data from the IOT network is the human ability to make sense of the data that is being generated. This leads towards a need to find a means of effectively acting upon the data coming from the IOT. The predictive analytics component will utilise machine learning to help predict future outcomes based on the current IOT sensor data combined with external information. While alternatives such as hard coded trigger points can be used to decide when to send data to the Cloud the design decision to use predictive analytics solution gives greater efficiency and effectiveness in dealing with changing data. An initial set of training data will be produced to represent historic data from the IOT network. This training data will include the target value or information we wish to predict. These target values have been recorded during past events. The aim of the predictive analytics component of the application is to predict this target values without having the full set of data. The target value is dependant on input values, all of which are used to determine patterns and predict future outcomes and trends.

### 3.4.3 Computation and Storage

As identified in the research section of this document the data generated on by the IOT must be stored and processed. Also identified in the research section is the continued reduction in price and size of storage infrastructure and the increase computational ability of mobile devices. This means the Edge Cloud becomes a more attainable solution. For this application the Edge Cloud operations, required to perform tasks or computation on the data, will be performed on hardware housed near the IOT network itself. This will reduce the network latency of the information and reduce the internet bandwidth required. In an industrial application of this architecture the data would communicate over the Local Area Network, from the sensor nodes to the Edge Cloud computation and storage resources housed near the sensor network. To test the hypothesis the application will be housed in a data centre along with the IOT simulation. This will simulate the reduced span between the Cloud and the sensors.

## 3.5 Use Case

The Use Case identified for the prototype application is a wind farm in the north west of Ireland as shown in [Figure 3.6](#). It comprises of 20 turbines and a maintenance and storage building suitable to house a number of compute, storage and ML infrastructure

resources. This is a good example of an industrial application which can benefit from this technology. These wind farms are remote and clustered together utilising a number of sensors to generate different metrics such as energy output and efficiency.

The proposed architecture above collects data from the source, makes sense of it and generates decisions as an output. All of which happens with limited human involvement. Again, this lends itself to the wind farm use case. The turbines have generated historical data based on the sensor information and status. [Table 3.1](#) outlines the historical data points that will be used in training the ML module. The ML Input values, retrieved from the IOT nodes, will then be used to predict the ML Output value.

<b>Machine Learning Inputs and Outputs:</b>	
<b>ML Input:</b>	Turbine number
<b>ML Input:</b>	Windspeed
<b>ML Input:</b>	Wibrations
<b>ML Input:</b>	Turbine speed
<b>ML Input:</b>	Energy output
<b>ML Input:</b>	Efficiency
<b>ML Output:</b>	Turbine status

Table 3.1: ML Inputs and Outputsw

When combined, these values can be used to generate a historical picture. In this use case there are three different conditions each turbine can be in as outlined in [Table 3.2](#) It is expected one of the following will be returned for the corresponding inputs.

<b>Turbine Status:</b>	
<b>0</b>	The turbine is broken and needs maintenance.
<b>1</b>	The turbine is working at a reduced capacity and needs maintenance
<b>2</b>	The turbine was working well and required no further action

Table 3.2: Turbine Status Codes and expected ML Output values

With this historical data it is possible to generate a set of test input data that, when combined with third party test data such as weather forecasts, could predict which turbine may need maintenance in the near future. The windspeed, recorded for the training data, could be replaced with windspeed forecast. This, along with the other available sensor data, can provide an insight as to when action should be taken by the computational resource.

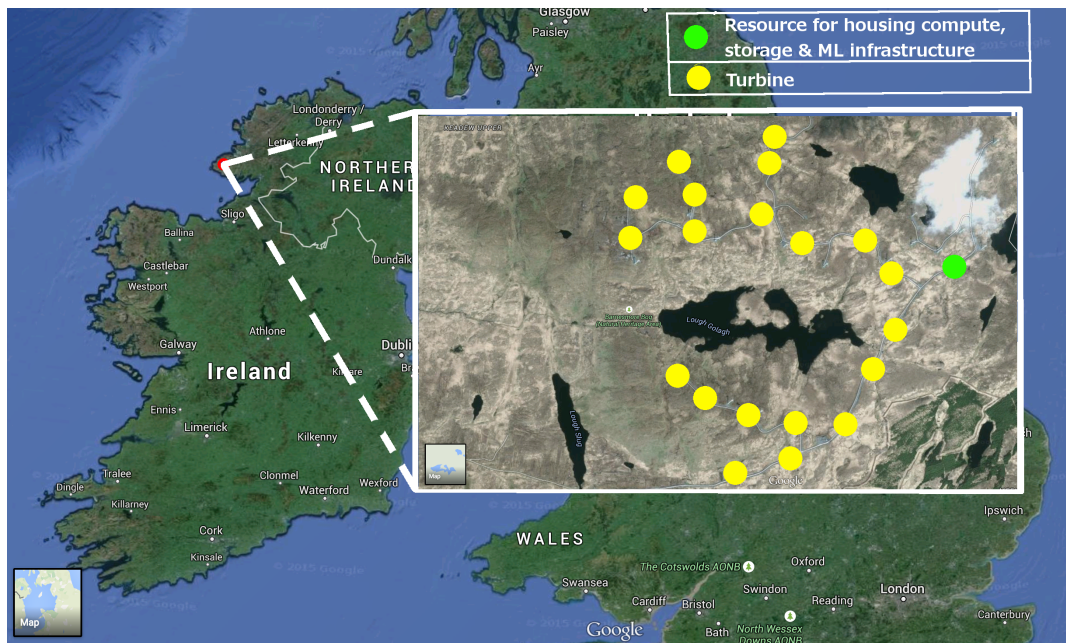


Figure 3.6: Wind Farm use case for Edge Cloud

## Chapter 4

# Implementation

This chapter provides information on the implementation of the prototype application. The implementation of the IOT simulation is presented in [section 4.1](#). The set up of Reverse Proxy is detailed in [subsection 4.1.3](#). The workings of the Predictive Analytics is describe in [section 4.2](#) and the Cloud Resources is outlined in [section 4.3](#).

### 4.1 Data Source - Internet Of Things

The technologies used in the implementation of the IOT sensor network are discussed in this section. In order to deploy the sensor on a low power device the [Contiki-os](#) was identified as a suitable solution. Contiki allows for the testing of node software on a simulated environment prior to deployment onto costly hardware. Different hardware configurations have been emulated on the cooja which means that any developed software can be used in future hardware implementations.

#### 4.1.1 Contiki-os

Contiki-os has been identified as a suitable operating system to be hosted on the IOT nodes as it allows low-cost, low-power micro-controllers to be connected to the internet. In order to create a larger number of IOT devices the prototype application will use the Contiki simulator called cooja. This will allow the application to add IOT devices without the need for costly hardware.

Contiki-os uses the Constrained Application Protocol for machine-to-machine communication, allowing simple RESTful operations to be performed, such as the GET, POST, PUT and DELETE methods.

The following instructions show how to setup the Contiki toolchain in a freshly installed Ubuntu 13.04. In order to install the Contiki-os on ubuntu the dependant packages must also be installed. These packages can be installed with the following command:

```
1 sudo apt-get install build-essential binutils-msp430 gcc-msp430 msp430-libc binutils ↔  
-avr gcc-avr gdb-avr avr-libc avrdude openjdk-7-jdk openjdk-7-jre ant ↔  
libncurses5-dev doxygen git
```

The Contiki sources can then be downloaded from Github using the following command:

```
1 git clone --recursive git://github.com/contiki-os/contiki.git contiki
```

### 4.1.2 Cooja - The Contiki Network Simulator

Contiki devices often make up large wireless networks. Developing and debugging software for such networks is really challenging. Cooja, the Contiki network simulator, makes this tremendously easier by providing a simulation environment that allows developers to both see their applications run in large-scale networks or in extreme detail on fully emulated hardware devices. In order to view Cooja the following commands are run:

```
1 cd tools/cooja  
2 ant run
```

With the Cooja simulator running a new simulation can be created. Open menu:

**File » New simulation » Create**

A simulation with the default parameters is created. We first need to create a mote type. The mote type determines which Contiki applications are to be simulated. First off we must add the RPL Boarder Router. Open the menu:

**Motes » Add Motes » Create new mote type » Cooja Mote Type**

A dialog allowing you to configure the new mote type appears. Enter a suitable description, such as: "Boarder Router". Click **Browse** and select the boarder router application that will run on this mote. Open the menu:

**Motes » Add Motes » Create new mote type » Cooja Mote Type**

Enter another suitable description for the motes: "Senor Nodes"

Click **Browse**, and select the application that will run on the motes. This is a c file write to generate the Json response for the the GET request from the application running

on the Edge Cloud. Next you must compile the Contiki shared library by clicking **Compile**. When the compilation finishes, load the library and create the mote type by clicking **Create**. We have now added a mote type, however, the simulation does not yet contain any simulated motes. In the Add Motes dialog enter the required number of Motes and click **Add motes**. A few plugins are started such as, a control panel for starting and pausing the simulation, a visualiser that shows the node positions, and a log listener showing printouts of all simulated nodes. Press **Start** in the Control Panel plugin to start the simulation as per [Figure 4.1](#).

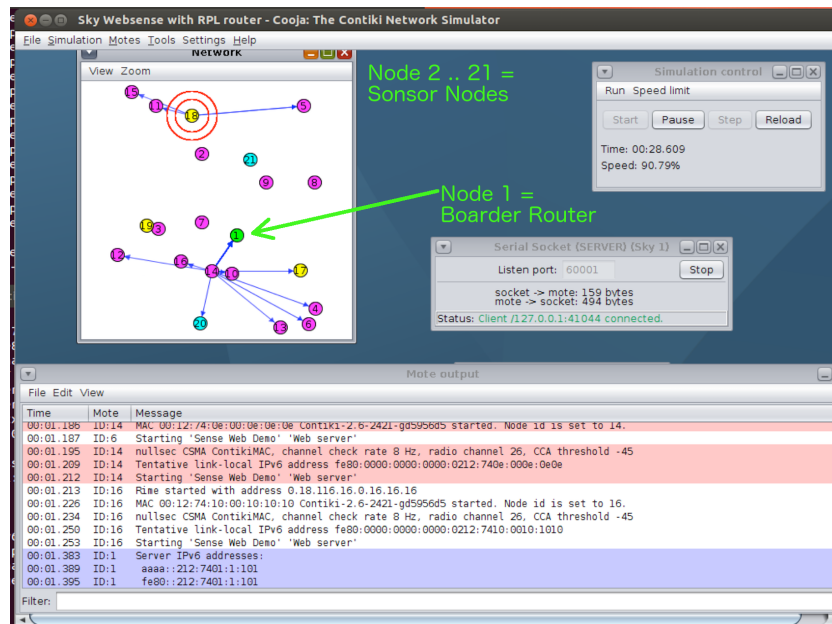


Figure 4.1: Cooja Simulator running 20 sensor nodes and 1 boarder router node

As can be seen in [Figure 4.1](#) there is a single boarder router, denoted as node 1. In order to test both the Conventional Centralised Cloud environment and the Edge Cloud environment there was a need to have a consistent dataset. The sensor nodes, from node 2 to node 21, have been identified as producing data that will place the status in the one of the three possible conditions as outlined in the use case in [Table 3.2](#).

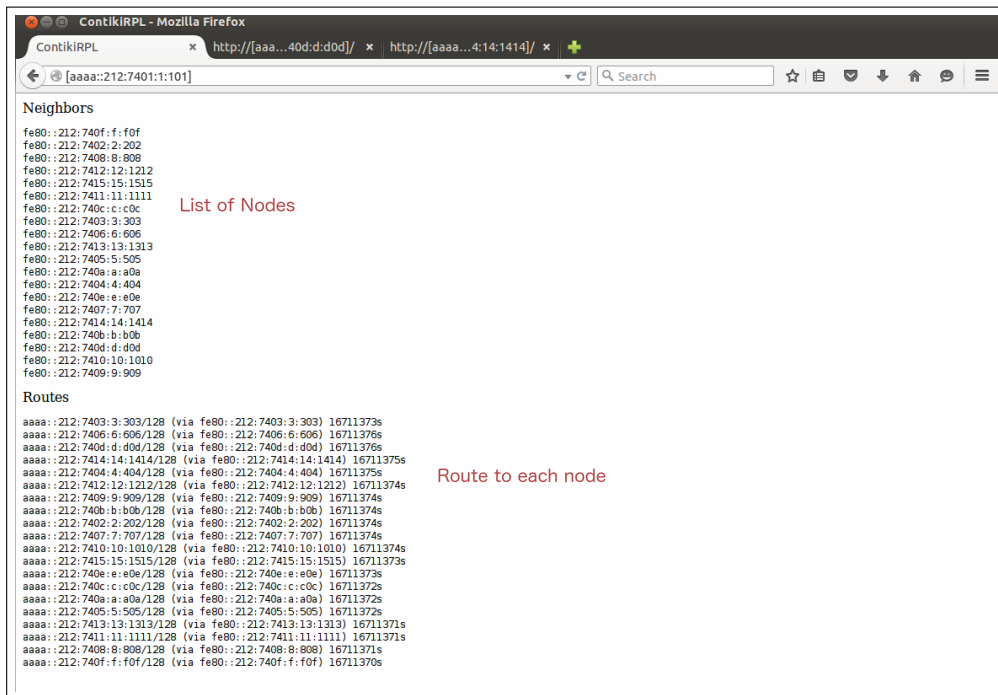


Figure 4.2: Boarder Router web page showing Nodes and Routes

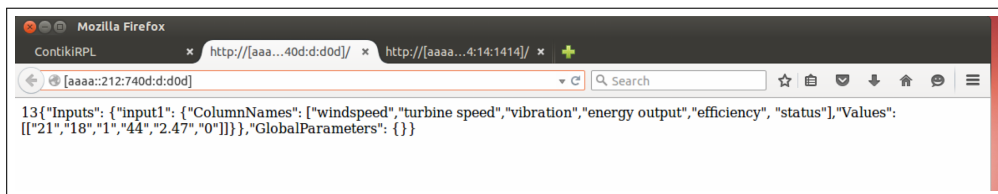


Figure 4.3: Sensor Node showing Json String

### 4.1.3 Reverse Proxy

As discussed in the subsection 3.4.1 the reverse proxy is required in the implementation of this prototype as the simulation is run on a VM behind a firewall. In order to access this internal IPV6 address on the VM a means of routing the request to the correct node running a web server is required. The solution implemented to address this issue was a Reverse Proxy Server. Using this solution all incoming requests can be directed to the VM IP address and correct port for the Reverse Proxy Server. It will then make a request to the IP address of the necessary Node. The URL parameters of the originating URL will be used to identify the IPV6 address. For Example:

1 `http://23.102.177.186:8080/aaaa::212:7401:1:101/`

Here the VM IP address is 23.102.177.186, the Proxy Server is running on port 8080 and the IPV6 address of the boarder router is aaaa::212:7401:1:101. As per [Figure 4.2](#) the boarder router will then list the nodes and the routes to these nodes. Data from each node can then be retrieved by using the IPV6 addresses listed to return the node data as per [Figure 4.3](#).

The technology used for this proxy is [Node.js](#). Node.js was selected as the Reverse Proxy Server as it is a lightweight and efficient solution that uses an asynchronous event driven framework with a non-blocking I/O model meaning many connections can be handled concurrently. Each request from the Cloud can be handled independently, returning the sensor data separately from each other. Other alternatives such as Apache web server were considered however with Node.js support of web sockets and the greater number of concurrent requests compared to apache it was selected as the best solution to the increasing number of requests that may be expected in the future. The implementation of the Node.js Proxy Server can be seen in [Listing A.1](#).

## 4.2 Predictive Analytics

In this section we discuss the Predictive Analytics implementation in the prototype application using the Microsoft Azure ML web service ([AzureML](#)). Other solutions such as the Amazon Machine Learning service ([Amazon](#)) were also considered, however the Azure web service was selected to ensure the VM and ML components could be located in the same data centre, supporting the premise of moving the application to the data. Another design option considered was creating a local ML implementation, however due to time constraints this was not possible and thus has been highlighted in as future work in [subsection 6.2.1](#). In order to test the performance of having a co-located Predictive Analytics agent residing alongside the Cloud infrastructure the Azure Machine Learning service will be used. In order to emulate the Edge Cloud setup this service will then be located in the same datacenter as the Cloud and the IOT simulation.

### 4.2.1 Azure ML Web Service

The resource provides an end to end visual workflow to create experiments using existing data. Using the Azure studio the user can use the drag-and-drop tool to build, test, and deploy predictive analytics solutions. Using the existing historical dataset the ML service trains the machine learning to produce a model that can predict what results



future data will produce. The list below shows the workflow for the Azure Machine Learning training experiment:

1. The historical wind farm dataset is fed into the "Project Column" which identifies the columns to be used in the ML model.
2. The output is then fed into the "Split" which takes a specified percentage of the dataset to be used as training data and the rest to be used as test data for the "Score Model"
3. The column which is to be predicted is identified in the "Train Model" module which uses the "Linear Regression" solution method to predict the outcome.
4. The "Score Model" then scores the data and predicts the outcome.

Figure 4.4 shows the training experiment in the Azure ML portal.

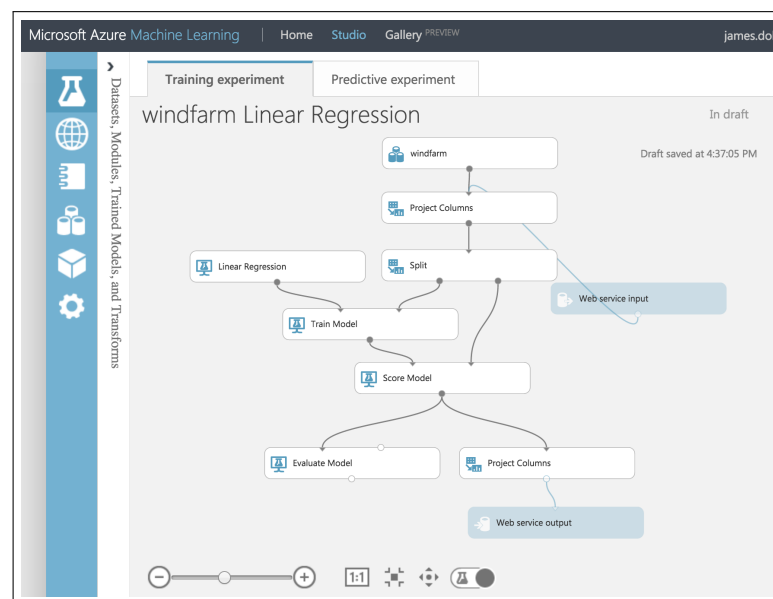


Figure 4.4: ML Training Experiment

The Training Experiment can then be published to produce a web service as seen in Figure 4.5. The input of the web service is a Json string and is fed into the score model to predict required column, as selected in the training experiment.

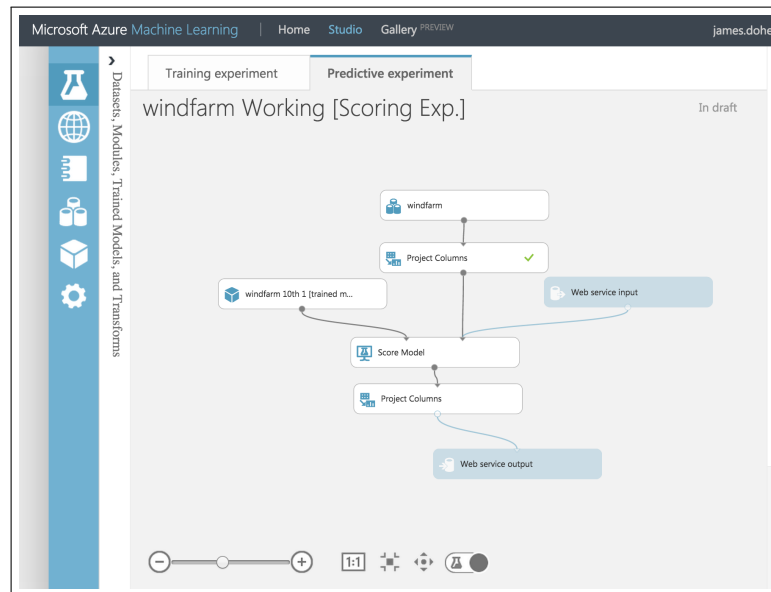


Figure 4.5: ML Web Service

The web service can then be accessed via python code from the Edge Cloud resources as per [Listing A.1](#)

## 4.3 Cloud Resources

To implement the Edge Cloud near the IOT network the Microsoft Azure resources have been selected. The Azure resources will allow the IOT simulation to run on a VM in the same data centre as the Edge Cloud and ML components. This Edge Cloud will gather, store and act upon the data from the IOT sensor network. To gather the sensor data the Cloud will use REST. REST is a software architecture style for exposing resources which will reside on the Contiki-os nodes in the IOT network. The technology used to retrieve the resource is [Python](#). Python is a open-source, powerful and fast language.

### 4.3.1 Data Retrieval

To retrieve the data from the IOT simulation the python code uses the urllib2. This allows for a connection to be opened to the URL of the boarder router.

```

1 import urllib2
2 connection = urllib2.urlopen('http://23.102.177.186:8080/aaa::212:7401:1:101/')
3 body = connection.read()

```

The IPV6 address of each sensor node can then be parsed out from the list supplied by the boarder router in the body of the response. A request is then sent to each IOT node to retrieve the data. In order to calculate stats on the performance of the request, such as on how long each takes, the time is saved before and after each request. This will be used in the analysis section of this document to understand the differences in using the co-located Edge Cloud versus the centralised Cloud. The times regarding the request sent to the ML web service are also logged. The improvement in execution time in using the local ML agent versus the centralised ML agent will be analysed in the analysis chapter.

### **4.3.2 Data Processing**

As described in [section 4.3](#) the ML component is implemented as a web service which allows the Edge Cloud resources to use REST to retrieve the predicted outcome based on the input data. The Cloud can then act on these results. For this prototype application the data will be forwarded to the scalable cloud when the turbine status is predicted to be either "Broken" or "Needing Maintenance". In the event of either of these statuses being returned from ML the associated data from the node is to be forwarded to the centralised, scaleable cloud for further analysis. This approach, it is hoped, will see a reduction in the amount of data that is required to be sent to these centralised, scaleable cloud resources. Only important data needing additional analysis or further action is sent. The data sent to the centralised cloud is saved locally and compared with the total amount of data received which is also saved locally on the Edge Cloud.

## Chapter 5

# Evaluation

In this chapter the experiments conducted on the prototype application are outlined and results are evaluated against the hypothesis. This evaluation will help determine the potential gains of using the Edge Cloud against the traditional centralised Cloud.

### 5.1 Test Case Outline

To compare the Edge Cloud performance with the traditional Centralised Cloud the experiments are conducted on two environments. The environment setup is detailed in the subsequent sub sections. The test case that will run on both the Edge Cloud and the Conventional Cloud will be as follows:

1. Open the IP address of the Boarder Router.
2. Retrieve the list of IPV6 addresses for the 20 nodes.
3. Record time taken to retrieve data from the node.
  - 15 nodes have been designed to remain in a status of "Working Well" (ML returns 2) during the full test cycle.
  - 3 nodes have been designed to enter a status of "Maintenance Required" (ML returns 1) after 60 request iterations in the test cycle.
  - 2 nodes have been designed to enter a status of "Broke" (ML returns 0) after 75 request iterations in the test cycle.
4. Send data to the ML component.

5. Record time taken to retrieve ML results.
6. Determine the status of the node based on the results from the ML.
7. Determine the action to be taken for the node based on the status.
  - node is working well, store data locally as analysis is complete and no further action is required.
  - node is in need of maintenance, store data locally and forward the data to the Cloud for further analysis and actions.
  - node is broken, store data locally and forward the data to the Cloud for further analysis and actions.
8. Repeat the process 100 times to generate a data set of 2000 data points.

## 5.2 Test Environments Setup

Each test environment is setup as follows:

- VM running the IOT simulation.
- VM with the compute and storage infrastructure.
- ML web service hosted by Azure.

Depending on the type of Cloud, either the Edge Cloud or Centralised Cloud, the location of each VM will be migrated to the correct data centre as outlined in the subsequent sub sections.

### 5.2.1 Edge Cloud setup

The Edge Cloud test environment will see all components located in a single data centre. The Contiki cooja simulation hosting the IOT sensor network will run alongside the VM representing the Edge Cloud compute and storage infrastructure. The data centre selected for this is the US South central data centre as it also is the data centre housing the Azure ML web service. The test case identified in [section 5.1](#) will then be run against this environment.

## 5.2.2 Conventional Cloud setup

The VM housing the IOT network simulation for the Edge Cloud will be copied to blob storage. This VHD image will be used to create an identical test environment on another VM located in Europe. This will represent the Centralised Cloud. The IOT network will remain in the VM in US South Central. As the ML component is a service operated by a third party provider it will also remain as in US South Central as it cannot be migrated to any other data centre. The timing of all requests and responses is then logged at each point and a comparison will be drawn between each test environment in [section 5.3](#).

## 5.3 Results Analysis

In order to confirm or refute the hypothesis this section will evaluate the Bandwidth and the Application Latency results between each test environment.

### 5.3.1 Analysis of Bandwidth Usage

In this section we analyse the data that was generated during the test. This will represent the required bandwidth for transferring the generated data in both the Edge Cloud and the conventional Centralised Cloud setups. As expected, the amount of data generated in both the Edge Cloud test and the Centralised Cloud test were the same. This means that the IOT network was consistent in both scenarios. [Table 5.1](#) and also [Table 5.2](#) show the same results found in the two test environments. While in the Edge Cloud environment only the results outlined in [Table 5.2](#) would be sent via the internet, meaning a significant reduction in internet bandwidth usage in the long term. These results will be discussed in more detail in [subsection 5.3.2](#).

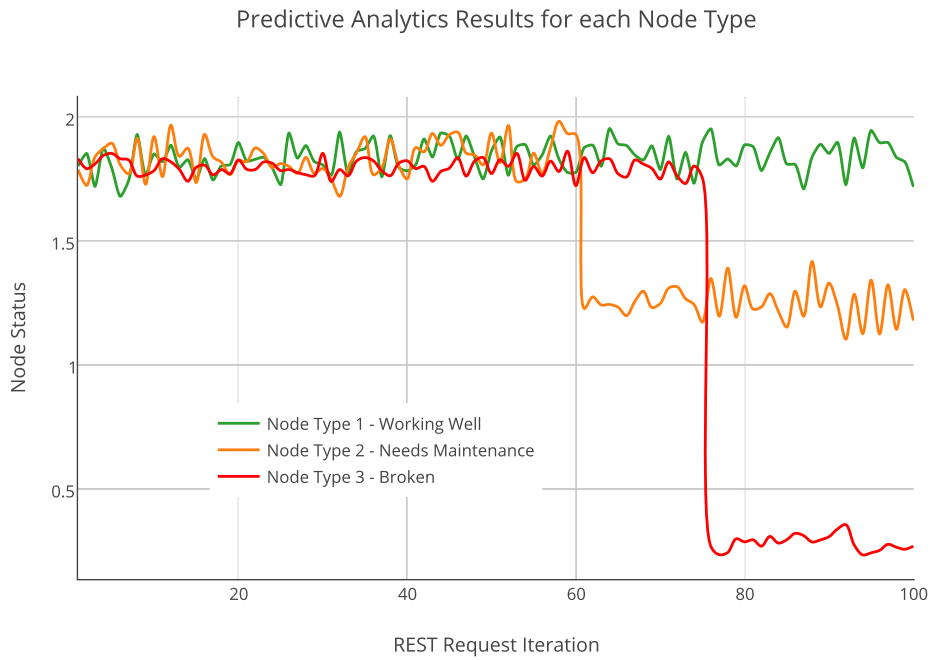


Figure 5.1: Results of ML for each type of Node

Figure 5.1 gives a sample set of results from the ML module for single instance of each type of node. It can be seen that in this test case the ML module returns a status of greater than 1.5 until request iteration number 60. On request iteration number 60 Node Type 2 then returns a result of less than 1.5. Node Type 3 returns a result of less than 0.5 after request iteration 75. Again these test results were consistent for both the Edge Cloud and Conventional Cloud setups, showing that the IOT VM produced consistent results.

At request iteration number 60 that the Edge Cloud begins to forward data to the Centralised Cloud for further processing or action. Prior to request iteration 60 all nodes are working well and required no further action, meaning no Internet bandwidth was used during this period. This result would support the hypothesis that the Edge Cloud, used in conjunction with centralised Cloud resources, would see improvements in the utilisation of Internet bandwidth measured against the solitary use of centralised Cloud resources.

<b>Total File Size:</b>	<b>Number of Nodes:</b>	<b>Request Iteration:</b>	<b>Total Number of Data Points:</b>
223KB	20	100	2000

Table 5.1: Total Data Set

<b>Data Size Sent to Cloud:</b>	<b>Number of Nodes:</b>	<b>Total Number of Data Points:</b>
19KB	5	170

Table 5.2: Data Set sent to the Cloud

As shown in [Table 5.1](#) and [Table 5.2](#) the test scenario shows that it is necessary to send only 8.5% of the total dataset to the Centralised Cloud. This saving of 91.5% would translate into a significant reduction in the bandwidth used in the test scenario for the wind farm application.

[Table 5.3](#) outlines the amount of requests made by the Edge Cloud application per hour as well as the amount of data transferred per request. From this data we have extrapolated the total amount of data transferred per node per year as well as the total amount of data that would be transferred per year for this wind farm use case had all data been sent to the Cloud.

<b>Prototype Application results:</b>	
<b>Data points / node / hour:</b>	595
<b>Data points / node / day:</b>	14285
<b>Bytes / datapoint:</b>	111.5
<b>Bytes / node / day:</b>	1592857
<b>GBytes / node / day:</b>	1.592
<b>GBytes / node / year:</b>	581.08
<b>TBytes / year (20 nodes):</b>	11.6216

Table 5.3: Prototype Application Results:

### 5.3.2 Analysis of Request/Response Times

In this section we analyse the results of the request/response times in both the Edge Cloud and the conventional Centralised Cloud setup. As can be seen from [Table 5.4](#) the



Edge Cloud demonstrates improvements when compared to the Conventional Cloud in individual request/response execution times and thus also in the total execution time of the test case. Using the co-located Edge Cloud approach, it has been shown that the median request/response time for data retrieval from the IOT simulated network has improved by 0.2536 seconds, while the median request/response time for the co-located ML module gives improvements of 0.4565 seconds. Together this equates to over 0.7 of a second improvement from initiation of the request for data, to the time that action can be taken by the compute resources. While this prototype use case may not have identified a scenario where such time savings would benefit the application, other use cases, such as time critical or safety critical application, would see dramatic benefits with an improvement of 0.7 seconds per request.

	Conventional Cloud				Edge Cloud			
<b>Execution Time:</b>	03:48:14				03:22:37			
	<b>IOT:</b>		<b>Predictive Analytics:</b>		<b>IOT:</b>		<b>Predictive Analytics:</b>	
<b>Median: (Sec)</b>	5.838		0.563		5.584		0.1068	
<b>Mean: (Sec)</b>	6.0337		0.5753		5.769		0.1205	
<b>Standard Deviation:</b>	1.5156		0.088		1.2214		0.0579	
<b>95% below: (Sec)</b>	8.53		0.721		7.78		0.216	
<b>95% between: (Sec)</b>	3.002	9.065	0.399	0.752	3.327	8.213	0.005	0.236

Table 5.4: Test Case Results

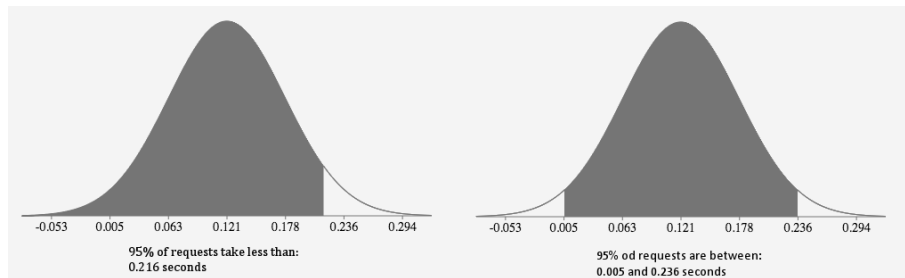


Figure 5.2: Normal Distribution for ML request/response times on Edge Cloud

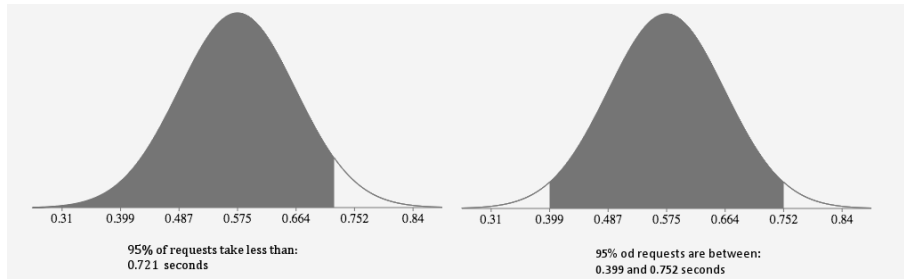


Figure 5.3: Normal Distribution for ML request/response times on Conventional Cloud

As indicated in [Figure 5.3](#) and [Figure 5.2](#) the improvement in the 95th percentile of responses from the ML module is 0.505 seconds. These results support the results from [Table 5.4](#) that the Edge Cloud would see improvements in request/response times when locating the ML resource near the compute and storage resources.

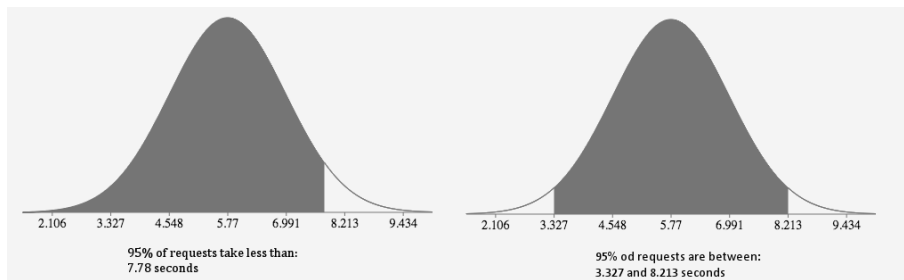


Figure 5.4: Normal Distribution for IOT request/response times on Edge Cloud

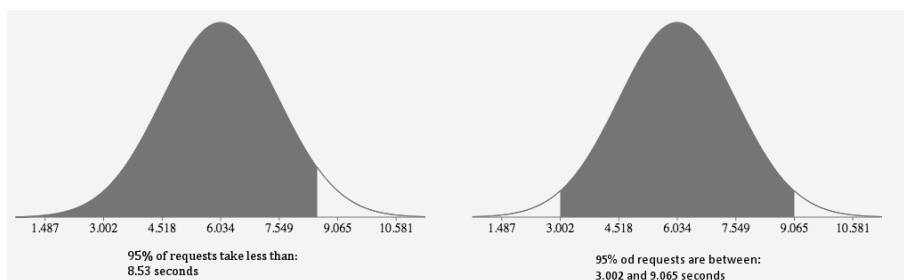


Figure 5.5: Normal Distribution for IOT request/response times on Conventional Cloud

As indicated in [Figure 5.5](#) and [Figure 5.4](#) the improvement in the 95th percentile of responses times from the request for data to the IOT network is 0.78 seconds. These results support the results from [Table 5.4](#) that the Edge Cloud would see improvements in request/response times when locating the compute and storage resource near the IOT data source. These results would support the hypothesis that the Edge Cloud, used in conjunction with ML resources, would see improvements in the application latency measured against the solitary use of centralised Cloud resources.

## Chapter 6

# Conclusion

This research has proposed a novel approach of combining the new Edge Cloud paradigm and machine learning with the aim of improving application latency and reducing internet bandwidth usage. The conventional Cloud sees data brought to the application, however the Edge Cloud sees the application brought to the data. With the number of connected devices in the Internet of things expected to increase significantly in the coming years this approach would see a reduction in both the bandwidth usage and latency of applications as shown by the results from our proof of concept evaluation.

In order to compare the performance of the Edge Cloud environment with that of the Centralised Cloud environment an identical test scenario was performed on each environment setup. The IOT source was simulated and replicated for both scenarios to produce identical test data. Data on time taken to retrieve and process the test data was logged along with the test data itself. Analysis performed identified a significant saving in using the Edge Cloud compared to the Centralised Cloud in both request/response times and Internet bandwidth usage for this prototype application.

### 6.1 Limitations of Proposed Approach

#### 6.1.1 Data Retrieval Technique

An Identified limitation of the proposed solution is the retrieval method used to gather the IOT sensor data. The method implemented is to request the data from each of the nodes, however for future work a more sustainable approach would be to post the data from each node to a queue on the Cloud resources.

## 6.2 Future work

Next I discuss several possible directions for future work.

### 6.2.1 Implement a local ML resource

Rather than using the third party Azure ML resource in the future work a local implementation of the ML module would be created to service the requirements. This would then allow the full architecture to be located on any available hardware infrastructure.

### 6.2.2 Use of Hardware

A future step in testing the benefits in using the Edge Cloud would be the use of real hardware, including nodes and also the compute and storage infrastructure. While this was initially the intention for this thesis the cost of acquiring the hardware proved an obstacle. The utilisation of this hardware would be studied to understand if the Edge Cloud resources would be used at a higher rate of utilisation. After determining the number of nodes that would be used in the use case and calculating the computational requirements of the application during the normal daily operations, it would be expected that the Edge Cloud would see higher utilisation rates, hence removing the concern of resource under utilisation. Again, the Centralised Cloud could be used to perform the required additional analysis in the event of unexpected outcomes.

### 6.2.3 Cost Analysis

Cost analysis would also be conducted to identify any benefits in using the Edge Cloud resources versus the Centralised Cloud. Given, as identified in [subsection 6.2.2](#) the higher utilisation of cheap local resources, a potential cost benefit could also be expected.

# Bibliography

- M. Aazam and Eui-Nam Huh. Fog computing and smart gateway based communication for cloud of things. In *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*, pages 464–470, Aug 2014. doi: 10.1109/FiCloud.2014.83.
- S.A. Ajila and A.A. Bankole. Cloud client prediction models using machine learning techniques. In *Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual*, pages 134–142, July 2013. doi: 10.1109/COMPSAC.2013.21.
- Fawaz S Al-Anzi, Ayed A Salman, Noby K Jacob, and Jyoti Soni. Towards robust, scalable and secure network storage in cloud computing. In *Digital Information and Communication Technology and it's Applications (DICTAP), 2014 Fourth International Conference on*, pages 51–55. IEEE, 2014.
- Amazon. Amazon Machine Learning . <https://aws.amazon.com/>. Accessed: 2010-09-30.
- Kevin Ashton. That ‘internet of things’ thing. *RFiD Journal*, 22(7):97–114, 2009.
- AzureML. Microsoft Azure ML. <https://studio.azureml.net/>. Accessed: 2010-09-30.
- Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.
- C. Bormann, A.P. Castellani, and Z. Shelby. Coap: An application protocol for billions of tiny internet nodes. *Internet Computing, IEEE*, 16(2):62–67, March 2012. ISSN 1089-7801. doi: 10.1109/MIC.2012.29.
- Hyunseok Chang, A. Hari, S. Mukherjee, and T.V. Lakshman. Bringing the cloud to the edge. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, pages 346–351, April 2014. doi: 10.1109/INFCOMW.2014.6849256.
- Contiki-os. Contiki-os. <http://contiki-os.org/>. Accessed: 2010-09-30.
- Romano Fantacci, Tommaso Pecorella, Roberto Viti, and Camillo Carlini. Short paper: Overcoming iot fragmentation through standard gateway architecture. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 181–182. IEEE, 2014.
- Martin Frické. The knowledge pyramid: a critique of the dikw hierarchy. *Journal of information science*, 35(2):131–142, 2009.
- Tao Fu and Mengchi Liu. A gateway from html to xml. In *Database Engineering and Applications Symposium, 2004. IDEAS'04. Proceedings. International*, pages 205–214. IEEE, 2004.

- Panagiotis Kalagiakos and Panagiotis Karampelas. Cloud computing learning. In *Application of Information and Communication Technologies (AICT), 2011 5th International Conference on*, pages 1–4. IEEE, 2011.
- Kevin Li, Charles Gibson, David Ho, Qi Zhou, Jason Kim, Omar Buhisi, Donald E Brown, and Matthew Gerber. Assessment of machine learning algorithms in cloud computing frameworks. In *Systems and Information Engineering Design Symposium (SIEDS), 2013 IEEE*, pages 98–103. IEEE, 2013.
- G. Moritz, F. Golatowski, C. Lerche, and D. Timmermann. Beyond 6lowpan: Web services in wireless sensor networks. *Industrial Informatics, IEEE Transactions on*, 9(4):1795–1805, Nov 2013. ISSN 1551-3203. doi: 10.1109/TII.2012.2198660.
- A.Y. Nikraves, S.A. Ajila, and Chung-Horng Lung. Measuring prediction sensitivity of a cloud auto-scaling system. In *Computer Software and Applications Conference Workshops (COMPSACW), 2014 IEEE 38th International*, pages 690–695, July 2014. doi: 10.1109/COMPSACW.2014.116.
- Node.js. Node.js. <https://nodejs.org/>. Accessed: 2010-09-30.
- Daniel Peintner, Harald Kosch, and Jörg Heuer. Efficient xml interchange for rich internet applications. In *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, pages 149–152. IEEE, 2009.
- Python. Python . <https://www.python.org/>. Accessed: 2010-09-30.
- Robert R Schaller. Moore’s law: past, present and future. *Spectrum, IEEE*, 34(6):52–59, 1997.
- Monika Simjanoska, Sasko Ristov, and Marjan Gusev. Machine learning approach for performance based cloud pricing model.
- Dominik Slezak and Marcin Kowalski. Intelligent granulation of machine-generated data. In *IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), 2013 Joint*, pages 68–73. IEEE, 2013.
- Moataz Soliman, Tobi Abiodun, Tarek Hamouda, Jiehan Zhou, and Chung-Horng Lung. Smart home: Integrating internet of things with web services and cloud computing. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, volume 2, pages 317–320. IEEE, 2013.
- JI Walker, DL Blodgett, I Suftin, and T Kunicki. Progress on big data publication and documentation for machine-to-machine discovery, access, and processing. In *AGU Fall Meeting Abstracts*, volume 1, page 1522, 2013.
- Chip Walter. Kryder’s law. *Scientific American*, 293(2):32–33, 2005.
- P. Wehner, C. Piberger, and D. Göhringer. Using json to manage communication between services in the internet of things. In *Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2014 9th International Symposium on*, pages 1–4, May 2014. doi: 10.1109/ReCoSoC.2014.6861361.
- Tim Winter. Rpl: Ipv6 routing protocol for low-power and lossy networks. 2012.
- Jiehan Zhou, T Leppanen, Erkki Harjula, Mika Ylianttila, Timo Ojala, Chen Yu, Hai Jin, and Laurence Tianruo Yang. Cloudthings: A common architecture for integrating the internet of things with cloud computing. In *Computer Supported Cooperative Work in Design (CSCWD), 2013 IEEE 17th International Conference on*, pages 651–657. IEEE, 2013.

## Appendix A

### Code Listing

```
1 import urllib2
2 import json
3
4 data = \{"Inputs": \{"input1": \{ "ColumnNames": ["windspeed", "turbine speed", " ←
      vibration", "energy output", "efficiency", "status"],"Values": [ [ "0", "0", "0" ←
      , "0", "0", "0" ], [ "0", "0", "0", "0", "0", "0" ], ] \}, \}, "GlobalParameters ←
      ": \{\}\} # Sample of the Json string format. Replace with actual data
5
6 body = str.encode(json.dumps(data))
7
8 url = 'xyz123' # Replace this with the URL of the web service
9 api_key = 'abc123' # Replace this with the API key for the web service
10 headers = {'Content-Type':'application/json', 'Authorization':('Bearer '+ api_key)}
11
12 req = urllib2.Request(url, body, headers)
13
14 try:
15     response = urllib2.urlopen(req)
16     result = response.read()
17     print(result)
18 except urllib2.HTTPError, error:
19     print("The request failed with status code: " + str(error.code))
20     print(error.info())
21     print(json.loads(error.read()))
```

Listing A.1: Python code to utilise ML Web Service

```

1 var express = require('express'),
2   http = require('http'),
3   url = require('url'),
4   app = express();
5 app.get('/:host*', function (request, response, next) {
6   var proxyurl = url.parse(request.url);
7   var path = request.params[0];
8   if (!!proxyurl.search) {
9     path += proxyurl.search;
10  }
11  http.get({
12    host: request.params.host,
13    path: path,
14    headers: {}
15  }, function(res) {
16    var body = '';
17
18    res.on('data', function(chunk) {
19      body += chunk;
20    });
21    res.on('end', function() {
22      response.end(body);
23    });
24  }).on('error', function(e) {
25    console.log("Got error: ", e);
26    response.end("<NodeID>");
27  });
28 });
29 app.listen(8080, ':::', function() {
30   console.log('Node server listening on localhost port 8080');
31 });

```

Listing A.2: Javascript NodeJs Reverse Proxy Server