An Open Vendor Agnostic Fog Computing Framework for Mission-Critical and Data Dense Applications

Presented by:
Tasimba Denford David Chirindo

Prepared for:
Dr. Joyce Mwangama
Department of Electrical Engineering
University of Cape Town

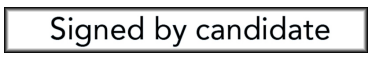
Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Master of Science in Engineering degree in Electrical Engineering.

February 11, 2019

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.
5. I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own. This thesis/dissertation has been submitted to the Turnitin module (or equivalent similarity and originality checking software) and I confirm that my supervisor has seen my report and any concerns revealed by such have been resolved with my supervisor.

Signature:



Tasimba Denford David Chirindo

Date: February 11, 2019

Acknowledgements

I would like to thank the following individuals for their help during the course of this project.

My dear family; my parents: Denford Chirindo and Placcidias Chirindo. And, my dear sisters: Kudzanayi, Tsungai, Thandiwe and Rutendo Chirindo. Thank you for always believing in me and for your encouragement.

Dr. Joyce Mwangama, for her supervision and guidance throughout the duration of this project. I am so lucky to have had such an insightful learning experience under your guidance. I am fortunate to have had your mentorship as well.

The architects of OpenBaton from Fraunhofer FOKUS and to all contributors to open source initiatives used throughout the project. Without the dedication and support of these individuals, the open nature of this work would not have been possible.

The directors at my present employer, OPSI Systems, Mr David Lubinsky, Mr David Clark, Mr Tim Mitchell, Mr Paul Baker, Mr Sean Aspoas and Mr Sam Nudelman; I thank you.

Thank you to all of the past and present members of the Communications Research Group.

Rodwell Mangisi, thank you my mentor. Pamela, Admire and Pardington thank you best mates.

Abstract

Digital innovation from the Internet of Things (IoT), Artificial Intelligence, Tactile Internet and Industry 4.0 applications is transforming the way we work, commute, shop and play. Current deployment strategies of these applications emphasize mandatory cloud connectivity. However, this is not feasible in many real-world situations particularly where data dense and mission critical applications with stringent requirements are concerned.

Cloud computing offers unlimited on-demand computing, storage and networking power for industry to leverage. However, as its scope and scale continues to expand, its limitations like high latency, accessibility, security and compliance shortcomings prevent its greater use and applicability particularly in scenarios where real-time communication and the quality of rapid computing delivered is a necessity. Fog computing hopes to bridge this gap by introducing an intermediary computing layer between end users and the cloud.

At present, architectures for fog computing exist in specialized areas with current implementations being proprietary, vendor-locked and requiring dramatic and non-transferable changes to hardware and software to meet vendor requirements. Moreover, fog computing is still quite a recent area which makes the state of the art incipient regarding architecture definitions, middleware and real-world implementations. There is therefore an urgent need for standardization of these technologies. This is of paramount importance as otherwise, there will exist multiple and not necessarily compatible solutions which could lead to a fragmented marketplace that would fail to grow.

In an effort to address these limitations in current fog architectures, this dissertation proposes and implements a novel fog computing architecture that aligns the reference architectures from a leading industry consortium, OpenFog, and a leading standards setting organization, the European Telecommunications Standards Institute (ETSI). This cooperation framework from industry, academia and regulatory institute aims to make it easier for both application developers and infrastructure solution providers to develop towards a common, open and interoperable fog computing environment. The

proposed framework has the following attributes: modular, plug-in design, generic, open, standards compliant, vendor agnostic and runs on high volume standard hardware whilst preserving the benefits offered by public clouds such as containerization, virtualization, orchestration, manageability and efficiency. Moreover, for the various stakeholders in the fog value chain where it is key to strike a balance between information technology and business operations, this thesis tenders insights and best practices to help achieve these multiple and sometimes competing goals.

The proposed framework was implemented in a testbed environment made up entirely of free and open source software, therefore creating a convenient point of departure for further research by others. Two geographically distributed fog node data centres and a cloud management and orchestration tool were setup in the testbed. While this evaluation framework and practical implementation demonstrated proof of concept, further evaluations were conducted to benchmark the performance against existing alternative solutions. These evaluations were based on a prototype industrial IoT application that was deployed on the testbed to evaluate the impact of the Open Vendor Agnostic Fog Framework (OVAFF) solution on application performance.

The implementation showed that the proposed OVAFF solution is feasible, implementable and supports distributed edge cloud data centres. Results from the prototype application showed that OVAFF can extremely provide up to tenfold throughput and ultra-low latency, jitter and packet loss rate better than the remote clouds. Moreover, there is more superiority exhibited by the OVAFF for other non-performance based attributes like data reduction, compliance and geographical locality of control. In addition, the results also pointed towards the viability of open business models like federated infrastructure sharing and a fog market place in the fog ecosystem. Finally, this thesis tackled the highlighted open challenges in current fog systems such as orchestration, distribution, tiering, heterogeneity and resilience; which were outlined in the research motivation and problem definition.

Contents

List of Figures

List of Tables

List of Acronyms

3GPP	3rd Generation Partnership Project.
5G	5th-Generation Wireless Systems.
AI	Artificial Intelligence.
AMQP	Advanced Message Queuing Protocol.
APIs	Application Programming Interfaces.
AR	Augmented Reality.
AWS	Amazon Web Services.
C8	Containers.
CDN	Content Delivery Network.
CMTS	Cable Modem Termination System.
COTS	Commercial Off-The-Shelf.
CPU	Central Processing Units.
CRG	Communications Research Group.
CRM	Customer Relationship Management.
CSPs	Communications Service Providers.
DNS	Domain Name System.
DPDK	Data Plane Development Kit.
DSL	Digital Subscriber Line.
EMS	Element Management System.
ERP	Enterprise Resource Planning.
ETSI	European Telecommunications Standards Institute.
FOA	Fog Orchestration Agent.
FOSS	Free and Open Source Software.
I/O	Input/Output.
IaaS	Infrastructure-as-a-Service.
ICT	Information and Communications Technologies.

IoT	Internet of Things.
ISG	Industry Specification Group.
IT	Information Technology.
KPIs	Key Performance Indicators.
KVM	Kernel-based Virtual Machine.
LTE	Long-Term Evolution.
M2M	Machine to Machine.
MANO	Management and Orchestration.
MCC	Mobile Cloud Computing.
MEC	Multi-Access Edge Computing.
MoU	Memorandum of Understanding.
ms	milliseconds.
NFV	Network Function Virtualization.
NFVI	Network Function Virtualization Infrastructure.
NS	Network Service.
NSD	Network Service Descriptor.
NSR	Network Service Record.
NTP	Network Time Protocol.
OLT	Optical Line Terminal.
OS	Operating System.
OSS	Operating Suppport System.
OTT	Over The Top.
OVAFF	Open Vendor Agnostic Fog Framework.
POC	Proof of Concept.
POCs	Proof of Concepts.
PON	Passive Optical Network.
PoP	Point of Presence.
QoE	Quality of Experience.
QoS	Quality of Service.
RAN	Radio Access Network.
REST	Representational State Transfer.
SDKs	Software Development Kits.
SDN	Software Defined Networking.
SLA	Service Level Agreement.
SR-IOV	Single-root Input/Output Virtualization.
SUT	System Under Test.

TCP	Transmission Control Protocol.
UCT	University of Cape Town.
UDP	User Datagram Protocol.
UE	User Equipment.
VIM	Virtualization Infrastructure Manager.
VM	Virtual Machine.
VNF	Virtual Network Functions.
VNFC	Virtual Network Function Container.
VNFD	Virtual Network Function Descriptor.
VNFM	Virtual Network Function Manager.
VNFR	Virtual Network Function Record.
vSwitch	Virtual Switch.

Chapter 1

Introduction

The rise of 5th-Generation Wireless Systems (5G), the Internet of Things (IoT) and more importantly, the new applications often associated with them, will drive digital innovation that will transform the way we work, commute, shop and play [1]. This innovation trigger has caught the attention of Communications Service Providers (CSPs) worldwide and it has become clear to industry that edge computing architectures will play an important role in ensuring the successful roll-out of these new applications beyond 2020 [2]. Edge computing brings virtualization, self-service Application Programming Interfaces (APIs) and on-demand provisioning of compute, storage and networking infrastructure closer to the user and the access network.

Increased network speeds and data feeds will characterize the 5G arena. 5G technology is expected to deliver at the following specifications: up to 10+ Gb/s peak data rates, 100Mb/s sustained rates, 10 to 100 times (10-100x) more devices supported, radio latency and reliability of less that 1ms. Combination of these network improvements with network slicing will create a whole range of new services that will connect machines, users, enterprises and governments [2]. Data from these newly connected services is expected to grow from 1.1 zettabytes per year in 2016 to 2.3 zettabytes per year by 2020 [1]. Cloud-only computation approaches currently prevalent on the market cannot keep up with this envisioned high volume and velocity of data across the network, thereby creating a bottleneck to the value that can be created and captured from these investments.

In recent years, the Information Technology (IT) industry has been stormed with digital transformation of on-premise data centre infrastructure migrating to the cloud. One of the biggest benefit offered by the cloud is that industry can now avoid upfront

costs and complexity of owning and maintaining their own IT infrastructure by simply paying for what they use, when they use it.

Cloud computing is built on the pillar of utility computing where a collection of resources (compute, storage, networking) are pooled to serve multiple customers using a multi-tenant model. Public cloud vendors have built a few large scale data centres around the world which can serve a very large number of users. However, this centralization of resources results in a large separation between end users and their clouds which causes large and or unpredictable end-to-end (E2E) network delays. Moreover, in the context of connected things, sensing and actuating devices generate tremendous amount of data and may need to be managed in real-time. Sending all this data to a distant cloud is often prohibited due to regulations and data privacy concerns. Consequently, this also requires prohibitively high network bandwidth which is expensive and fickle in mobile networks. These amongst other challenges motivate the need for edge computing paradigms that favour 'more decentralization' and 'more distribution' of cloud computing resources since evidently, unfettered cloud-only approaches cannot sustain the projected real-time and high data requirements of the Internet of Things (IoT) in Industry 4.0 [3].

The notion of computing resources being close to users is not a new idea, but the application of cloud computing principles is. In recent years, several related architectures in the edge computing space namely Fog Computing [4], Multi-Access Edge Computing (MEC) [5] and Mobile Cloud Computing (MCC) [5] have emerged. The common attribute in these paradigms is the deployment of cloud computing-like principles at the edge of the network.

Fog computing provides the missing link in the cloud-to-thing continuum by distributing virtual computing resources closer to users along the hierarchy. In fog computing, the edge refers to nano-data centres called Fog Node Hosts that are placed on one or multiple access points which may be the Radio Access Network (RAN) for Long-Term Evolution (LTE)/5G, Passive Optical Network (PON) Optical Line Terminal (OLT) for fibre, radio network controller for WiFi, Cable Modem Termination System (CMTS) for cable or an appropriate access point for other networks such as Zigbee, Digital Subscriber Line (DSL) and private LTE [2]. The main differentiator between fog computing from MCC and MEC is that fog computing is favoured by cloud service providers, enterprises and data processing companies whereas MEC is favoured by the telecos and middleware companies that actually own the back-bone and or radio networks. The market and use cases of fog computing are thus broader and the technology is transferable across other vertical markets in the enterprise.

1.1 Definition and Characteristics of Fog Computing

Foundational work for distributed intelligence in operations networks has a rich history from well before cloud, IoT or fog computing [6]. Examples of this history include distributed intelligent systems in the space program, voice telephone networks, fly-by-wire systems, and industrial programmable logic controllers for machine automation [6]. More recently, the multi-core era has effected a variety of widely adopted computing paradigms such as full blown super computers, grid computers, cluster computers, accelerator based computing and cloud computing [7]. Figure 1.1 shows the timeline view of the evolution of distributed computing systems [8]. Despite this growth, there continues to be a significant need for more computational capabilities to meet future challenges [7].

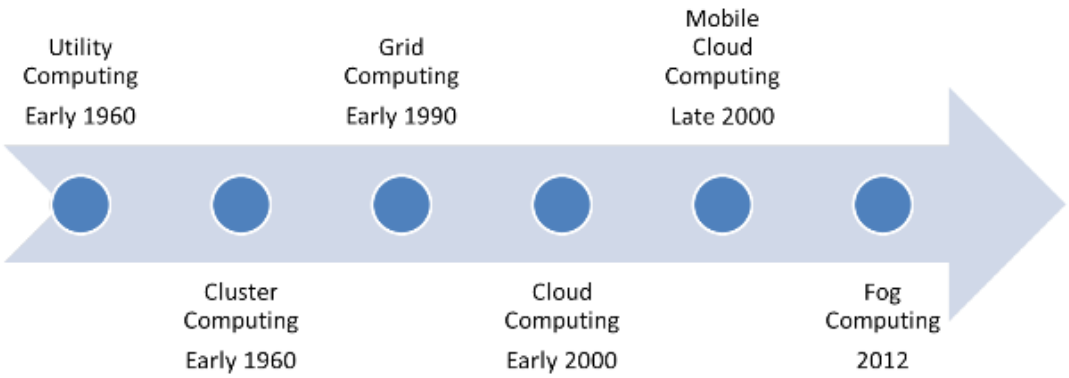


Figure 1.1: Timeline view of the evolution of distributed computing systems [8].

It is forecast that between 20 to 50 billion previously unconnected devices will be added to the internet by 2020, thereby creating an economy of over $3 trillion [1] [6] [8]. Consequently, data from these newly connected services is expected to grow from 1.1 zettabytes per year in 2016 to 2.3 zettabytes per year by 2020 [1] and with prevalent computation methodologies, will need to be processed by cloud data centres.

Currently, most industries heavily rely on the cloud only model but this will soon become an untenable computing model [9]. This is simply because the frequency and latency of communication between end systems and geographically distant cloud data centres will increase beyond that which can be handled by current communication and computing infrastructure [6] [8]. To effectively manage this data explosion, applications will need to process data closer to its source. However, this may not be possible on end devices and systems since they have relatively restricted hardware resources. Hence, there is a strong motivation to look beyond the cloud towards the edge of the network to harness computational capabilities that are currently untapped [6].

The concept of distributed computing on the edge of the network in conjunction with the cloud is referred to as fog computing. This computing model is based on the premise that computational workloads can be executed on fog nodes situated in-between the cloud and a host of user devices to reduce communication latencies and offer better Quality of Service (QoS) and Quality of Experience (QoE) [1] [6].

1.1.1 Definition of Fog Computing

Fog computing is an architectural construct built on the foundations of well-known technologies including cloud computing, distributed control systems, industrial networks and wireless infrastructure [6]. Although fog computing is a relatively new paradigm as illustrated in the timeline in Figure 1.1, it can also be interpreted as the convergence of technologies that have been around for a while developing and maturing, most of the time independently of each other [10].

Fog computing is first of all an architectural approach, secondly a software and thirdly a hardware component. The term fog computing was created by Cisco Systems in 2012 [11] and in its initial definition, fog computing was considered "an extension of cloud computing capabilities to the edge of an enterprise's network" [11].

Based on the numerous advances that can be introduced by the fog computing model, the initial definition was latter revised and expanded [12]. Under the new definition, fog computing is not a mere extension of cloud computing, but a paradigm of its own. The OpenFog Consortium [13] defines fog computing as "a horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum" [1].

From these definitions, fog computing therefore does not cannibalize traditional cloud-based computing models but compliments them with architectural implementations that reside in multiple layers of the network's topology. Despite its distributed nature, fog computing preserves the benefits offered by the cloud such as containerization, virtualization, orchestration, manageability and efficiency [1].

Fog computing therefore exists to provide a low-latency, cost effective, secure, reliable and high-bandwidth collaborative cloud environment for applications, services and content to be placed in close proximity to the network and end users. Centralized cloud servers coexist with fog nodes but are not essential for the execution of fog services.

The idea is that processing is not done in one location, but in a tiered continuum of locations as shown in Figure 1.2. It can be noted that fog computing is not an offloading solution; rather it is a continuum that ranges from the cloud to the ground level [8]. This way, the nature of processing changes along the hierarchy; fog nodes closer to the devices or process being monitored will spend more time on data collection, filtering and processing; whereas nodes farther up perform data aggregation, analytics and intelligence creation [2].

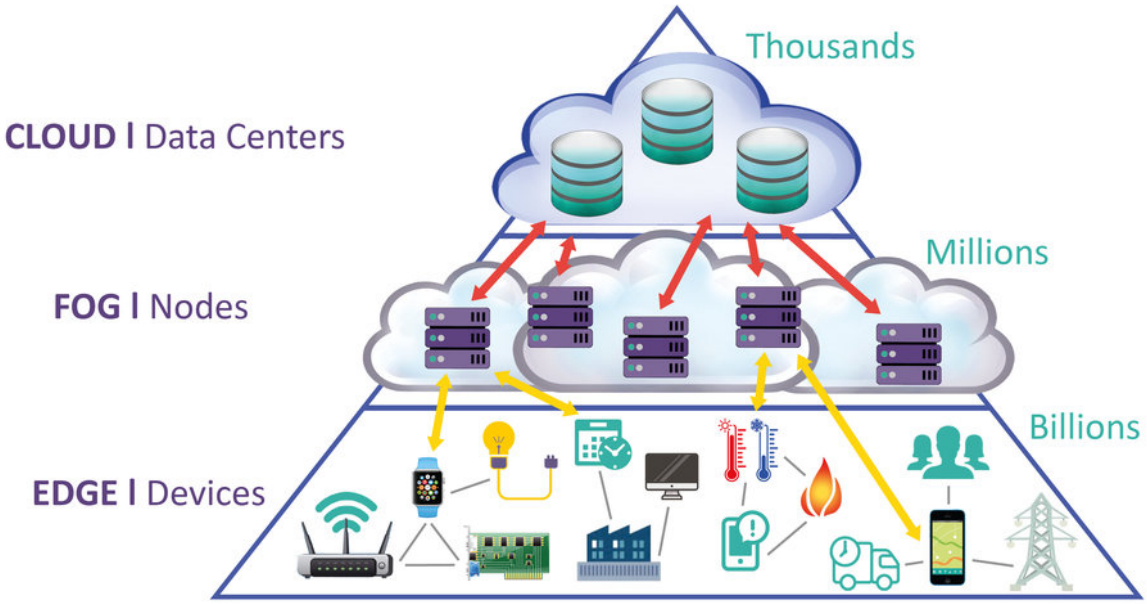


Figure 1.2: Representation of the fog architecture [14].

Fog computing is often erroneously ascribed as edge computing; but there are key differences. Fog works with the cloud as shown in Figure 1.2 whereas edge is defined by the exclusion of the cloud [1]. Moreover, fog computing is hierarchical whereas edge tends to be limited to a small number of layers. Finally, fog computing was originally defined as a platform for enhancing applications and services in the context of the Internet of Things (IoT). Yet, at present, several studies and implementations have proven that this paradigm can be applied in a broad range of other use-cases outside the scope of IoT [11] [12].

1.1.2 Use Cases of Fog Computing Across Different Vertical Markets

Fog computing technology is applicable to a wide variety of potential industry use cases that serve many key vertical markets. Table 1.1 lists some examples of the vertical markets and several selected use cases of fog computing technology in each market [6] [15].

Table 1.1: Examples of fog use cases across different industry vertical markets. [6]

Industry vertical	**Example fog-enabled use cases**
Transportation	Smart roads, autonomous vehicles, rail, unmanned aerial vehicle ground support, maritime, ports, logistics.
Utilities	Smart grid, smart meters, water distribution, sewer monitoring, energy management, renewables.
Smart cities/Smart buildings	City-level fog, smart buildings, lighting, emergency services, sanitation, carpeted spaces.
Manufacturing	Plant automation, robotics, analytics, smart supply chain, quality control, distribution, logistics.
Retail/Enterprise	Smart store, branch-in-a-box, security, asset tracking, digital signage, analytics, thin clients.
Service providers	Smart networks, fog-as-a-service, media caching, microcells, resiliency.
Oil/Gas/Mining	Exploration, rig-in-a-box, production monitoring, pipeline control, refinery control.
Health care	Continuous patient monitoring, cognitive assistance, exercise.
Agriculture	Irrigation, crop monitoring, yield assessment, pest control, autonomous equipment.
Government/Military	Homeland security, autonomous vehicles, electronic warfare, connected fighter.
Hospitality	Front desk, bell robots, entertainment, security, cruise ships, camp grounds, dormitories.

This list of markets and use cases is by no means exhaustive. Rather, it is more illustrative of the sorts of problems that can be addressed with the fog model. Many of these use cases are impossible to satisfactorily implement cloud-only and will experience significant challenges without the use of fog techniques

1.2 Research Motivation

Today's data is extremely dispersed and is often delivered continuously in large volumes to and from a large number of heterogeneous connected systems. Contemporaneously, the fog model offers data processing, computing power and storage at a really distributed level using a tiered hierarchy of processing along a continuum of locations from the edge to the centralized cloud.

Although this approach has many well-documented benefits and the field is rapidly maturing, there are still a number of open challenges that must be solved before adoption becomes more widespread. Some of these challenges are inherent in the concept of fog computing itself, whereas some are introduced by existing implementations and early Proof of Concepts (POCs) whilst others are as a result of lack of standardization, openness and cooperation between vendors.

Fog computing is still quite a recent area which makes the state of the art incipient regarding architecture definitions, middleware and real world implementations [16]. Benefits of fog computing are currently demonstrated on research use-cases which dominantly focus on the computing enhancements brought about by the fog ecosystem. This concentration greatly limits the extensibility of these frameworks to the enterprise where integration of fog computing with existing business support systems is critical for the control, operations and corporate governance requirements of systems. Moreover, there are few commercial fog computing services that integrate the edge and cloud models. Current widely adopted solutions in the market like Amazon Web Services (AWS) Greengrass [17], Azure Stack [18] and IBM's Watson IoT Platform [19] are proprietary, vendor locked, run on customized hardware, costly to on board and are tightly coupled to the public clouds of their vendors.

In addition, orchestration in the fog computing space is still an open outstanding area of concern. Orchestration techniques are leveraged to enhance system reliability and alleviate the difficulties in maintenance of fog systems. However, open techniques on how to efficiently deal with dynamic variations and transient operational behaviour are still pending within the context of choreographing the complex services in fog computing. This brings about a need for open compossible and concrete orchestration methodologies for supporting the development of distributed novel IoT applications.

The tiered and distributed nature of the fog model contributes significantly to

fog system resilience since distribution means that there is no single point of failure. Generally, distributed systems are known to scale better than centralized systems which may experience performance bottlenecks. Managing diverse applications running on potentially heterogeneous infrastructure which may potentially be owned by third parties is still another open challenge to be addressed in the fog ecosystem. This challenge requires new techniques for packaging and on boarding applications across the distribution as well as discoverability and inter-node communication capabilities. These requirements have not yet being addressed in a single architecture in existing works.

Moreover, resilience on its own is a challenging area of concern. While distribution is an enabler of resilience, it significantly complicates management and configuration of fog applications across different hosts and networks. Currently, many IoT devices and fog applications must be individually configured, requiring fog application providers to use many different vendor specific and low-level languages. In some instances, moving from one vendor to the other often requires a rewrite of applications. This makes IoT device configuration and network management complex and error-prone. As the deployments scale, the administration becomes even more complex. The end result is usually a system where innovation is slow, management and operational costs are high, and errors are difficult to trace and fix. Despite the challenges of resilient systems, fog computing is expected to preserve the agility and flexibility offered by the cloud.

Also, 5G networks are earmarked to be run as software appliances on top of high volume standard hardware. Since the Radio Access Network (RAN) is a potential physical edge location for fog deployments, there are concerns in industry around the Network Function Virtualization (NFV) alignment of fog computing architectures. Here, fog applications and Virtual Network Functions of RAN services may need to run at the same location. Operators and fog providers will have to perform gap analysis and either share the same infrastructure, share similar infrastructure hence simplifying procurement or use different infrastructures if the requirements diverge significantly. Aligning fog computing to NFV is an avenue that has been rhetorically discussed, but not physically implemented or quantitatively defined.

Finally, the last challenge in enabling the vision of fog computing is the establishment of innovative business models by fog providers. Of these potentially explosive candidate business models is a fog computing market place that will make fog nodes visible and publicly accessible in a commercial fog computing model. Without concrete reference architectures that integrate multi-vendor solutions and define open communication APIs and Software Development Kits (SDKs), this bottleneck will continue to throttle efforts of

establishing a solid foundation upon which discussions around establishing a fog market place and federated infrastructure sharing agreements can commence between fog node owners and cloud service providers.

1.2.1 Research Questions

The main research questions investigated can be summarized as follows:

i. What synergies can be drawn from the main edge computing reference architectures, ETSI ISG and the OpenFog Consortium, to create a robust fog computing framework that fills the gaps in current fog computing architectures?

ii. Can alignment of fog computing with NFV be achieved with open standards in a vendor agnostic approach using existing standards?

iii. What are the architectural requirements and considerations necessary for establishing a generic fog framework that runs on high volume standard hardware?

iv. What elements of the envisioned fog computing architecture are crucial to enable hierarchical placement of applications and services on heterogeneous infrastructure and across multiple service providers?

v. What strategies and techniques are required to effectively manage and orchestrate geographically distributed applications in the heterogeneous multi-stakeholder fog ecosystem?

vi. What considerations will be taken into account in determining the dimensionality of each fog node data centre based on its envisioned function and multiple scalability axes such as performance, capacity and number of end-users or devices?

vii. What are the critical attributes missing in current fog frameworks that will enable innovative business models like federated infrastructure sharing and a fog market place for the various stakeholders in the fog value-chain?

viii. What high level partitioning strategies are key in determining what goes where in the application placement hierarchy across the varied use-cases in different vertical fog market segments?

1.3 Dissertation Objectives

Success in actualizing fog computing in the enterprise will be benchmarked on the broad market applicability and industry relevance of the OVAFF technology across different vertical markets. Scale, cost and agility of fog deployments form a subset of some of the important metrics that will be used to quantify this success.

The objectives of this dissertation can be summarized as follows:

i. The first objective of this dissertation is to present available literature on fog computing and to highlight the need, use cases and standardization efforts around the technology by expanding on the relevant aspects of each topic.

ii. The second objective of this dissertation is to present and analyse related works on other proposed architectures, as well indicating their scope and limitations.

iii. The primary objective of this dissertation is to design and implement an ideal Open Vendor Agnostic Fog Framework (OVAFF) that industry can leverage on in the enterprise.

iv. Finally, the feasibility, capabilities and benefits of the proposed framework are articulated clearly by means of Key Performance Indicators (KPIs), benchmarking tests and a proof of concept application use case from industry.

1.3.1 Contributions

The contributions of this work are the following: This work proposes a generic Open Vendor Agnostic Fog Framework (OVAFF) that demonstrates how ideas of fog computing can be realized in the enterprise. The OVAFF is designed off a cooperation framework between OpenFog and ETSI, thereby representing a significant step towards adoption of standards by the industry. Aligning efforts of a leading industry consortium and a leading standards setting organization should push for more fog computing technology to be deployed with scale and agility hence actualizing broad market applicability and industry relevance of this technology.

To achieve these goals, the OVAFF is designed as an end-to-end fog computing architecture off the synergies drawn from the reference architectures from the ETSI

Industry Specification Group (ISG) and OpenFog Consortium, the main industry organizations driving standardization of edge computing. These reference architectures compliment each other thus filling-in the gaps in pre-existing fog computing architectures. The OVAFF is designed to be generic, open, vendor agnostic, plug-in, modular, ETSI NFV [20] and OpenFog [1] compliant, runs on high volume standard hardware and is integrable with multiple public clouds. Over and above that, the framework preserves the benefits of the cloud such as containerization, virtualization, orchestration, manageability and efficiency.

To the best of my knowledge, this is the first work to realize a multi-tiered fog computing reference architecture provided by the OpenFog Consortium [1] using open NFV principles. This work thus aligns fog computing with NFV, making the architecture robust, future proof and 5G ready. Furthermore, it is the first work to realize an open enterprise ready fog framework that is conformant to NFV standards and runs on high volume standard hardware. A standardized OVAFF provides a firm foundation for exploration of business models around a fog market place and federated infrastructure sharing in the fog ecosystem.

Finally, the OVAFF is designed to support high level application development and to seamlessly support non-telco applications, thus addressing the configuration and automation concerns present in current architectures.

1.4 Scope and Limitations

The scope of this dissertation is limited to architecture definition and implementation of an OVAFF. The OVAFF is intended to be a tiered and standards compliant fog framework that spans across heterogeneous hardware and infrastructure service providers. Algorithmic aspects are however not considered in this dissertation despite the critical role algorithms play in architecture definitions. In future extensions of the OVAFF, architectural and algorithmic perspectives will be comprehensively considered.

The scope is further limited to designing the OVAFF off pre-existing open standards and APIs. This work does not propose any extensions to these current standards since they are constantly added to and updated at a faster pace.

In addition, the scope excludes deeper delve into areas like the security, confidentiality

and data protection attributes of the OVAFF. The importance of these aspects are however acknowledged. These issues have been excluded because their scope is very broad and they are therefore better exhaustively covered in dedicated surveys.

Finally, while this work is intended to be operable in the 5G space, the test bed implementation will be limited to the deployment of fog applications into a nano data centre and not 5G RAN infrastructure. At the time of writing, no 5G technology currently exists ad it is still being defined.

1.5 Dissertation Outline

The remainder of this document is structured as follows:

Chapter 2 presents the background to the study and literature review. Firstly, architectural imperatives of fog computing and its contributing key enablers are presented. Then, brief overview on the standardization efforts around fog computing then follows before the main edge computing reference architectures are presented. Finally, research activities of existing works and their contributions are discussed.

Chapter 3 describes the design and specification requirements of the OVAFF. This chapter presents the proposed architecture of the OVAFF.

Chapter 4 outlines the full test bed implementation of the proposed OVAFF. The testbed is described and implemented to showcase proof of concept and a practical evaluation framework. Then, a prototype industrial IoT application to be used for evaluation is presented. Finally, the validation and performance testing tools are also described.

In Chapter 5, the results from the testbed setup and an evaluation of the solution is presented. An analysis of the KPI performance of the prototype application use-case is presented based on a comparison to its equivalent performance on existing solutions.

Chapter 6 presents the conclusions drawn from the dissertation and summarizes the contribution. Future work is then discussed before recommendations are made for areas of further study.

Chapter 2

Background to Study and Literature Review

The previous chapter introduced the basic concepts, aims and challenges of fog computing to set the tone for the paper. This chapter reviews the motivating factors in the advancement of open fog computing technologies before presenting the collaborative industry initiatives towards the sustainable development of this industry. Then, the main reference architectures for edge computing paradigms are presented. Finally, related works from other research activities and existing works are presented.

2.1 Architectural Imperatives for Fog Computing

Certain clusters of requirements are difficult to implement on networks built with heavy reliance on the cloud-only or intelligent-things-things only paradigms. These requirements are particularly influential in the decisions of migrating to fog-based architectures.

To optimize the architectures of fog networks, it is essential to first understand and identify the critical requirements of the general use cases that will take advantage of the fog model. This dissertation refers to these decision criteria as architectural imperatives for fog computing. This section presents six unique benefits offered by the fog model that a centralized cloud data centre cannot offer.

2.1.1 Enhanced Security and Privacy

Previously, data breaches and hacking episodes were costly but physical harm as a result was rare because these systems could not directly and immediately impact the physical world [21]. Nowadays, IoT systems are fully connected to the public internet and cloud. Their actuators directly control multi-megawatt systems like locomotives, potentially dangerous systems like refineries or medical implants, and privacy-critical systems like surveillance drones [6]. If the security of these networks is compromised, there is a real danger that people will die.

The locality of fog nodes can reduce the attack surfaces and distances over which data must flow by moving the security perimeter closer to the source [21] [22]. Moreover, fog nodes can be mechanically more resistant to damage, have unauthorized tampering or outright theft than highly constrained IoT devices, making them more trustworthy [6].

2.1.2 Low Latency

In the fog context, latency measures the absolute delay control information takes in a real-time system [6]. This delay may be one way but will more typically be round-trip. If latency exceeds a certain application-specific threshold, the process can go unstable, results arrive too late and control is lost.

There are an increasing number of applications that are real time in nature and cannot tolerate latency more than in the order of tens (10's) of milliseconds (ms) [2]. Additionally, these applications are also sensitive to jitter, the variation in latency. A few examples of these applications are connected cars, tactile internet, Industry 4.0 and smart cities. Table 2.1 includes some example fog use cases, their typical latency targets and whether fog techniques are likely to be needed.

Once the latency requirements fall below a few tens of milliseconds, fog techniques become essential. The fog can provide latency in milliseconds, while the multiple hops and long data transmission distances mean that the latency to nearest CDNs is in-between the 50ms to 150ms range [6]. Latency to centralized data centres and the public cloud is even greater, compounded by any queueing delays in the network or servers.

Table 2.1: Application examples and allowable latency [2].

Application examples	**Latency**	**Implementation**
Big data file download, offline backup	100 s	Easy with cloud
YouTube, home automation, video surveillance	10 s	Easy with cloud
Web search, sensor readings	1 s	Challenging with cloud
Interactive web site, smart building, analytics	100 ms	Challenging with cloud
Virtual reality, smart transportation, games, finance	10 ms	Impossible with cloud, needs fog
Haptics, robotics, real-time manufacturing	1 ms	Impossible with cloud, needs fog

2.1.3 High Throughput

The tremendous bandwidth available to users from the fog, served via cached or locally generated content, can be orders of magnitude greater than from a core data centre or even the CDN [6]. This optimization in throughput is a critical factor since protocols such as Transmission Control Protocol (TCP) do not react well to sudden changes in the dynamic fog environment which can swing by up to an order of magnitude. Other network level metrics, such as round-trip time and packet loss can improve by 30 to 60 percent [2].

2.1.4 Data Reduction

Network bandwidth between things and the cloud is often a serious constraint in data networks [8]. Sometimes data sets are so large that the transmission delay to move data to the cloud, even over fast access facilities is unacceptable [6]. Sometimes data streams are too high in average bandwidth to be affordably carried on available data transmission facilities [6] [8]. And sometimes the data sources are so remote that no terrestrial network connection is available [6].

Fog application providers can substantially cut down the amount of data that has to be sent upstream by running applications such as data analytics at the edge [2]. This cuts costs and allows for other applications to transfer data.

2.1.5 Environmental Constraints

In an age when there is universal connectivity, it seems strange that there are actually environments with unreliable or no connectivity [2]. These could include cruise ships, planes, mines, farms, oil rigs, trains, pipelines, solar power plants or electric grids [6]. Some of these environments are not always connected to the Internet over high speed links. Fog computing is critical to roll out new services to these environments.

2.1.6 Compliance and Geographic Locality of Control

In today's data driven environment, information is most useful near its point of creation or use and of limited use at longer distances [22]. For various political, regulatory, and policy reasons, it is sometimes required to maintain control of information so that it does not cross some physical or logical boundary such as staying within a country, corporate campus, military base or a private network [6].

Fog applications can help with privacy or data location laws. Fog computing offers the ability to control the flow of information based on physical or logical boundaries, and limit the flow based on rules and policies [6]. If a network of local fog nodes implements all the computation and storage capabilities of a cloud-based service, there is little reason to send the data beyond its boundaries.

2.2 Key Enablers for Federated Fog Computing

In recent years, the IT and telecommunications industries have implemented several key technological advancements that aid the establishment of a universal fog computing ecosystem [23]. These key enabling technologies are presented in this section and their relevance explained.

2.2.1 Enabling the Fog Application and Service Ecosystem

If the fog computing industry is to flourish and prosper, it is essential that software and application vendors are able to develop and bring to market innovative and

ground-breaking services and applications which can utilize fog computing functions and capabilities [23].

There is therefore a need to encourage and expedite the development of new cutting-edge applications and adapting existing services and applications to the new fog computing environment [23]. The use of open standards and APIs, familiar programming models, as well as a relevant tool chain along with SDKs are key pillars for achieving this.

Finally, once the necessary standards have been created and become mainstream, industry can benefit from the establishment of a program aimed at ensuring that fog computing applications not only comply with these standards, but are also capable of being adapted for use on platforms from different vendors [23]. To this effect, several industry entities have joined together to push forward for the sustainable development of the edge computing industry. The need for and benefits of having such a consortia are discussed in-depth in Section 2.3.

2.2.2 High-volume Standard Servers

The use of industry standard high volume servers is a key element in the economic case of fog computing. Industry standard high volume servers are built using mainstream and standardized IT components (for example x86 architecture) which can be provided in a competitive manner as key elements in the economy of scale of fog computing [23] [24].

A common feature of these standard high volume servers is that there is competitive supply of the subcomponents which are interchangeable inside the server [24]. This enables rapid and cost-effective upgrades and maintenance.

In recent years, general-purpose IT platforms are becoming increasingly adept at handling applications and services that consume vast amounts of hardware resources. Ethernet controllers supporting 10 Gbps and even 40 Gbps are mainstream today, while the use of optimized drivers enables this high throughput, even in virtualized environments with general-purpose Central Processing Units (CPU).

2.2.3 Network Function Virtualization

In the telecommunications industry landscape, current networks are populated with a large and increasing variety of proprietary hardware appliances [24]. Launching a new network service often requires adding yet another variety. Finding space to accommodate these boxes is becoming increasingly difficult compounded by the increasing costs of energy, capital investment challenges and the rarity of skills necessary to design, integrate and operate increasingly complex hardware-based appliances [25]. Moreover, hardware-based appliances rapidly reach end of life, requiring much of the procure design-integrate-deploy cycle to be repeated with little or no revenue benefit[24]. Worse, hardware life-cycles are becoming shorter as technology and services innovation accelerates, inhibiting the roll out of new revenue earning network services and constraining innovation [25].

Network Function Virtualization (NFV) transforms the way that telecommunications operators architect networks by evolving standard IT virtualisation technology to consolidate many network equipment types onto industry standard high volume servers, switches and storage which could be located in data centres, network nodes and end user premises [24]. In NFV, network functions are implemented in software that can run on a range of industry standard server hardware, and that can be moved to, or instantiated in, various locations in the network as required, without the need for installation of new equipment [24].

NFV is applicable to any data plane packet processing, control plane functions and innovative third party applications. Some of the key benefits of NFV are: increased velocity of time to market, targeted service introduction based on geography or customer sets is possible, optimizing network topology in near real time based on the actual traffic patterns and service demand, support for multi-tenancy thereby allowing operators to provide tailored services for multiple users and Improved operational efficiency.

2.2.4 Cloud and Virtualization

The separation of software from hardware and the enablement of horizontal cloud-based solutions has transformed the IT industry over the past decade [23]. This transformation has fundamentally been made possible by the use of hypervisors which decouple the application and software environment inside the Virtual Machine (VM) from the underlying hardware resources [2] [23].

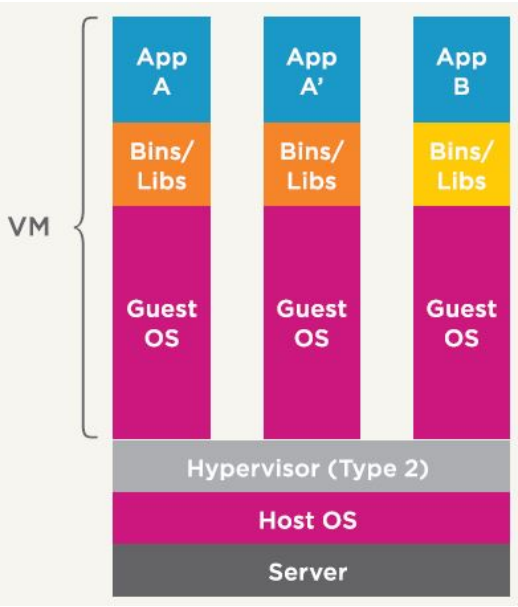


Figure 2.1: Structure of VM environment [2].

Multiple VMs can be deployed in a single platform as shown in Figure 2.1 and allowed to share the hardware resources in a controlled, efficient and flexible manner. Inter-VM communication is enabled by virtual switches in a robust and secured way [23]. Traffic can be routed to a VM from a physical interface and subsequently routed from the VM back to the physical interface.

Cloud solutions make use of virtualization technologies to provide multi-tenant computing and storage resources on-demand. This approach hence introduces new levels of automation, elasticity and flexibility in network and service deployments as well as a faster innovation cycle for top-line growth.

Cloud and virtualization technologies are being leveraged by Telco Cloud and Network Function Virtualization (NFV) [2]. They are currently transforming the communications industry in the way in which the IT industry was transformed over the past decade. These technologies are also key enablers for fog computing as they will allow applications to be deployed and run on top of the platform in a flexible, efficient and scalable way.

2.2.5 Rise of Containers

The ascendency of containers and associated micro-services is arguably one of the biggest fundamental changes to occur in IT in the last year [2]. Momentum for containers is building in industry with a focus on using micro-services architecture as a platform for

managing large, distributed applications with flexibility and agility.

Similar to traditional virtualization technologies, container technology is the solution to transparently deploy tasks on heterogeneous devices. However, different from traditional hypervisor-based virtualization technologies that may be too heavy-weight for lower-end fog resources, containers are lighter weight and more scalable as summarized in Table 2.2 [26]. Besides, the short startup time is also helpful for real-time IoT applications and fast deployments. Furthermore, if developers need to change some algorithms of an application or the configuration of operator graphs, they can easily launch a whole new container, since the overhead of restarting a container is small [26].

Table 2.2: Comparisons between containers and virtual machines [26].

	Container	**Virtual Machine**
Start Up	Seconds	Minutes
Capacity	MB	GB
Efficacy	Almost native	Slow
Scalability	Thousands	Dozen

The arrival of container solutions makes it simpler to package applications in a way that makes it possible to deploy them anywhere at will. Containers provide a higher level of abstraction that eliminate the need for navigating multiple types and classes of VMs. As shown in Figure 2.2, unlike VMs, containers can share the same Operating System (OS), thereby virtualizing at the OS level rather than at the hypervisor level.

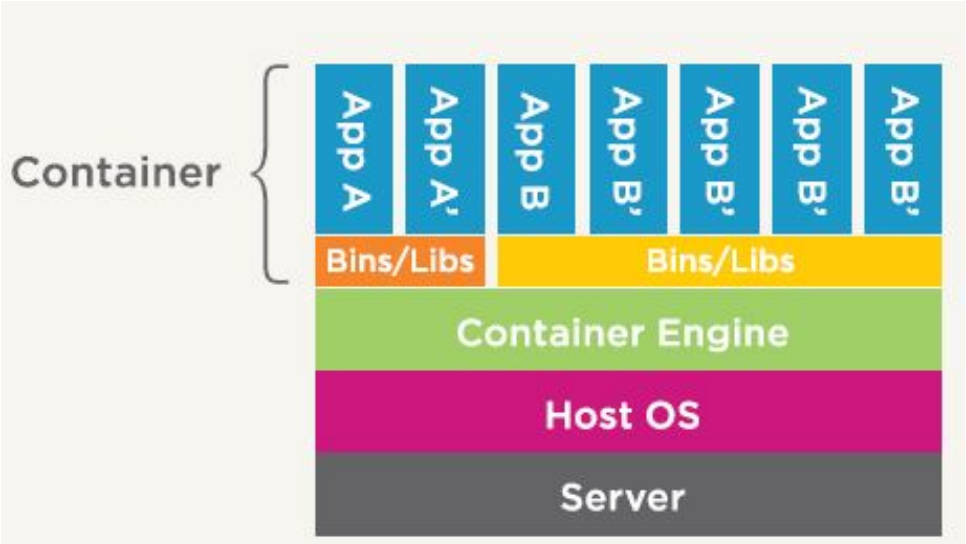


Figure 2.2: Structure of container environment [2].

2.3 Standardization Efforts Around Edge Computing

In recognition of the need to promote in-depth industry coordination, accelerate innovation, facilitate global market growth and boost the application of edge computing; several industry entities have joined together to push forward for the sustainable development of the edge computing industry. These collaborations facilitate the development of favourable market conditions which will create sustainable business for all players in the value chain [2]. Also, collaborations allow for a broader coalition of knowledge to the prevalent technical and market challenges. Activities within the established liaisons include identification and sharing of best practices, collaboration on standardization, test beds, research and development projects.

The three leading industry organizations driving for a standardized open edge computing environment are European Telecommunications Standards Institute (ETSI), the OpenFog Consortium and the Open Edge Computing Initiative. This section presents a synopsis of the goals and focus of each of these three organizations.

2.3.1 ETSI MEC ISG

The European Telecommunications Standards Institute (ETSI) creates globally-applicable standards and specifications for Information and Communications Technologies (ICT), including fixed, mobile, radio and internet technologies [23] [27]. Despite their name, their standards are globally applicable and not just for Europe. The ETSI is an independent and non profit association with over 800 member companies and organizations, drawn from 68 countries that participate directly in its work [27].

Within ETSI is the Multi-Access Edge Computing (MEC) Industry Specification Group (ISG) with a mission to standardize the architecture and interfaces for edge computing [23]. The ISG believes that standardization will make seamless the integration of applications and various software components required to make a MEC environment. MEC's work addresses multiple multi-access edge hosts deployed by different operator-owned networks which run edge applications in a collaborative manner [23]. In addition to standards, the ETSI MEC has a Proof of Concept (POC) framework where an ecosystem of companies can collaborate to prove out a specific MEC use case.

2.3.2 OpenFog Consortium

The OpenFog Consortium was founded on the posit that open architectures and standards are essential for the success of a ubiquitous fog computing ecosystem [1]. OpenFog argues that proprietary and single vendor solutions produce limited supplier diversity, which can have a negative impact on system cost, quality, market adoption and innovation [1]. The mission of the OpenFog Consortium is to speed up the deployment of fog computing technologies by enabling architecture, testing, composability and interoperability around fog computing.

The OpenFog Consortium was formed in November 2015 by ARM, Cisco, Dell, Intel, Microsoft and Princeton University [27]. The organization's goal is to accelerate the adoption of fog computing by solving latency, bandwidth and communications challenges around current and upcoming technology developments such as Artificial Intelligence (AI), IoT and tactile internet.

OpenFog is neither a standards body or an open source project, but rather a group that establishes frameworks, best practices, test beds and influences other standards body such as ETSI MEC [9]. In that sense, the consortium complements rather than competes with ETSI MEC. Moreover, in contrast to ETSI MEC, the consortium is approaching the problem of supporting mission critical and data dense next generation applications from an IoT point of view.

2.3.3 Open Edge Computing Initiative

The mission of the Open Edge Computing Initiative is to promote the adoption of edge computing amongst application providers, telecoms and cloud service providers [28] [2]. It does so by creating reference implementations, demonstrations and a testing centre called the Living Edge Lab located at Carnegie Mellon University (CMU) in the United States of America [28].

Open Edge Computing members include CMU, Nokia, Crown Castle, Intel, Deutsche Telekom and Vodafone. True to their mission, the Open Edge Computing Initiative's website has a number of demo videos [2]. Their GitHub repository [29] hosts the Carnegie Mellon Elijah project that has resulted in OpenStack++, an extension of OpenStack to support edge clouds often referred to as cloudlets.

2.3.4 OpenFog-ETSI MoU on Fog and Edge Applications

OpenFog Consortium and the ETSI signed a Memorandum of Understanding (MoU) to reduce the technical overlap across the multitude of fog domains and to pave a way forward for businesses working on 5G, mission-critical and data dense applications through fog computing [27].

The two organizations agreed to collaborate on standardization and interoperability requirements that will foster the deployment of fog computing technologies using open architectural frameworks [27]. The alliance believes that by adopting APIs across the OpenFog and MEC architectures, this will make it easier for developers design a common architecture, integrate management strategies and write application software modules capable of running on both architectures [27].

One of the first initiatives from the agreement will be focused on APIs that support edge computing interoperability. To that effect, the ETSI released in 2017 a package of MEC APIs [30] with important properties that can be adapted and used in the OpenFog reference architecture. These APIs are contained in the following specification documents [30]: GS MEC 009 [31], GS MEC 010-2 [32], GS MEC 011 [33], GS MEC 012 [34] and GS MEC 013 [35]. These API specifications address general principles for mobile edge service APIs, application life-cycle management, mobile edge platform application enablement, radio network information API and location API [30].

2.4 Edge Computing Reference Architectures

This section presents a technical overview of the reference architectures advocated by the ETSI MEC and OpenFog organizations.

2.4.1 ETSI MEC Reference Architecture

The ETSI Multi-Access Edge Computing (MEC) framework enables the implementation of mobile edge applications as software-only entities that run on top of a virtualization infrastructure located in or close to the network edge [36]. In its first phase, the ETSI MEC defined various hardware and software components that make up a MEC environment, defined APIs for applications to publish and consume services and outlined

how MEC applications should be orchestrated [23]. In next phases, the ETSI MEC activities envisions to work on technologies beyond 3rd Generation Partnership Project (3GPP) ones (hence the name MEC), add container support in addition to virtual machines, study new and innovative business models and finally investigate multi-operator use cases [23].

The MEC framework shows the general entities involved and these can be grouped into system level, host level and network level entities. The strong affinity between NFV and edge computing makes the ETSI reference architecture look remarkably like the ETSI NFV reference architecture.

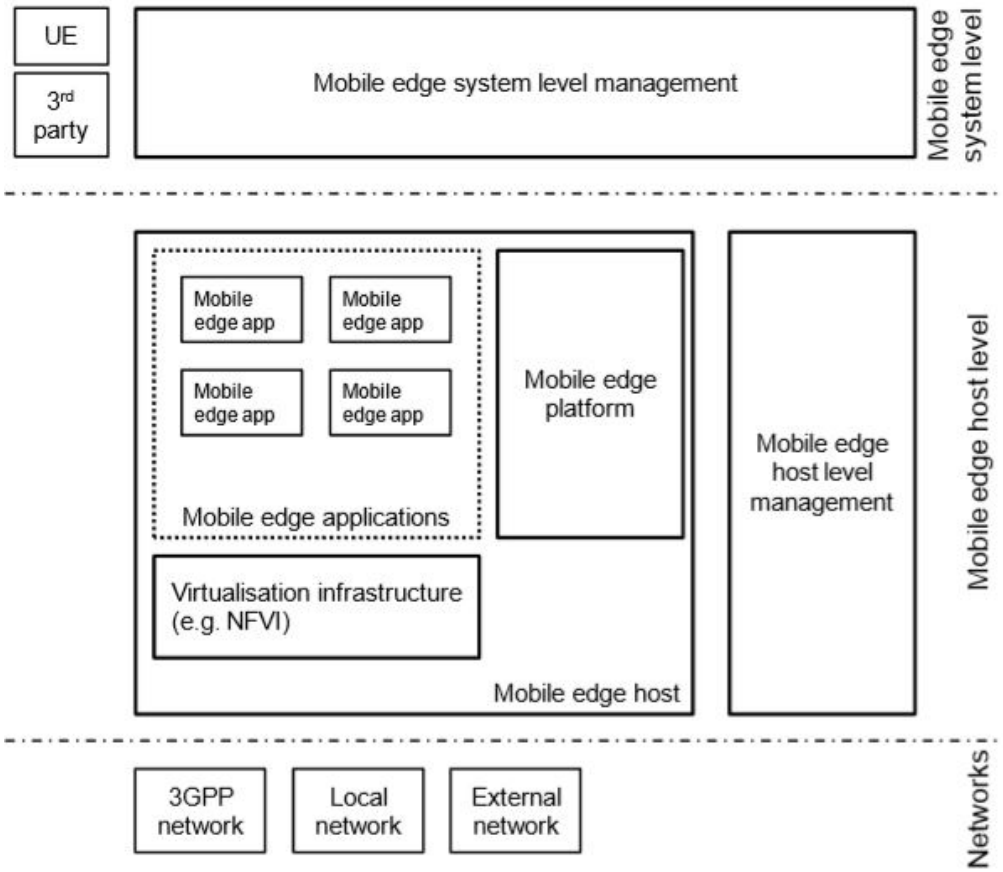


Figure 2.3: ETSI MEC reference architecture [36].

Figure 2.3 from the ETSI GS MEC 003 V1.11 specification document illustrates the ETSI framework for MEC, that is made up of the following entities [36]:

I. Mobile edge host made up of the following functional elements:

   (a) Mobile edge platform.

(b) Mobile edge applications.

(c) Virtualization infrastructure (virtual compute, data plane acceleration, virtual storage and virtual networking).

II. Mobile edge system level management.

III. Mobile edge host level management.

IV. External related entities such as network level entities.

Networks

This layer is composed of the different network entities to which the MEC node can be attached to. In the latest MEC specification, only 3GPP networks are currently supported [36].

Mobile Edge Host Level

This layer of the MEC framework is subdivided into mobile edge host and mobile edge host level management entities.

The mobile edge host includes the mobile edge applications, the virtualization infrastructure and the mobile edge platform which offers an environment and essential services needed by the mobile edge application [36]. Mobile edge applications run as VMs on top of the virtualization infrastructure provided by the mobile edge host. Also, mobile edge applications may have certain rules and requirements associated with them such as resources required, requisite services and maximum latency [23]. The virtualization infrastructure has a data plane for executing traffic rules and for routing traffic between applications, services, DNS server/proxy and various networks. The mobile edge platform offers an environment for applications to advertise, discover and consume services. It also receives traffic rules, DNS records, provides persistent storage and time of day information. Other essential services offered by the mobile edge platform may include radio network information, location service and bandwidth manager that are all available via standardized APIs [36].

The mobile edge host level management includes the mobile edge platform manager and the Virtualization Infrastructure Manager (VIM) [36]. The mobile edge platform

manager manages life-cycle of edge applications, provides element management for them and performs other management tasks like service authorizations, traffic rules, DNS configurations and resolving conflicts [36]. The VIM works as the cloud software layer plus the SDN controller [2].

Mobile Edge System Level

The core elements in the mobile edge system level layer are the mobile edge orchestrator and the Operating Suppport System (OSS) [36]. The mobile edge orchestrator is a critical piece of the architecture responsible for on-boarding applications, performing infrastructure selection, triggering application relocation as needed and tracking an overall view of available resources [23]. Some orchestrators also monitor applications by performing analytics on metrics, events, alarms and logs [23]. There are still some outstanding challenging in the MEC framework. Given the sheer number of edge locations, the task of managing and coordinating applications across these locations is going to be an interesting challenge [2]. This gets even more complicated with mobility. As users move across edge locations, the orchestrator is expected to move the application, state or configuration across locations.

The OSS refers to the Operating Suppport System (OSS) of the operator. This layer performs traditional operations and management tasks and grants application requests from User Equipment (UE) [36]. This layer of software also includes user application life-cycle management proxy that assists with application life-cycle management in response to UE events [36]. In latter releases, the OSS will receive requests from application on user equipment for relocating applications between external clouds and the mobile edge system.

2.4.2 OpenFog Reference Architecture

OpenFog is an architectural advancement from traditional closed systems and burgeoning cloud-only models, to an approach that emphasizes computation nearest the edge of the network [9]. The use cases for fog are dictated by business concerns and or critical functional requirements of the system [9]. To enable a common vocabulary and help support cross-organizational technical collaboration, OpenFog uses the ISO/IEC/IEEE 42010:2011 international standard as the guideline for describing architecture to

stakeholders [1]. Rather than defining a monolithic architecture, the OpenFog reference architecture OPFRA001.020817 specification document lays out the requirements from three different angles which are [1]:

- **Viewpoint:** A viewpoint is way of looking at a system. Currently, it is limited to functional and deployment viewpoints.
- **View:** A view is a representation of the structural aspects of the architecture. In the current revision, the structural aspects are limited to software view, system view and node view.
- **Perspective:** A perspective is a cross cutting aspect of the architecture. In the current revision, the perspectives are limited to security, performance, manageability, data analytics and control, IT business and cross fog applications.

This reference architecture, unlike ETSI MEC's, stays at functionality and interface level but delves deeper into topics that might concern hardware manufacturers, system architects, system integrators, software manufacturers and application developers.

OpenFog Viewpoint

The functional viewpoint outlines an architecture that addresses the various concerns of stakeholders that are required to satisfy specific use cases or scenarios [1]. Usually, each scenario focuses on the different aspects and market opportunities for fog computing. Visual security is the first scenario targeted by the OpenFog architecture [1]. This scenario will drive the functional viewpoint and also form the basis of the test bed infrastructure used to validate and refine the architecture [1].

Next, the RA describes the N-Tier deployment viewpoint. An example is shown in Figure 2.4.

The idea is that processing is not done in one location, but in a tiered continuum of locations along the edge to the centralized cloud with the goal of offloading some traditional workloads from the centralized cloud onto fog nodes. This way, the nature of processing changes along the hierarchy; fog nodes closer to 'things' will spend more time on data collection and processing; whereas nodes farther up perform data aggregation, analytics and generate intelligence [2]. Finally, the RA provides guidance on the number of tiers based on a number of criteria such amount of work required by each node.

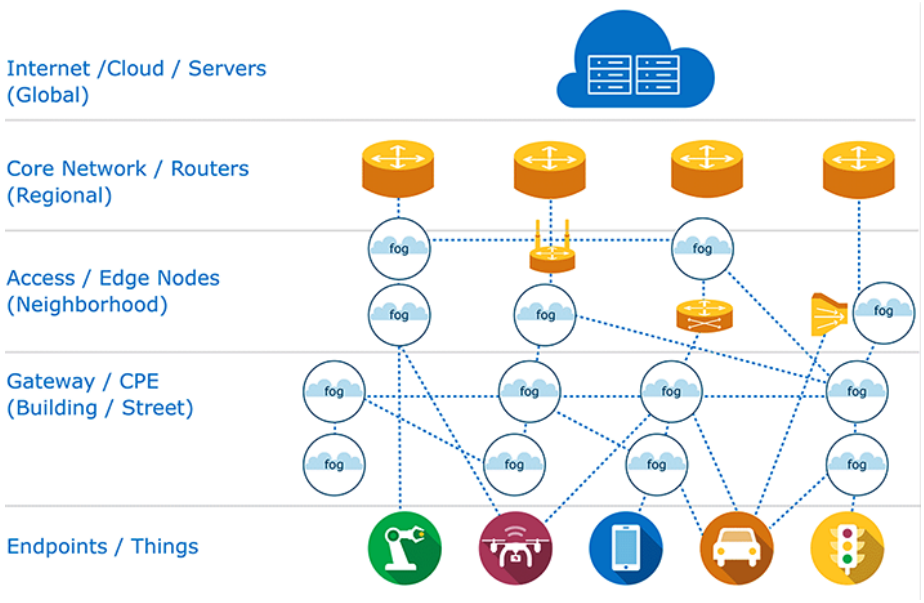


Figure 2.4: Example deployment viewpoint of the OpenFog reference architecture [2].

OpenFog View

The OpenFog view helps identification of the requirements for each stakeholder in the fog computing continuum [1]. This allows the various stakeholders like hardware manufacturers, system architects, system integrators, software manufacturers and application developers to delve deeper into topics of concern around the requirements. The RA covers three views: software, systems and node views. The top three layers comprise the software view. The next five layers make up the systems view. The node view consists of the lowermost layers. Figure 2.5 shows the OpenFog RA views.

OpenFog Perspectives

OpenFog perspectives are aspects of the architecture that affect all three layers of the view described in Section 2.4.2. The current perspectives are [1]:

i. Performance and scale.

ii. Security.

iii. Manageability.

iv. Data, analytics and control.

v. Information Technology business and cross fog applications.

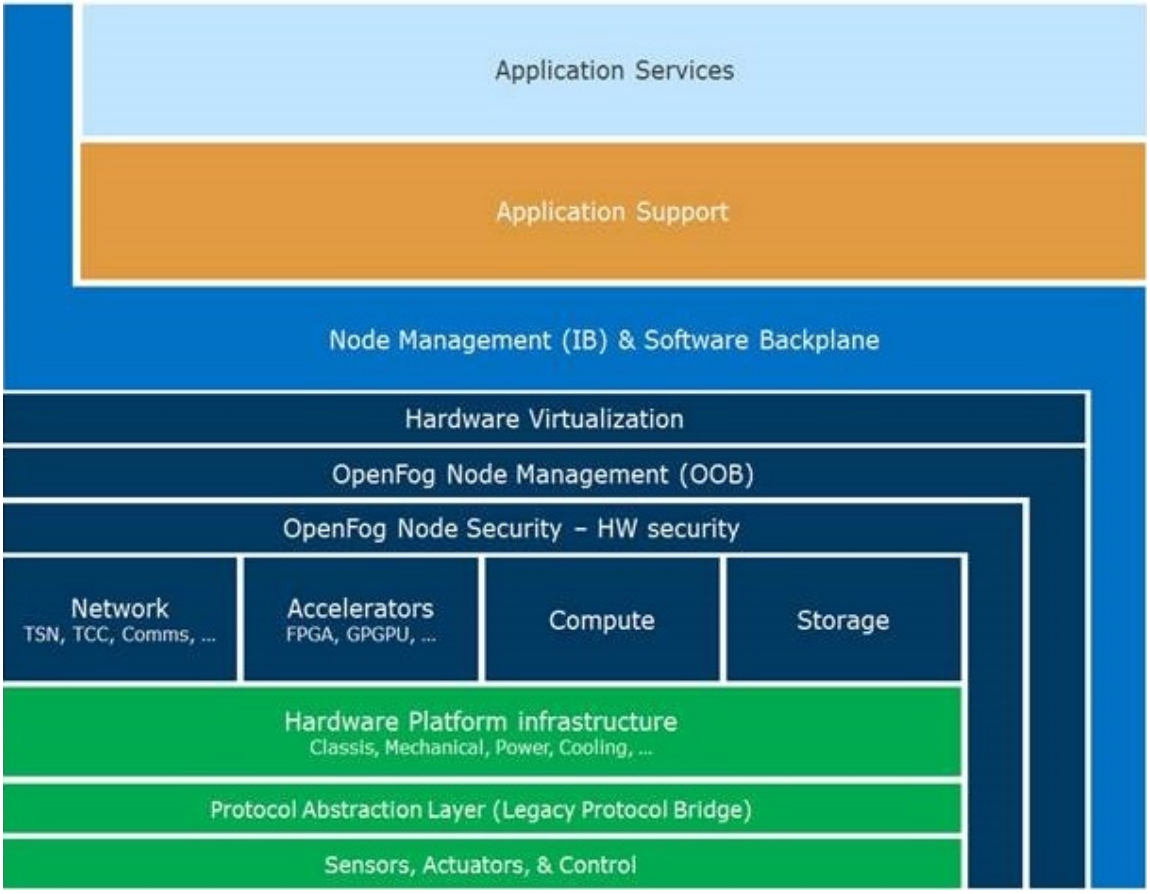


Figure 2.5: OpenFog reference architecture views [2].

Each of these items are discussed in detail in the OpenFog reference architecture [1] document.

2.5 Cloud Computing Orchestration Framework

This section presents the widely adopted three-layer orchestration framework in cloud computing. Since fog computing is an extension of cloud computing, existing solutions in cloud computing serve as important references to tackle similar issues in fog computing.

Orchestration refers to the processes of managing and coordinating the physical computational resources provided by the underlying infrastructure to serve the applications [37]. Figure 2.6 depicts a classical three-layer orchestration framework for cloud computing. The framework was initially suggested by the National Institute of Standards and Technology (NIST) and then widely adopted by modern cloud computing systems, such as OpenStack [38].

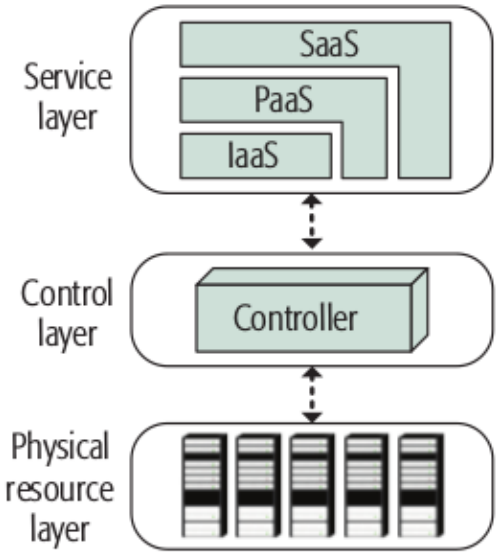


Figure 2.6: The orchestration framework in cloud computing [37].

In the framework, the top service layer provides abstractions for cloud applications to access the computing services. Typically, the services can be classified into three types, Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS), depending on the level of abstraction that is provided. The central controller in the control layer allocates the resources in the lower physical resource layer to the cloud applications in the upper service layer.

2.6 Literature Review and Related Work

This section reviews the current state of the art in cloudlet technology. The concept of cloudlets, which can be considered as a "micro data center in a box or node", has been widely applied in moving remote cloud resources to the edge of the network closer to users. However, the term fog node is still a nebulous expression with regards to what hardware appliances constitute as edge devices. In some implementations, fog nodes are realized as low power processing devices such as small single-board computers, mobile devices or network gateway devices. In other implementations, more powerful dedicated servers, VMs or cloudlets are used to realize the fog nodes.

The scope of this review is limited to cloudlet technologies for the following reasons. Firstly, cloudlets are more powerful, store more data and perform more robust computations than single-board computers or network gateway devices. Secondly, the attributes of cloudlets such as plug-in design, support for multi-tenancy and

virtualization-capable make them more closely linked to the OVAFF's attributes. Finally, for enterprise use cases of fog technology, it is highly unlikely that the implementations will be realized on clusters of resource constrained devices.

The rest of this section discusses the research activities, existing works and their major contributions in the area of developing computational fog computing architectures for mission critical and data dense applications. A review is conducted thereafter to discuss the limitations in these works and how the OVAFF addresses these gaps.

2.6.1 Related Works

Fog computing is relevant across the whole stack from the hardware to the applications. The scope of an open fog ecosystem is therefore extremely large. For this reason, the Open Vendor Agnostic Fog Framework (OVAFF) solution presented in this thesis predominantly focusses on the Infrastructure-as-a-Service layer. Moreover, fog computing is still quite a recent area which makes the state of the art incipient regarding architecture definitions, middleware and real world implementations. The bulk of existing publications on the subject matter of fog architectures are currently centred on specific application use cases in selected vertical domains. Additionally, these frameworks cannot be extended to other markets unlike the OVAFF. To the best of my knowledge, the concept of integrating or extending the main edge computing reference architectures has not being covered in any existing publications and this is the novelty of this work. It is for these very reasons that there are limited related works to present in this Section. The related works presented below focus on relevant cloudlet frameworks whose objectives form a subset of the outlined attributes of the OVAFF. A brief explanation for the choice of these related works to be presented has already been covered in the introduction of Section 2.6.

Sun and Ansari [39] propose a cloudlet based network architecture for analysing User Equipment (UE) data streams at the mobile edge. Each base station is attached to a cloudlet that serves as a distributed tiny data centre. Each UE is then associated with a specific dedicated VM providing private computing, communications and storage resources to the UE. Data streams generated from UEs are uploaded and analysed in virtual machines of the cloudlet with low end-to-end network delay. Finally, UEs and virtual machines in the cloudlet network communicate with public data centres via the Internet.

Li and Nabrzyski [40] propose a heuristic optimization algorithm to solve the VM

placement problem in a cloudlet mesh. A cloudlet mesh is a series of cloudlets that are connected by a wireless mesh network to serve a large number of IoT applications. The VM placement problem in meshed fog is critical especially in multi-tiered fog deployments where workload placement is critical because of the maximum bandwidth capacity restrictions in inter-cloudlet communication. The work of Li and Nabrzyski provides a heuristic solution for the data centre placement problem in multi-tenant scenarios.

Carames, Lamas and Albela [41] present an industrial Augmented Reality (AR) architecture which is based on cloudlets and the fog computing paradigm. The framework is evaluated in real world scenarios for varying AR payload sizes and when using a cloud, cloudlet and fog devices. Aside from providing a practical application of a data dense and latency sensitive use case, this work also evaluates the performance of cloudlets versus fog devices for varying payload sizes and high load scenarios.

2.6.2 Discussion of Related Works

There are several similarities that can be drawn from the presented works and the OVAFF. Firstly, all of the related works demonstrate that cloudlet based fog deployments are a better fit than deployments of swarms of resource constrained devices in terms of elastic scalability and performance for compute-intensive services. These works also use the fog layer for computationally heavy tasks and forward the summarized results to a cloud for persistence. The design of the cloudlets used by the authors in [40] is plug-in, a design construct that is used in the OVAFF. Moreover, all of the mentioned works leverage on the power of virtualization technology for the cloudlets.

However, there are several limitations in these frameworks that are addressed in the OVAFF. The works presented by the authors in [40] and [41] do not mention how workload placement, management and orchestration will be achieved in the cloudlets. This is unlike in the OVAFF where there is a logical centralized MANO function for this. These works also lack alignment of their fog computing model with any of the main edge computing reference architectures discussed in Section 2.3. Moreover, all of these works provide a practical demonstration of achieving fog computing at the network edge. However, none of these works constitute as a reference architecture that can easily be extended and applied in any of the relevant industry vertical markets outlined in Section 1.1.2. Finally, the related works mainly concentrate on the computing capabilities at the fog, which limit the extensibility of these works to the enterprise where various business support

systems are needed in place for the commercial, operations and governance requirements of these systems.

2.7 Summary

The chapter has presented the state of the art regarding the topics of digital disruption, cloud computing, fog computing and standardization efforts around fog solutions. It is clear that these topics are highly popular in the research community. There are a few implementations that showcase proof of concepts but the bulk of published literature is leaning towards the performance enhancements introduced by the fog and not the focus of this thesis, which is motivating for compatible fog solutions that will alleviate the current fragmented fog marketplace and ecosystem. This thesis places an emphasis on these identified gaps and presents solutions catering for this. The next few chapter goes into greater detail on the requirements, design considerations and solution architecture of the OVAFF. This solution architecture is subsequently implemented and evaluated in latter chapters.

Chapter 3

Requirements and Design of the OVAFF

The previous chapter introduced literature and related works to give some background on the subject matter. This chapter provides an overview of the envisioned fog ecosystem and its requirements. Then, in-depth detail on the design of the proposed architecture follows.

3.1 Overview of the Envisioned Fog Eco-System

The proposed fog architecture relies on two layers; a fog node host layer and a management and orchestration layer. Fog node hosts serve as nano cloud data centres for running fog applications whereas the Fog Management and Orchestration (MANO) handles management and orchestration of the large numbers of fog applications and nodes that are geographically sparse.

The orchestrator is a mandatory logically centralized server application for the headquarter of any organization with fog deployments. Access to fog nodes is established from the subscription and policy agreements an organization has with the fog node infrastructure providers. An organization with fog deployments can either be a fog node infrastructure provider or a lessee of fog cloud computing resources either from either third party fog infrastructure providers (i.e. fog market place) or from federated infrastructure sharing agreements.

Figure 3.1 shows a sample topology diagram of the envisioned fog ecosystem. The diagram is partitioned into a fog layer and a cloud layer. In the example, the fog layer is made up of N (where N ∈ Natural Numbers such that N = {0, 1, 2, ...}) distinct geographical areas that are inter-linked by the public internet via optic fibre and RAN links. In each area, there is a distinct organization with fog node infrastructure for serving its intelligent devices and fog based services.

Through the fog market place, organization 1 in area 1 has access to infrastructure resources of fog node host 2 in area 2. Therefore, organization 1 can deploy its fog services and distribute processing in the hierarchy (fog node 1 ⇔ fog node 2 ⇔ public cloud). Each organization has its own orchestrator for conducting MANO tasks outlined in Section 3.6.

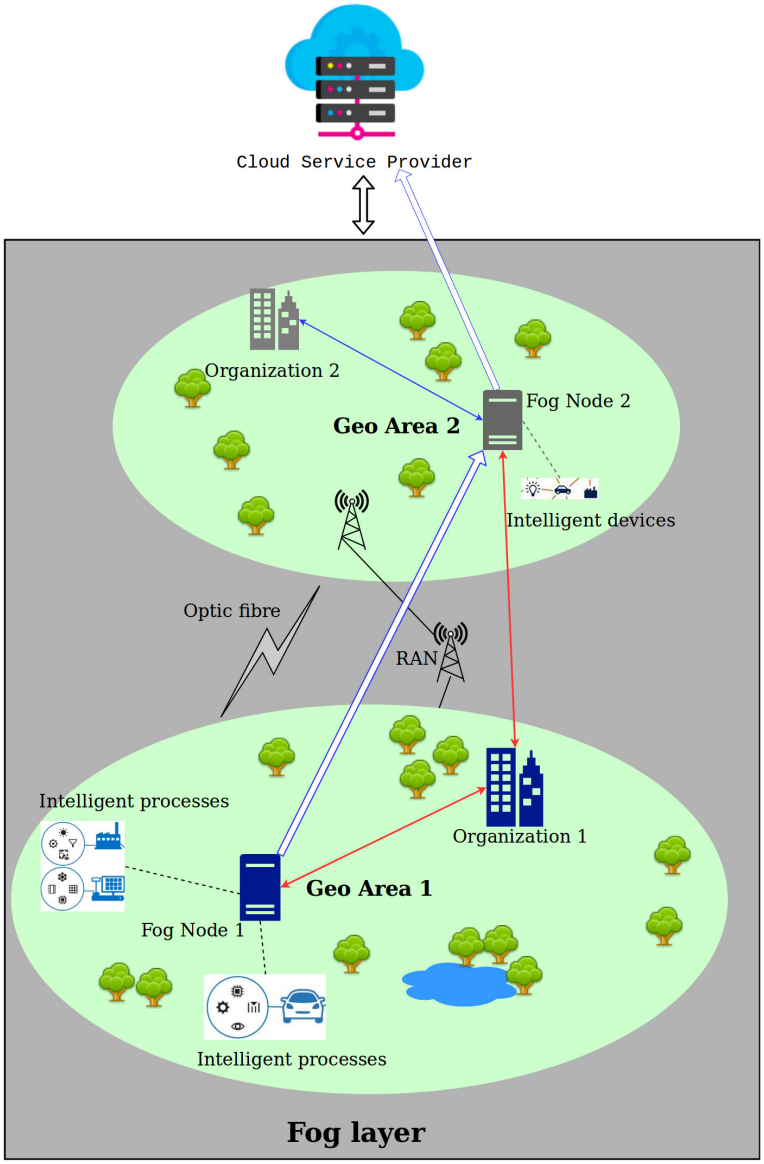


Figure 3.1: Overview of the envisioned fog ecosystem.

3.2 Key Design Considerations for OVAFF Deployments

This section presents some foundational design aspects to be considered in choreographing the roll out of OVAFF deployments. Architecture in the context of fog computing refers to the arrangement of physical and logical network elements, hardware and software to implement a cross-functional fog network. The key architectural decisions involve [6]:

i. The physical and geographical positioning of fog nodes.

ii. The arrangement, numbers, topology and protocols of fog nodes in a hierarchy.

iii. Data bandwidth capacities of the links between fog nodes, things and the cloud.

iv. The hardware and software design of individual fog nodes.

v. How a complete fog network is orchestrated and managed.

Overview of N-tier Fog Deployment Strategy

In most fog deployments, there are usually several tiers (N-tiers) of fog nodes. Tiers are created in order to deal efficiently with the amount of data that needs to be processed and to provide better operational and system intelligence [1]. In general, each level of the N-tier environment sifts and extracts meaningful data to create more intelligence at each level. Appendices A.1 and A.2 present the different available tiering strategies and their use cases. In the tiered setups, the nature of processing is such that [1]:

i. Fog nodes at the edge closer to the sensors and actuators are typically focussed on sensor data acquisition/collection, data normalization and command/control of sensors and actuators.

ii. Fog nodes in the next higher tier are focused on data filtering, compression and transformation. They may also provide some edge analytics for critical or near real time processing.

iii. Nodes at the highest tiers or nearest to the backend clouds are typically focussed on aggregating data and turning the data into knowledge.

Determination of the Number of Required Fog Tiers

Fog deployments are envisioned to come either as large or small scaled and this is all based on the scenario being addressed. The number of tiers in the fog deployment are dictated by scenario requirements, which can include [1]:

i. Number of sensors.

ii. Amount and type of work required by each tier.

iii. Capabilities of the fog nodes at each tier.

iv. Required reliability and availability of fog nodes.

Moreover, fog nodes may be linked to form a mesh for providing load balancing, fault tolerance, resilience, data sharing and minimization of cloud communication [1].

Dimensionality and Uniformity Constraints of Fog Node Hosts

Physical architectural elements of a node vary based on its role and position in the N-tier fog deployment [1]. As previously described, nodes lower in the hierarchy may need to be architectured with less processing, communications and storage than nodes at higher levels. However, Input/Output (I/O) accelerators required to facilitate sensor data intake at the edge may be much larger in aggregate than I/O accelerators designed for higher level nodes [1]. Figure 3.2 illustrates the typical dimensionality of different physical computing elements of fog node hosts.

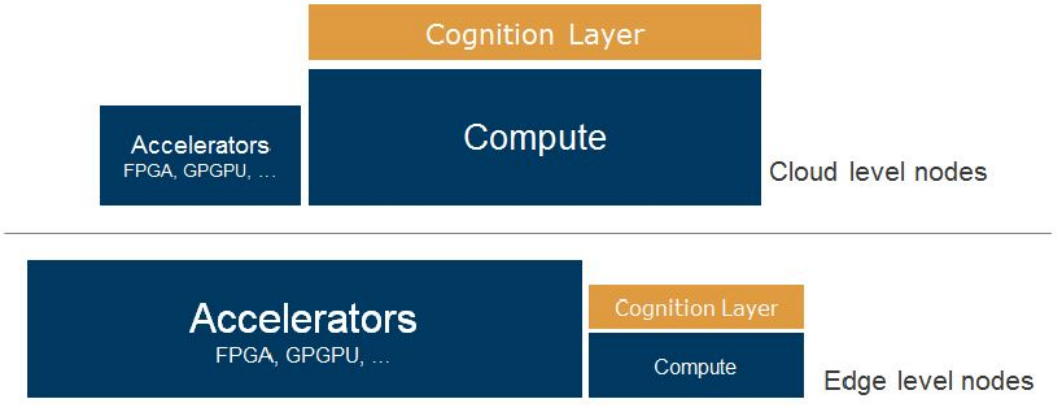


Figure 3.2: Dimensionality considerations for edge and cloud level nodes [1].

3.3 Key Requirements of the Envisioned Fog Eco-System

There are several key challenges inherent in fog platforms that will need to be overcome in the OVAFF. These challenges translate to system requirements which will form the basis of the OVAFF design architecture. These system requirements are the following [37]:

I. **Requirement to handle heterogeneity in fog environments:** The highly heterogeneous nature in the fog ecosystem is brought about by fog node hosts of different capabilities, hardware and software specifications. Fog applications launched on the OVAFF platform will need to be virtualized to isolate them from one another and to reserve heterogeneous resources. Virtualization allows for interoperability and transparent deployment of applications on the heterogeneous fog nodes. Interoperability enables heterogeneous fog nodes to operate under the same OVAFF architecture.

II. **Requirement of a universal programming model:** There is need for a suitable universal programming model for distributed fog applications. This is important to have so that applications distributed along the fog hierarchy can concurrently leverage resources from multiple tiered fog nodes. A universal programming model also brings software-programmability. Software-programmability eases the way for software developers to program based on general virtualized hardware where low level hardware details of fog nodes are shielded.

III. **Requirement of fog orchestration techniques:** The large number of virtualized applications and fog nodes require an orchestration tool to manage them. A well designed orchestration framework will enable fog applications to be partitioned into segments and deployed onto different fog nodes.

As previously mentioned, the OVAFF platform is built off the synergies drawn from the ETSI MEC and OpenFog reference architectures. ETSI MEC is based on Network Function Virtualization (NFV) and some architectural constituents from NFV are used in the design to satisfy the system requirements identified above. The rest of this section details how these requirements will be addressed in the OVAFF architectural design.

Figure 3.3 shows the NFV architectural framework illustrating the building blocks

vendors can choose to implement in order to provide NFV compatible nano-cloud data centres [25].

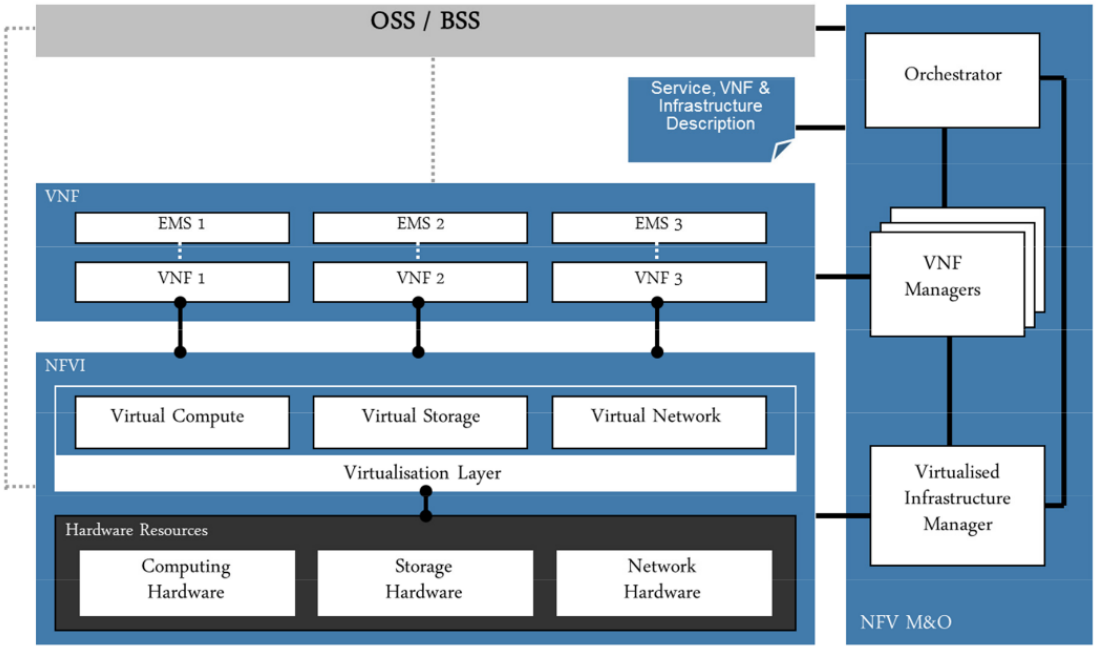


Figure 3.3: NFV Architectural Framework [25]

The OVAFF platform is based on the NFV specification to meet the first requirement for virtualization. NFV leverages standard IT virtualisation technology to consolidate many network equipment types onto industry standard high volume servers, switches and storage, which could be located in data centres, network nodes and in the end user premises [24].

Next, fog applications will run as virtual functions on top of virtualized resources. The requirement of a universal programming model is thus crucial for packaging and managing these applications. The principle is that fog applications will be developed in any high level languages of choice (for example Java, Python, C++ or Ruby) and then packaged into a virtual resource that can run on top of the virtualized OVAFF platform. That way, legacy applications can seamlessly be migrated into the fog platforms without any need for transformational changes to the underlying software applications. This dissertation will adopt the NFV approach of deploying fog applications as Virtual Network Functions (VNF) [24].

Finally, the orchestration framework should be compatible and integrable with the highly virtualized OVAFF platform. The orchestration tool used in this dissertation is adopted from the the high level design of the NFV MANO outlined in the ETSI NFV specification document [24].

3.4 Architectural Framework of the OVAFF

The overall OVAFF Architectural Framework is shown in Figure 3.4. It illustrates the base building blocks of the platform that vendors and fog service providers have to implement. The main building blocks of the framework are the Fog Node Host and the Fog MANO. The architectural constituents of these building blocks are further described and analysed in-depth in Sections 3.5 and 3.6 of this chapter.

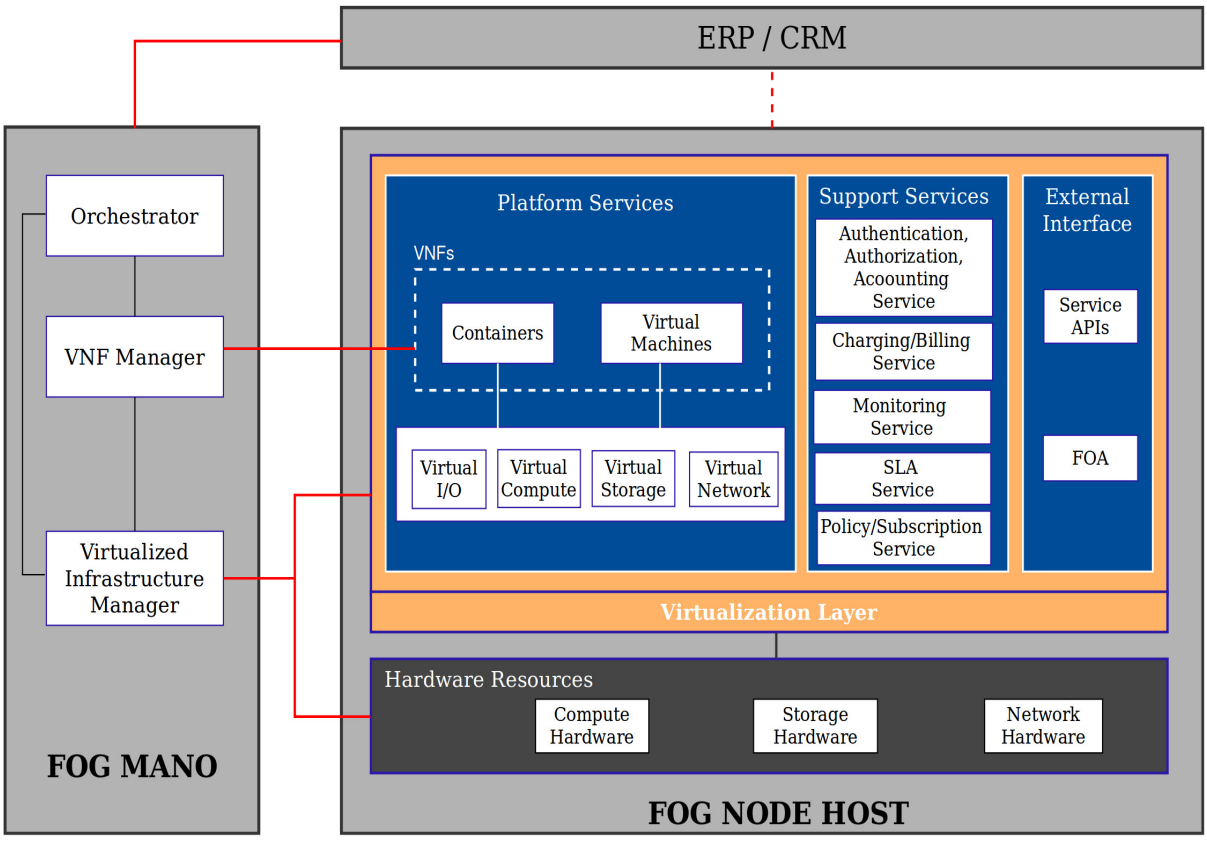


Figure 3.4: Mapping the Functional Architecture of the OVAFF to the NFV Architectural Framework.

The architectural framework constituents interact via defined reference points so that the various entities can be clearly decoupled to promote an open and innovative fog ecosystem.

In the context of the OVAFF, Network Function Virtualization Infrastructure (NFVI) illustrated in Figure 3.3 refers to the physical hardware and virtual computing resources on which the VNFs run on. VNFs refer to the packaged fog applications running on the NFVI either as containers or virtual machines.

The virtualization layer is a unified interface which decouples the software instances from the underlying hardware. This allows for hardware to be changed or upgraded on the fly without interfering with the applications. It also ensures an appropriate level of system resilience to hardware and software failures.

The reference points between VNF, NFVI and those between entities within the NFVI; deal with the abstraction and virtualization of resources from the hosted VNFs. This abstraction makes it possible to achieve high performant VNFs portable between different hardware vendors. This means that different choices of underlying hardware are possible [25]. This design hence allows for integrating multiple virtual appliances from different vendors. Moreover, fog infrastructure providers are therefore able to "mix and match" hardware, hypervisors and virtual appliances from different vendors without incurring significant integration costs and avoiding lock-in.

The reference points between Fog MANO and VNF; between Fog MANO and NFVI; as well as those between entities within the Fog MANO, deal with the management and operation of the OVAFF system. The related building blocks are designed in a way that allows the reuse of existing solutions such as cloud management systems and also interaction with existing ERP/CRM environment to which the OVAFF system needs to be connected to.

The Fog MANO also interacts with the external Enterprise Resource Planning (ERP) or Customer Relationship Management (CRM) landscape, which allows NFV to be integrated into an already existing network-wide management landscape. This design hence achieves co-existence of the OVAFF with pre-existing business support systems used in the enterprise for the commercial, operations and governance requirements of their IT systems.

3.5 Fog Node Host Architecture

The fog node host architecture proposed in this dissertation calls for a modular and plug-in fog node host composed but not limited of the following main components: platform services, support services, external and network interface. This design is modular enough to support the use cases stated in the OpenFog reference architecture [1].

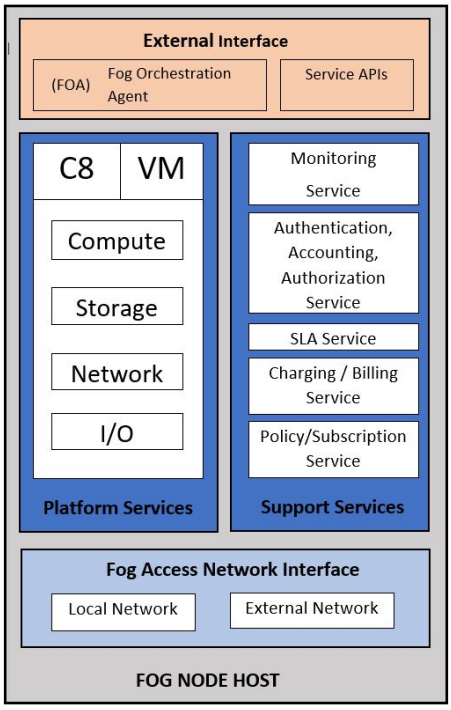


Figure 3.5: Fog Node host high level architecture.

Figure 3.5 presents a high level visual representation of the proposed fog node host architecture and its four core functional blocks which are:

I. **Platform Services (PS):** Provide physical and virtual resources (like computation, storage, Input/Output (I/O), networking) for fog applications. On top of these resources, runs Containers (C8) or virtual machines that encapsulate fog applications that will run as VNFs on the OVAFF NFVI.

II. **Support Services (SS):** These can be regarded as fog platform services that provide specific functionality common to all fog applications from different vertical markets and domains. The minimum set of requisite support services includes the monitoring service; authentication, authorization and accounting (AAA) service; Service Level Agreement (SLA) service, billing service and subscription service.

III. **External Interface (EI):** This serves as the service request point of entry into a fog node from the outside world. It is composed of two main services but not limited to the Fog Orchestration Agent (FOA) and Service APIs. The FOA abstracts the local Software Defined Networking (SDN) and NFV drivers necessary for the fog nodes to liaise with the Fog MANO. Also, the FOA is responsible for: start/stop resources when requested by the Fog MANO and when some event happens following some predefined rules, local management of services running in the fog node and fog node resource management [42]. Service APIs is a set of APIs that can be used by third parties to interact with platform and support services.

IV. **Network Interface (NI):** This is responsible for establishing communication channels with the underlying access points to which the fog node host will be plugged into. It is intended to be interoperable with various access points envisioned for the fog node host such as the RAN, PON OLT, radio network controller and CMTS. The NI also offers a standardized way to do Machine to Machine (M2M) communication taking into consideration different protocols whilst providing standardized access to fog node hosts [42].

Figure 3.6 shows a lower level visual graphic of the proposed fog node host architecture. This design is an extension and adaption of the ETSI NFV architectural framework for use in the fog computing paradigm.

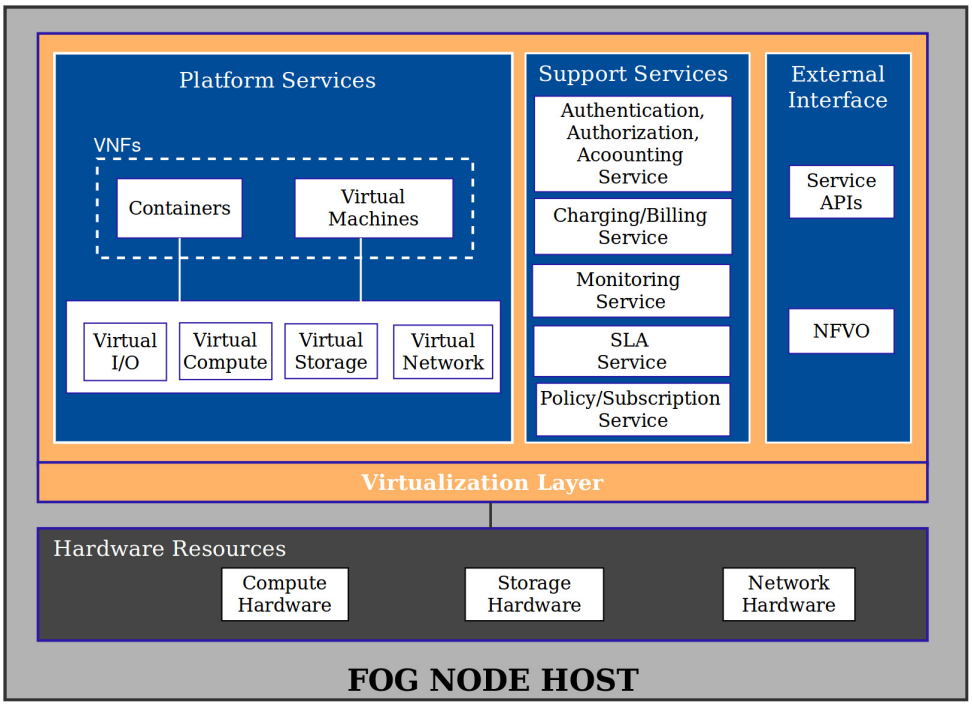


Figure 3.6: Fog Node host low level architecture.

3.6 Fog MANO Design

At the heart of the OVAFF is a centralized Fog MANO (Management and Orchestrator) that has accessibility to the fog marketplace and all fog node hosts that an organization has subscriptions to in federated infrastructure sharing agreements. The Fog MANO therefore manages this group of fog nodes as its physical infrastructure. It can organize fog nodes into groups, thereby enabling fog application providers to separate their utilized infrastructure resources into multiple domains [42]. Also, these logical domains can also be useful for separating nodes by their capabilities and objectives [42].

The MANO focuses on the virtualization-specific management tasks necessary in the fog node hosts. Additionally, the MANO is a critical piece of the architecture that on-boards and manages fog applications, performs global orchestration and infrastructure selection and keeps track of the global resources availability. The major tasks of the MANO are depicted below in Figure 3.7 [2].

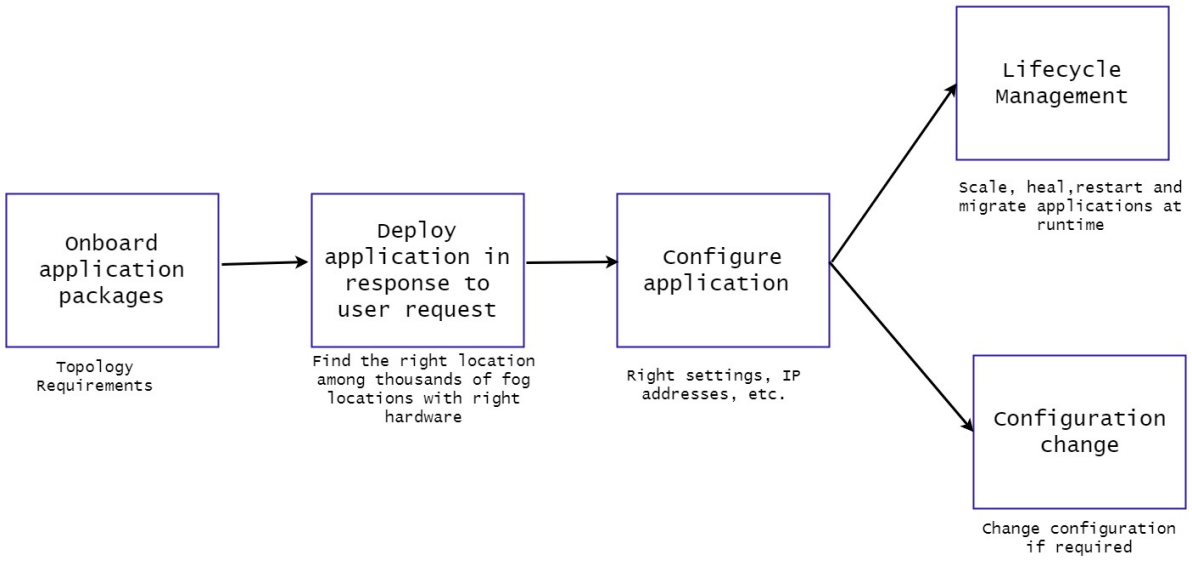


Figure 3.7: Summary of tasks performed by the Fog MANO.

3.6.1 Fog MANO Architecture

The completeness of the orchestration solution is based on the co-interaction of the Fog MANO with the Fog Orchestration Agent component in every fog node host. The interaction between these components is achieved through interfaces which were mentioned in Section 3.4. To standardize the management and orchestration of resources (computing, storage, networking, containers and virtual machines) in fog nodes, the

MANO is designed off some characteristic features borrowed from the NFV MANO [20] specification document, an ETSI-defined framework for cloud data centre management.

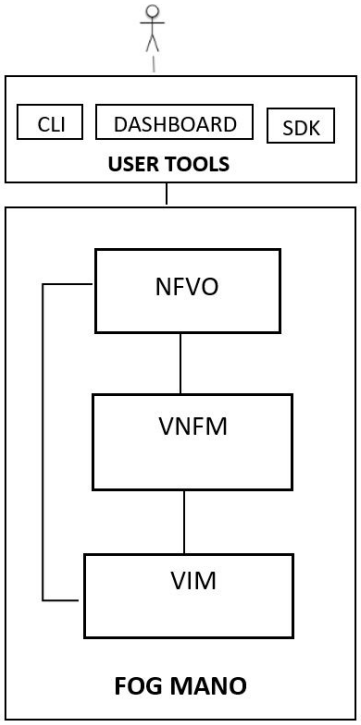


Figure 3.8: Fog MANO high level architecture.

Figure 3.8 presents a visual representation of our proposed Fog MANO architecture. The Fog MANO is composed of three main functional blocks:

I. **NFV Orchestrator (NFVO):** responsible for on-boarding network services (NS) and Virtual Network Functions (VNF) packages; Network Service (NS) life cycle management, global resource management; validation and authorization of NFVI resource requests.

II. **VNF Manager (VNFM):** oversees life cycle management of VNFs. Life cycle events supported can include instantiation, deployment, undeployment, scaling and healing/fault management policies.

III. **Virtualized Infrastructure Manager (VIM):** controls and manages the NFVI compute, storage and network resources.

IV. **User tools:** provides a programmable human computer interface between users and the MANO. It is composed of the command line interface (CLI), software development kit (SDK) and dashboard.

3.6.2 VNF Life cycle

As described in Section 3.5, one of the tasks performed by the MANO is life cycle management on the fog applications it deploys. This dissertation proposes six life cycle events for VNFs. These are shown in Table 3.1 below.

Table 3.1: Supported VNF life cycle events.

Event Name	Description
INSTANTIATE	Scripts that run at instantiation of VM instance
CONFIGURE	Scripts that run to configure the VM and VNF instance
START	Scripts that run at start-up of VNF instance
TERMINATE	Scripts that run at termination of VNF instance
SCALE	Increase dimensionality of VNF
MIGRATE	Migrate VNF instances during fault management

Figure 3.9 shows a state machine diagram of the life cycle events supposed by the proposed MANO.

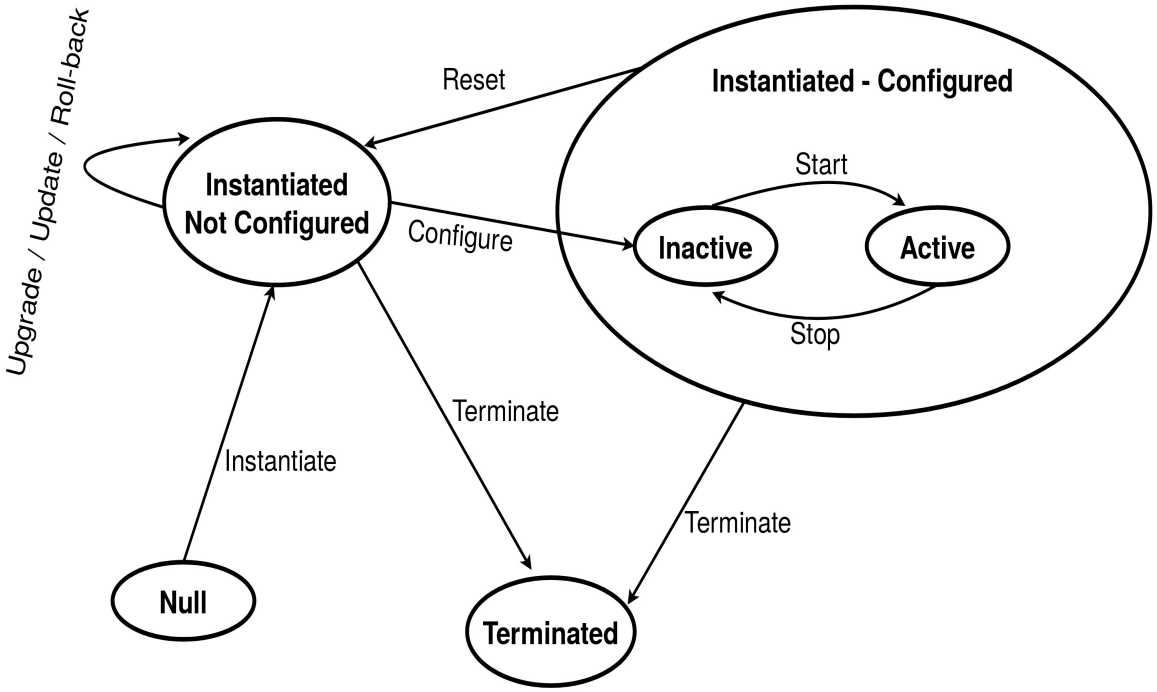


Figure 3.9: State machine diagram for life cycle events supported by the Fog MANO [43].

3.7 Fog MANO and Fog Node Host Operational Flow

The operation flow of on-boarding a Fog application from the Fog MANO onto Fog nodes is presented in Figure 3.10. Firstly, a user via the Dashboard or CLI Fog MANO console requests for a list of candidate physical locations and the capacity of computing resources available at each site. Next, using services in the SS module, the contacted fog nodes respond to the query thus allowing the user to make a decision on the choice of fog nodes to which to deploy their application. The user's choice of sites, fog application's bootstrapping scripts, VNF life-cycle policies, network configuration and specification of required computing resources for the VNFs are contained in a Network Service Description (NSD) that is uploaded by the user to the Fog MANO via the Dashboard or CLI. Finally, the Fog MANO deploys the fog application as a network service onto fog nodes specified in the fog application's network service description. At runtime, the SS of the fog nodes will send monitoring information of a fog application's VMs or Containers to the various functional blocks of the Fog MANO for runtime management and orchestration.

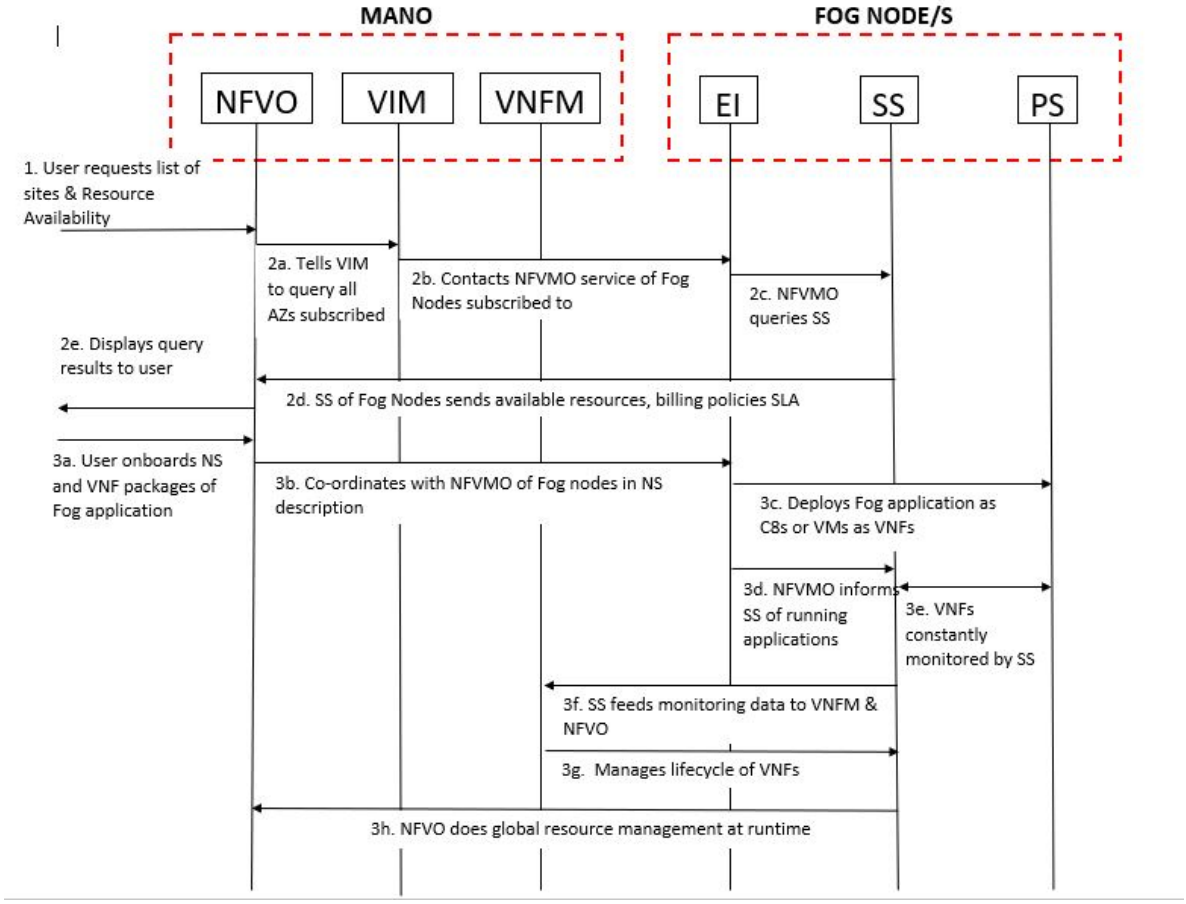


Figure 3.10: Fog Node Operation Flow for on-boarding a fog application

3.8 Discussion

The chapter has detailed the extensions and adaptation of the NFV architectural framework to meet the requirements outlined for the envisioned fog ecosystem. The proposed architecture looked at all levels of the design and integration of the components within and across the layers of the architecture.

The next chapter details the practical implementation and realization of the proposed architectures to show proof of concept and provide a platform for evaluations.

Chapter 4

Implementation of Proposed Framework

The previous chapter outlined the requirements of the envisioned fog solution, and then presented a design of its architectural framework. This chapter details the implementation of this proposed OVAFF solution.

To substantiate the envisioned feasibility of the OVAFF solution in typical deployment scenarios, there is need for a practical testbed implementation where evaluations on the framework can be carried out in a realistic environment. Whereas a simulation of the proposed solution is possible in Riverbed Modeler [44] software modelling tool, this will typically be subject to a number of assumptions that, though theoretically accurate, could omit some of the variables of a practical solution. In a testbed environment, deployed technologies can be proved almost beyond doubt.

The implementation of an evaluation framework was achieved on a testbed facility residing at the University of Cape Town (UCT). The UCT Communications Research Group has developed an extensive cloud testbed environment for SDN, NFV and Cloud, the enablers for mobile network architectures and future internet technologies.

The testbed implemented in this work's evaluation comprises entirely of Free and Open Source Software (FOSS), thereby opening up the system development for all. This means that the testbed components can be easily and legally re-instantiated, realized and extended by others, which is critical for the growth and establishment of emerging technologies.

4.1 Overview of Testbed Setup

This section describes a high level overview of the overall testbed setup and the software tools used to implement the testbed. Figure 4.1 below shows the physical machine make up and network interconnections in the data centre testbed.

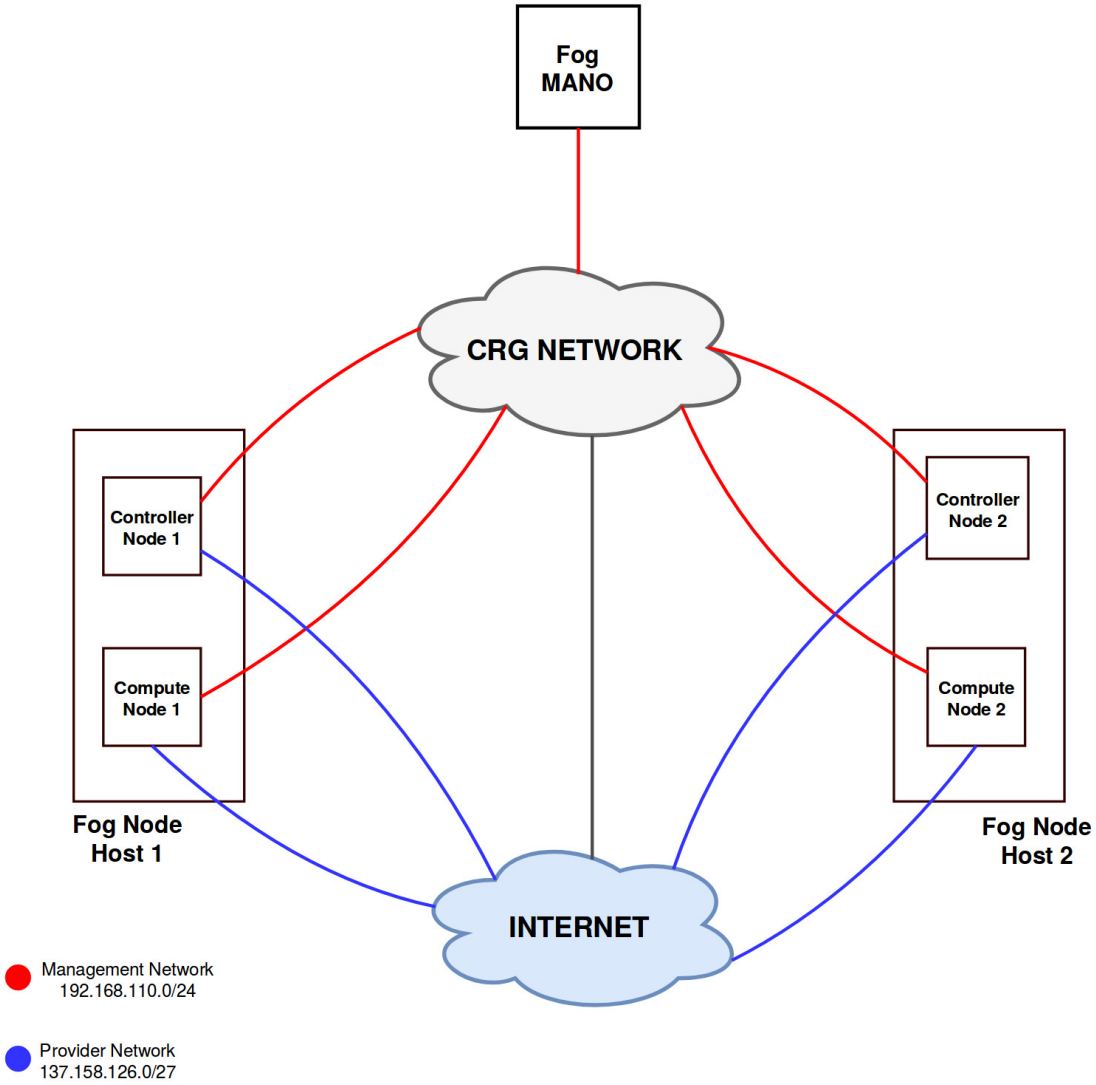


Figure 4.1: Testbed physical server and networking architecture.

The testbed consisted of two data centres resembling two independent fog node hosts that are interconnected over the public internet. The data centres run on standard Commercial Off-The-Shelf (COTS) hardware. For each data centre, a multi node approach was taken such that each data centre can be easily scaled horizontally on standard hardware by the addition of extra nodes. Moreover, this setup is cheaper and robust through the use of non-specialized PC hardware.

Nodes in a data centre need to communicate with each other and they also require

internet access for administrative purposes such as package installation, security updates, Domain Name System (DNS), and Network Time Protocol (NTP). As part of the networking best practices outlined in RFC 1918 [45], it is thus crucial to have networking separation for the management and provider networks. A management network was created within the CRG network to cater for inter-node communication within each data centre whereas a provider network was created to administrative tasks that require a node to break out to the internet. Table 4.1 shows the physical testbed networks.

Table 4.1: Physical testbed networks.

Network Name	**Network Address**	**Purpose**
Data centre management network	192.168.110.0/24	To carry management traffic between all physical nodes within a fog node data centre.
Data centre provider network	137.158.126.0/27	To allow nodes to break out to the internet for administrative tasks. Also assigns public floating IP addresses to VNF instances instantiated on Fog Node Hosts

The fog node hosts were implemented over a two node OpenStack platform. OpenStack is an open source cloud virtualization platform that enables deployment of VNFs using COTS hardware. The two nodes deployed are the controller and compute nodes. One OpenStack controller node runs the Identity service, Image service, management components of compute and networking services, networking plug-ins and the dashboard service. It also includes supporting services such as the SQL databases, the message queue manager that allows the two nodes to effectively communicate with each other, and the NTP service. The OpenStack controller node also runs portions of the orchestration and telemetry services. The compute node runs the Openstack compute service, an Infrastructure-as-a-Service (IaaS) system. This service interacts with OpenStack Identity for authentication; OpenStack Image service for disk and server images; and OpenStack dashboard for the user and administrative interface.

The Fog MANO was realized using use of an open source MANO for ETSI MEC, OpenBaton. OpenBaton is a an extensible and customizable NFV MANO-compliant framework capable of orchestrating network services across heterogeneous NFV infrastructures [16]. It provides the minimum pre-requisite functionality of the Fog MANO outlined in Section 3.6.

The hardware specifications of all the nodes used in the testbed are outlined in Table 4.2.

Table 4.2: Physical testbed hardware specifications.

Data Centre	Node	RAM	CPU
1	Controller 1	8GB	2 vCPU
1	Compute 1	32GB	8 vCPU
2	Controller 2	8GB	2 vCPU
2	Compute 2	32GB	8 vCPU
-	Fog MANO	8GB	2 vCPU

It is also of interest to investigate the impact that the proposed OVAFF architecture has on the performance of particular applications. A practical data dense smart industrial IoT application for mission critical monitoring is developed and deployed to permit this examination. This application is chosen for several reasons. Firstly, it demonstrates how as IoT deployments grow in size and complexity, the ability to have sensors send all their data to a single, monolithic backend becomes impractical. This is because first, the sheer amount of data begins to become overwhelming for any one sensor. Second, there are few systems that can handle the volume of sensor data.

4.2 Testbed Components

A number of core technologies are necessary for the proposed OVAFF solution to be implemented and verified in a practical environment. These three main technologies from the open source community are OpenStack [46], OpenBaton [47], Zabbix [48] and TICK Stack [49]. These technologies will later be translated and mapped to the components in the design architecture of the OVAFF in Section 4.3 of this Chapter. This section presents an overview of these technologies and provides motivations for their choice.

4.2.1 OpenStack

OpenStack is a cloud operating and management system that allows for the control of large pools of storage, compute and networking resources in a data centre [46].

OpenStack's mission is to provide a flexible solution for both public and private clouds of any size, and for this matter, it considers two basic requirements are considered: clouds must be simple to implement and massively scalable.

The OpenStack platform provides the foundation for the NFV architecture, which essentially makes it a fit-for-purpose cloud for deploying, orchestrating and managing virtual network functions [50]. OpenStack has inbuilt support for multi-site data centres, fault management, upgradeability and deployment solutions. The open, modular and interoperable framework of the OpenStack project offers telecoms and enterprises the ability to design the NFV system of their choosing, without unnecessary components [50]. In this reference implementation of the OVAFF, OpenStack will be used to realize the Fog Node Hosts.

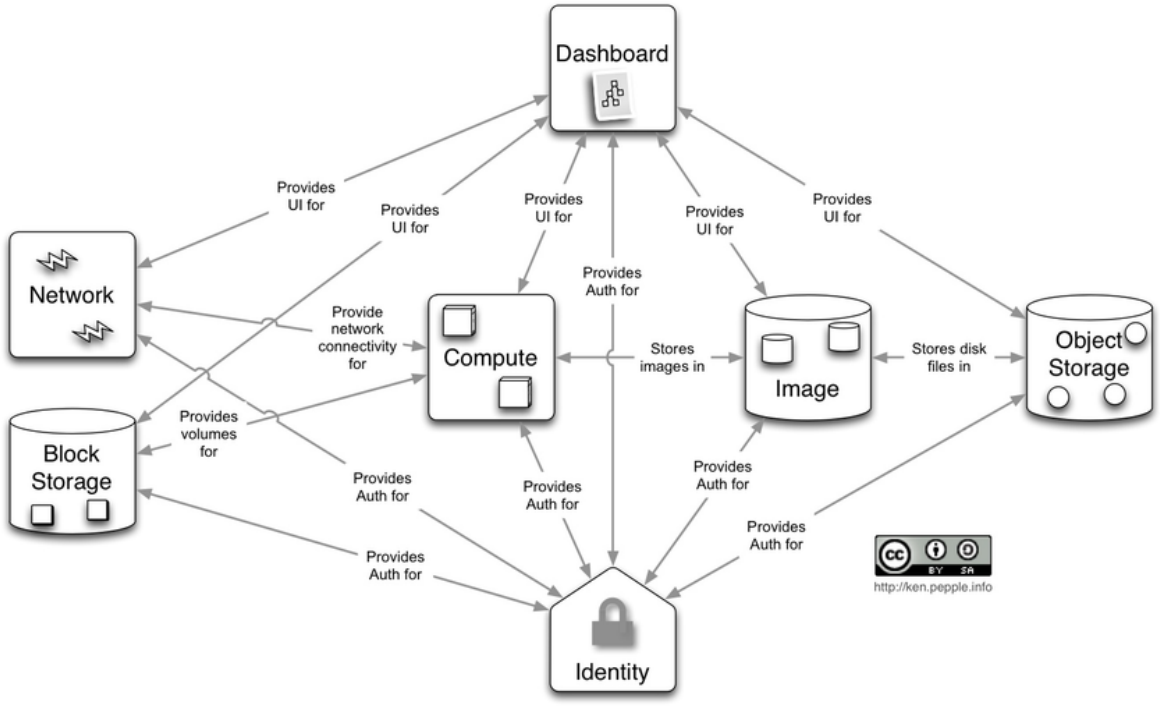


Figure 4.2: OpenStack Folsom Conceptual View [51].

OpenStack is divided into different independent components, named OpenStack services that work together. Their integration is achieved through APIs that are offered and consumed by each service. OpenStack creates a convention by code naming these individual services it comprises. The key services critical for the implementation of Fog Node Hosts are the Compute (Nova), Image (Glance), Identity (Keystone), Telemetry (Ceilometer), DashBoard (Horizon) and Networking (Neutron) services. Figure 4.2 shows a simplified view of the OpenStack architecture and the conceptual relationships between OpenStack services.

The role of each OpenStack service is summarized below:

- **Horizon -** The dashboard service provides end users and administrators a graphical interface to OpenStack's services.
- **Keystone -** The identity service provides authentication and authorization for all OpenStack services, and a catalog of these services of a particular cloud. Keystone also manages a service catalog that allows users and services can locate other services.
- **Nova -** The compute service retrieves images and associated metadata, and transforms user requests on virtual machines and containers as VNFs. Nova interacts with OpenStack Identity for authentication; OpenStack Image service for disk and server images; and OpenStack Dashboard for the user and administrative interface. Each compute node runs a daemon that creates and terminates instances through the hypervisor API.
- **Glance -** The Image service provides a catalog and a repository that enables users to discover, register, and retrieve virtual machine images. It offers a Representational State Transfer (REST) API that enables users to query virtual machine image metadata and retrieve an actual image. A number of periodic processes run on the OpenStack Image service to support caching. Replication services ensure consistency and availability through the cluster. Other periodic processes include auditors, updaters, and reapers.
- **Neutron -** The network service provides virtual networks as a service between devices managed by other OpenStack services, such as virtual machines or containers from Nova. Neutron also allows users to create their own networks and then link them to the devices of their choice.
- **Ceilometer -** The telemetry data collection service provides the following functions: efficient polling of metering data related to OpenStack services, collection of event and metering data by monitoring notifications sent from services and publishing collected data to various targets including data stores and message queues.
- **Cinder -** The block storage service provides persistent storage for hosted VNFs.

The installation of an OpenStack data centre is well documented on the OpenStack website [46] and is an easy process to follow.

4.2.2 OpenBaton

Open Baton is a ETSI NFV compliant MANO framework. Open Baton enables the deployment of Virtual Network Services on top of multiple cloud-infrastructures. Open Baton provides a standard interface for communication between its entities and it simplifies the interoperability with external components and VNFMs through the use of an Advanced Message Queuing Protocol (AMQP) based standard messaging system, RabbitMQ.

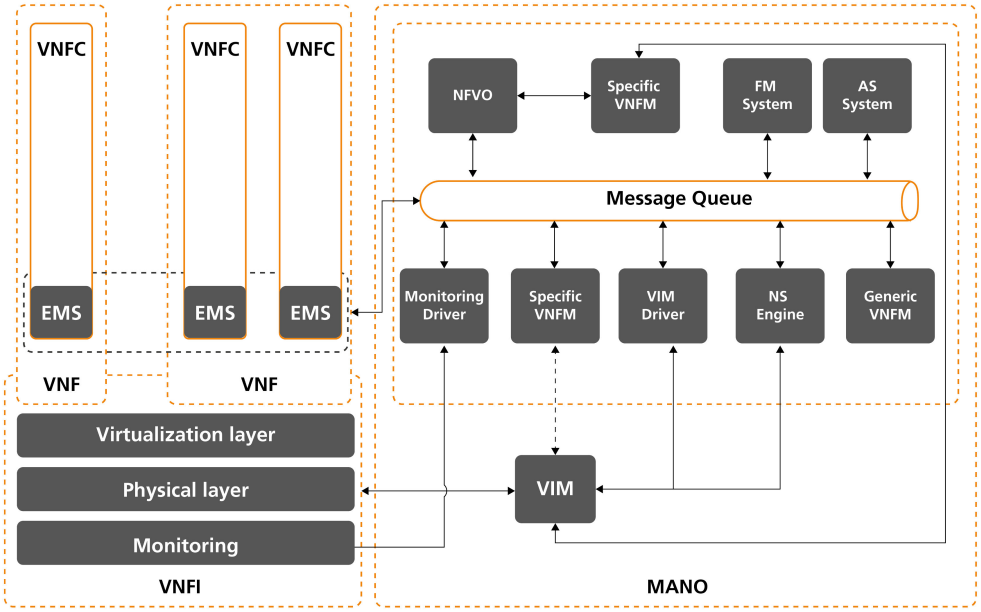


Figure 4.3: OpenBaton Folsom Conceptual View [52].

Figure 4.3 shows an architecture view of the OpenBaton software and the connection interfaces to an NFVI. OpenBaton is made up of the following components [47] [52]:

- A Network Function Virtualization Orchestrator (NFVO) that dynamically orchestrates carrier-grade network functions and services as well as infrastructure resources.
- A generic Virtual Network Function Manager (gVNFM) that dynamically manages the VNFs.
- A plug-in mechanism for adding and removing different types of Virtualized Infrastructure Manager (VIM) without having to re-write anything in the orchestration logic.
- A powerful event engine based on a pub/sub mechanism for the dispatching of life cycle events to registered external modules.

- An auto-scaling engine which can be used for automatic runtime management of the scaling operations of the Virtual Network Functions (VNFs).
- A fault management system which can be used for automatic runtime management of faults at any level.
- A set of libraries for the creation of customized VNFMs.
- A Command Line Interface (CLI), useful for quickly managing network functions from a terminal.
- A user-friendly dashboard through which the platform can be administrated.

4.2.3 Zabbix

Zabbix is an enterprise-class open source distributed monitoring solution that is used to monitor selected resources in a data centre network [48]. It utilises a flexible notification mechanism that enables a fast reaction to detected events and server problems. Zabbix also offers reporting and data visualisation of stored data via a web based frontend with access to all of Zabbix's reports, statistics and configuration parameters.

The key software components of Zabbix are outlined below:

1. **Zabbix agents** are deployed on monitoring targets in order to actively gather information on local resources. They then forward the telemetry data to a central Zabbix server for further processing.
2. **Zabbix Server** is a central process of Zabbix to which agents report information and statistics. The server also monitors, interacts with Zabbix proxies and agents, calculates triggers, sends notifications and maintains a central repository of data.

The testbed network utilises one central Zabbix server and monitoring agents on all nodes.

4.2.4 TICK Stack

The TICK Stack is an open source time series platform that provides services and functionality to accumulate, analyse and act on time series data. The platform is a

collection of products from the developers of the time-series database, InfluxDB [49]. The platform is made up of the following components [49]:

I. **Telegraf -** collects and reports time-series data from a variety of sources. Telegraf can source metrics directly from the system it is running on, pull metrics from third party APIs and it can even to listen for metrics via messaging queue consumer services. It also has output plug-ins to send metrics to a variety of other datastores, services, and message queues, including InfluxDB, Graphite, OpenTSDB, Datadog, Librato, Kafka, MQTT, NSQ, and many others.

II. **InfluxDB -** stores time-series data. InfluxDB is a custom high performance datastore built to handle high write and query loads specifically for timestamped data, IoT sensor data and real-time analytics.

III. **Chronograf -** visualizes and graphs the time-series data. Chronograf is the administrative user interface and visualization engine of the TICK platform. It has templates and libraries that aid users to rapidly build dashboards for real-time data visualizations and creating alerting and automation rules.

IV. **Kapacitor -** provides alerting and detects anomalies in time-series data. Kapacitor is a data processing engine that can process both stream and batch data from InfluxDB. It supports custom logic or user-defined functions for processing alerts with dynamic thresholds, match metrics for patterns, compute statistical anomalies, and perform specific actions based on these alerts.

Each of these components can be used separately, but if used together, they give a scalable, integrated open-source system for processing time-series data. In this dissertation, this platform will be used as a monitoring and alerting system for a complex event processing engine of a smart industrial data dense IoT use case.

Summary

In summary, this section provided an overview of the software tools that were used to make up the testbed hardware and software. Each component was elaborated on, with descriptions on how they can fit into the OVAFF.

4.3 Implementation of Design using Testbed Components

This section describes the actual implementation of the OVAFF design using the testbed components elaborated on in Section 4.2. A demonstration of how each of the constituents in a testbed component maps to a building block in the design architecture will be presented.

4.3.1 Fog Node Host

The Fog Node Host was implemented using the OpenStack Pike release series. Figure 4.4 shows how the individual OpenStack services map to the functional blocks of the Fog Node Host architectural design.

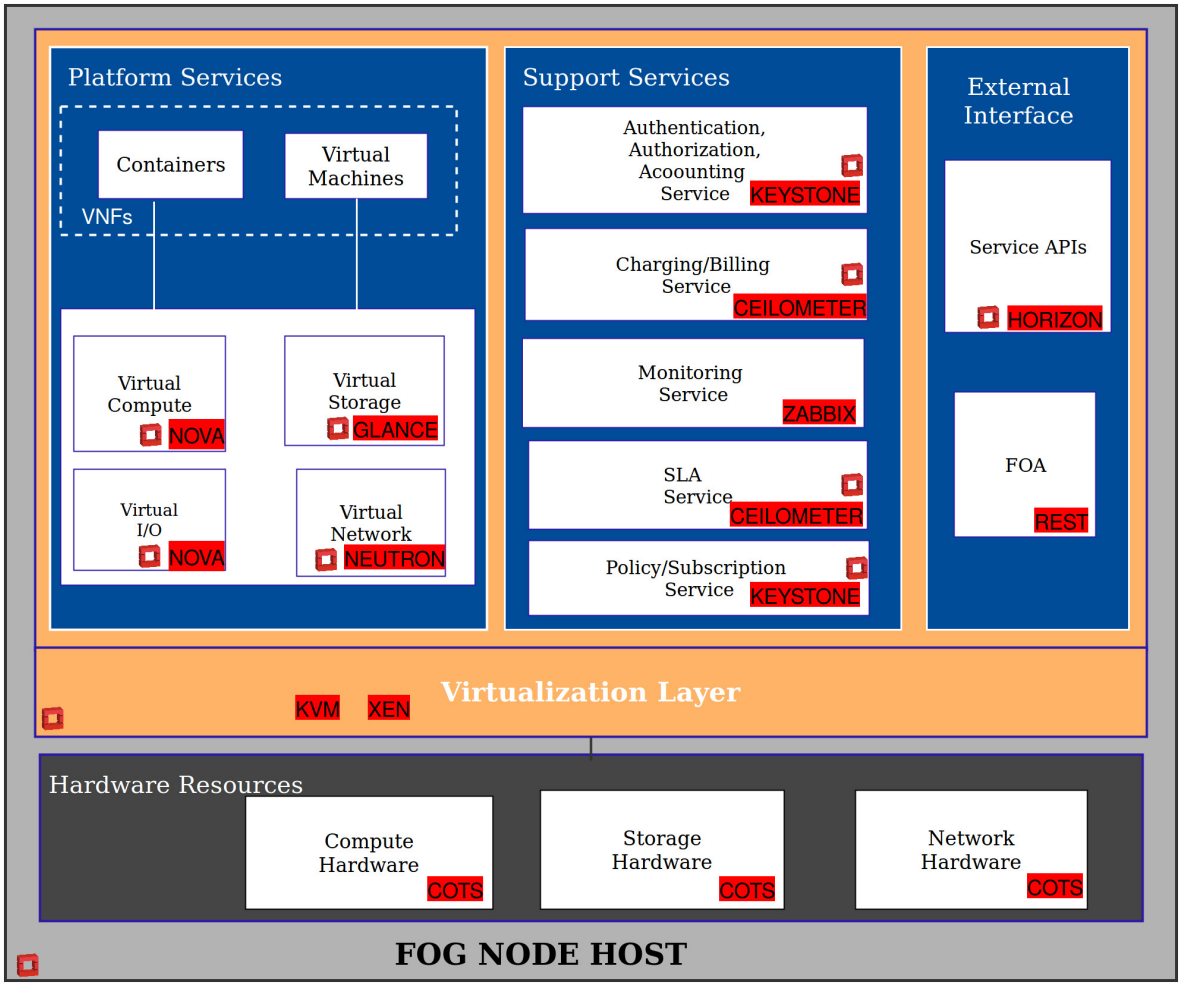


Figure 4.4: Fog Node Host Implementation as OpenStack services.

The hardware layer was implemented on Commercial Off-The-Shelf (COTS) hardware with the specifications listed earlier on in Table 4.2. The virtualization layer used the Kernel-based Virtual Machine (KVM) hypervisor. KVM uses a modified QEMU [53] program to launch its virtual machines thereby making the OVAFF support different virtual disk formats such as raw images, qcow2 and VMware formats. The rest of the building blocks inside the core functional blocks were realized using raw OpenStack services.

4.3.2 Fog MANO

The Fog MANO was implemented using the OpenBaton release version 6.0.0. series. Figure 4.5 shows the practical implementation of the Fog MANO's three main functional blocks; the VIM, VNFM and NFVO using OpenBaton modules. Generic modules were used to simplify the implementation so that it covers a broad range of use-cases.

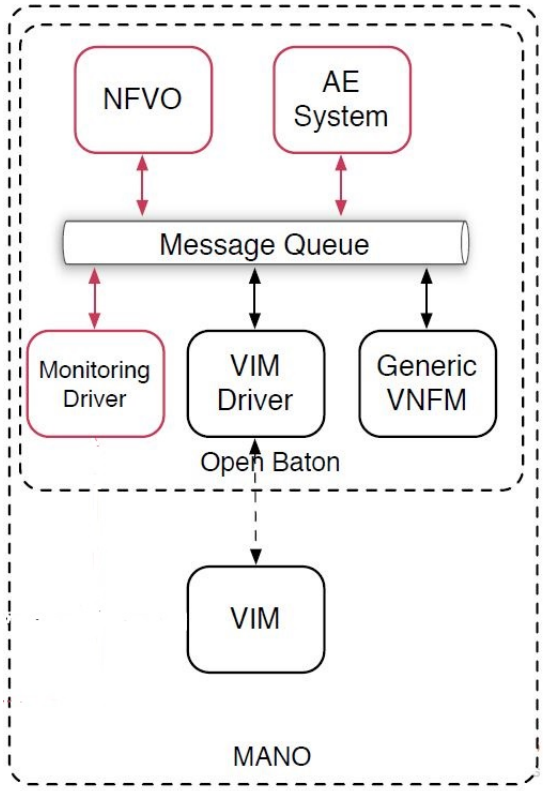


Figure 4.5: Fog MANO implementation in OpenBaton.

The Fog MANO entities are inter-linked via a message queuing communicating channel. Additional services added were the auto-scaling (AE) and fault-management (FM) modules that add robustness to the solution.

4.4 Application Development and Deployment Workflow

This section describes the high level end-to-end application deployment guide for the OVAFF. The guide is limited to new application roll outs and upgrades to running services.

OVAFF Application Development and Deployment Cycle

Figure 4.6 shows a flow diagram of the development and deployment process of an application in the OVAFF ecosystem.

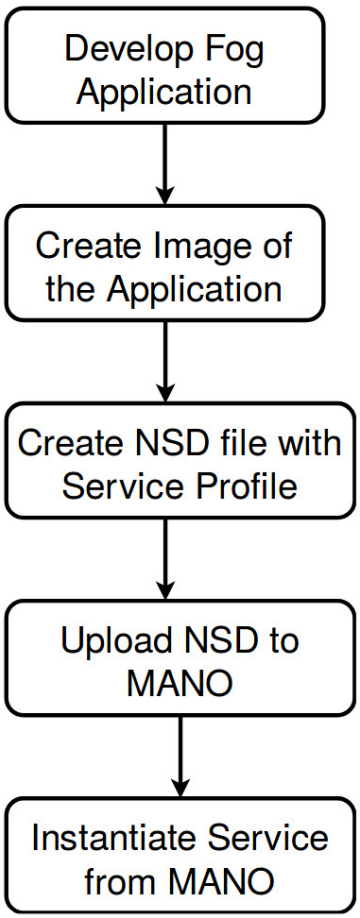


Figure 4.6: Life cycle of Application Development and Deployment.

The cycle begins with the software development process of the fog application. Users develop their applications in any programming language of their choice.

Once the application is production ready, the next step is for the users to create a virtual machine or container image of the application. An image is an executable package that includes everything needed to run the application. This includes the source code, a runtime, libraries, environment variables, configuration files and an OS (for virtual machines). The image is then uploaded to an image repository which can be DockerHub [54] or the Glance service in OpenStack.

Next, the user specifies the service description in an Network Service Descriptor (NSD) file. This file describes attributes of the service such as the application image to use, networks, physical hosts where the service will be deployed, VNF dependencies (in the case of chained or hierarchical applications) and the life cycle rules of the service.

Next, once the NSD file is complete, the user uploads the file to the Fog MANO. The MANO uploads this service profile to a repository where it can retrieved from at service instantiation.

Finally, the service is instantiated from the Fog MANO onto the fog node hosts at the invocation of the user.

Rolling out of Fog Application Upgrades

Figure 4.7 shows a finite state machine diagram of the process of rolling out an upgrade version to a deployed fog application. Two upgrade scenarios are considered in this guide. The first scenario encompasses changes to the source files or image packages. Such updates result in a new image of the application. The other scenario covers a change in the network service descriptor file where the specification of the service profile is updated.

Top roll out an upgrade, the process begins with the user stopping the running network service from the Fog MANO. If the upgrade involves an image upgrade, the user needs to first upload the updated image to the image repository before re-instantiating the network service. Otherwise, if the upgrade involves modifying the service description, the user needs to first upload the modified NSD file to the Fog MANO before re-instantiating the network service.

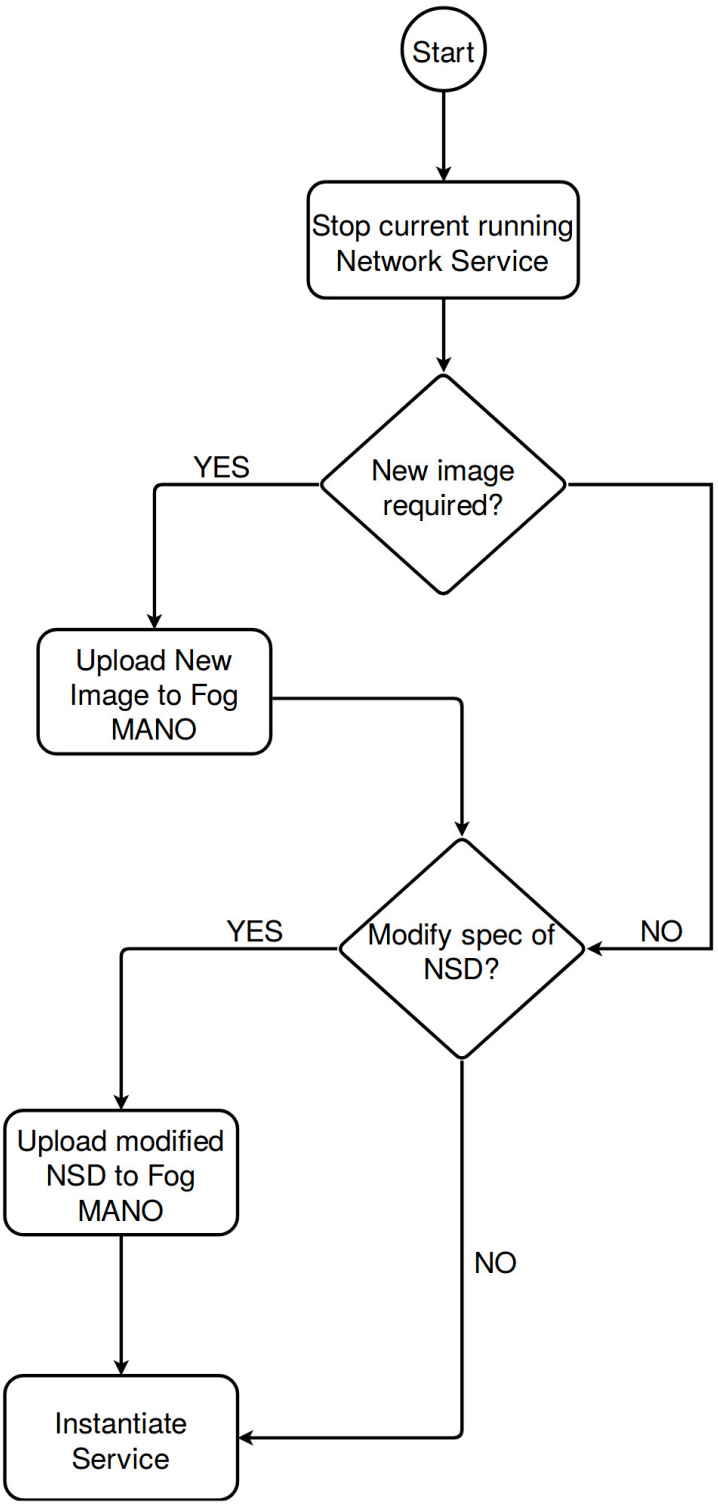


Figure 4.7: Life cycle of Rolling out application upgrades.

4.5 Proof of Concept Industrial IoT Application

To illustrate the impact of an OVAFF implementation in a real world use-case scenario, a physically distributed, low-latency and QoS aware industrial IoT application was developed and deployed for evaluation. This application mocks a typical IoT architecture and provides insights on how the distributed dense data collection, scaling and data aggregation challenges can be handled by the OVAFF.

This section provides a walk through of the scenario description; beginning from the problem definition, then potential conventional solutions and their limitations and, finally the fog solution using the OVAFF.

4.5.1 Problem Definition: Overview of Scenario IoT Architecture

Most IoT architectures involve a few basic components like sensors, a place to collect and store the data, some way to visualize and interact with the data, and often some way for actions to be taken based on events in the data.

The role of sensors is to collect and forward data to a server for storage. That data is then made available for analysis and, based on that, some actions can be taken. However, as IoT deployments grow in size and complexity, the ability to simply have sensors send all their data to a single, monolithic backend become very impractical. This is because, first, the sheer amount of data eventually becomes overwhelming for sensors. Secondly, there are few systems that can handle in real time this sheer volume of streaming sensor data. Thirdly, in real world production systems, systems could scale up. This scaling can be as a result of either taking more variety of readings per sensor, increasing frequency of event collection or the number of deployed sensors.

Description of Experiment

To illustrate these limitations in monolithic cloud backends, experiments were conducted in this dissertation to mock a modest-sized IoT deployment of 1000 sensors deployed across an enterprise. Each sensor takes a series of a reading of temperature, pressure and humidity every second. This translates to 1000 readings/second streaming over the

internet to a server. Each sensor data point is 1kB of data. This translates to 3kB of data per sensor every second or 3000kB of data per second overall.

It is fairly reasonable to assume that most competent backend systems can handle this fairly modest amount of data. But now, if the solution is scaled to something that would actually go into production, the numbers rapidly grow out of control. Sensors that collect 10 readings per second, at 3kB of data per reading, grow to 30kB of data per second per sensor. At 1000 sensors, the solution is streaming 30000kB (30MB) of data per second.

Conventional Solutions To The Scaling Problem

To handle this sheer amount of data, it is common to compress the data before sending it. However, there is compute overhead in doing so and this can affect the performance of the system. Alternatively, the frequency of data collection can be scaled down but this has the effect of tempering with the resolution of the data and the ability of the system to detect and respond to anomalies. Otherwise, the data collection, analysis and response can be pushed out from the monolithic cloud server to a distributed fog computing platform.

The Distributed Fog Solution

For the IoT scenario under review, it is reasonable to segment the deployment of sensors into groups where each group connects to the internet via a gateway. This approach distributes the load of data collection, processing and analysis to the closest point where the data is actually generated.

Each gateway acts as a mini data collection, analysis and response machine that assists with the overall scaling problem. This approach solves the data rate scaling problem however introducing a data distribution problem. Data is now sitting on 1000 different devices and needs to be aggregated somehow.

Given that the data pre-processing and analysis tasks have been offloaded to gateway devices, there is no need to send the highly granular data to the cloud backend. Each gateway downsamples its data before sending it. A rolling 5-minute mean of the sensor data will be periodically computed and then sent to the cloud. In the cloud backend, the

aggregate data from multiple gateway devices will be visualized to get the overall trend analysis.

4.5.2 The Distributed Fog Solution using the OVAFF

This section presents how a practical implementation of the proposed fog solution can be realized on the OVAFF. The description will follow the end-to-end OVAFF application development cycle presented in the previous section.

The data collection, storage, analysis and visualization aspects of the application were realized using the Kapacitor, InfluxDB and Chronograf software components from the TICK Stack [49]. A complete set of these components forms the gateway service which runs in the OVAFF as a VNF. Figure 4.8 shows a high level representation of the deployment's logical architecture.

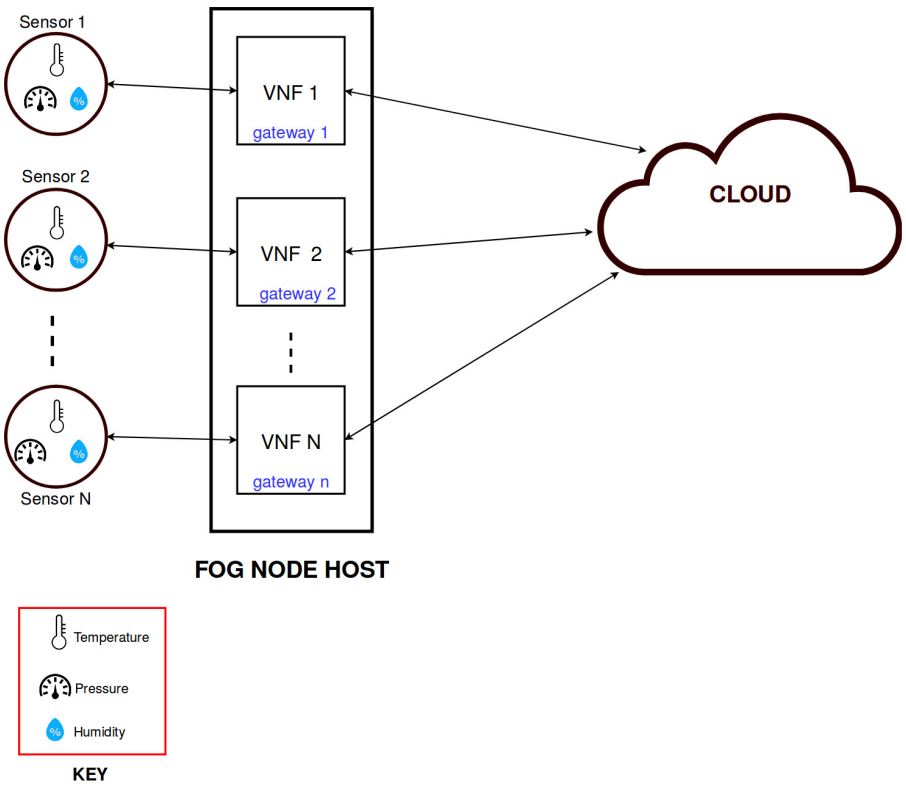


Figure 4.8: Deployment architecture of the proof of concept industrial IoT application on the OVAFF.

The NFV capabilities of the OVAFF enable the gateways to be realized as software appliances thereby guaranteeing agility, flexibility and scalability to each gateway device.

POC Fog Application Development Procedure

During the fog application development phase, the following algorithmic procedures were followed.

Algorithm 1 Create image of fog application

Inputs:
- Incoming sensor data in real-time.

Outputs:
- Dashboards for monitoring data at the gateways.
- Alerts on anomalies in streaming data.
- Downsampled five (5) minute moving average of sensor data periodically sent to cloud.

Development Environment:
- Python 2.7.
- TICK Stack
- Ubuntu 16.04 LTS.
- Gedit Text Editor.

1: Create program to generate emulated data stream of sensor measurements.
2: Install and run a data store of incoming streaming data in time-series database.
3: Install and run the data visualization dashboards for incoming data streams.
4: Create program for performing complex data processing and alerts where anomalies are detected on streaming data.
5: Create and test script for downsampling sensor data by taking a rolling five minute mean of streamed data over a five minute window.
6: Create handle for uploading the downsampled data up to a cloud for long term storage and further analysis.
7: Create a Firebase object relational database for storing the aggregate down sampled sensor data.
8: Upload source files to a source control repository once the application has been tested.

Appendix B.1 contains the source code of the program for step 1 of the algorithm. The process of installing and configuring step 2 and 3 are well documented on the InfluxData website [49]. Appendix B.2 describes the source code for the program that will compute a mean for every 5 minute window of streamed data. Appendix B.2 contains the TICK script for configuring alerts and uploading downsampled data to the cloud endpoint. The process of creating and configuring a Firebase database on the Google Cloud Platform is well documented on the Firebase website [55].

Create Image of Application

The image for the fog application was created at runtime during the instantiation process of the network service. This process is documented in Appendices B.3 and B.4.

Create NSD of Application

A Network Service Descriptor (NSD) is a deployment template which describes a VNF in terms of deployment and operational behaviour requirements. The Virtual Network Function Descriptor (VNFD) also contains connectivity and interface requirements that can be used to establish appropriate Virtual Links between the Virtual Network Function Components of the VNF or from the VNF to another one. The source file of the NSD of the application is contained in Appendix B.4.

4.6 Summary

This chapter has detailed the components necessary to create a fully functional OVAFF testbed. The testbed comprises elements in the Fog Node Hosts, Fog MANO and proof of concept application use case. All developed elements exhibit strict conformance to relevant ETSI and OpenFog Consortium standards. This is critical to ensure that when the testbed is evaluated, the results are an accurate representation of a realistic fog computing deployment.

The framework is a fully open testbed that comprises entirely free and open source software. This exposes the complex concepts to a wide range of audience and helps accelerate technology acceptance. It also provides a convenient point of departure for further research in the field as evaluations are for the most part reproducible in the practical environment and the components can easily and legally be extended to incorporate future experimental technologies.

This testbed implementation chapter demonstrated proof of concept, the next chapter details the extensive evaluations conducted on the proposed concepts.

Chapter 5

Results and Discussion

The previous chapters presented the design architecture and practical implementation of the OVAFF platform. A proof of concept industrial IoT application scenario was described, designed, developed and deployed on the OVAFF to demonstrate the impact of the OVAFF solution on application performance. While the evaluation framework and the practical application use case demonstrates proof of concept, the OVAFF solution still needs to be subjected to realistic evaluations where metrics are used to benchmark the performance and feature matrix of the solution.

In distributed computing systems, the scope of performance testing is very broad. Firstly, this is because of the major differences in testing methodologies employed in physical and virtual environments. Ideally, both environments need to be validated prior to their deployment in the field. Secondly, the very nature of virtualisation introduces many new controls and variables (for example multi-tenancy, acceleration techniques, etc) that impact performance. Appendix A.3 provides a summary of the impacts of virtualization on NFVI testing methods and the variables it adds. For these reasons, the testing plan outlined in the 'ETSI NFV Pre-deployment Testing Guide and Report on Validation of NFV Environments and Services' specification document, the GS NFV-TST 001 [56], will not be followed in the end-to-end performance metric evaluation of the OVAFF. Instead, only an evaluation of specific non-functional aspects of the implementation will be conducted. Moreover, several studies and white papers evaluate the performance of the OpenStack cloud data, control and management planes under the different configurations of virtualization variables. These reports are publicly available on the performance documentation tab on the OpenStack website [46] and in the context of the OVAFF solution, they can be used to provide a general indication of the performance

capabilities of this solution.

The scope of the evaluation metrics presented in this dissertation are thus limited to OVAFF feature verification and an evaluation of their non-functional aspects. In these tests, two Key Performance Indicators (KPIs) are considered; i. Time to Orchestrate and ii. Opportunity Losses. Time to Orchestrate will measure the start time and transient system behaviour when orchestrating different services in different setups, for example, variating images sizes and image location. Opportunity Losses will measure and verify whether, if having all the requirements at a point in time, the orchestration will successfully instantiate. Finally, the industrial IoT prototype application is analysed to provide insight on the impact of the OVAFF solution on real-life application use cases.

5.1 OVAFF Feature Verification

The feature verification tests covered in this section are aimed at verifying the functional integration of the main components in the OVAFF architectural framework, the Fog MANO and the Fog Node Host. The test plan employed for evaluating these features is borrowed from the methodology defined by the ETSI NFV testing working group in TST002 [43] [57]. This test plan aims to verify and validate the end-to-end capabilities and proper behaviour of NFV systems-under-test using integration, functional and usability tests on the system features of the OVAFF advertised in Sections 3.5 and 3.6. The capabilities that will be evaluated include management of descriptors (NSDs and VNFDs) and the life-cycle management of network services and virtual resources.

In the scope of these tests, the systems under test are the following: two Fog Node Hosts providing NFVI and VIM functionality, one Fog MANO solution providing pre-integrated NFVO and VNFM functionality and several VNFs and Network Service (NS) from the prototype IoT application. Test and support VNFs were used to build the reference NSs required to validate the proper behaviour of the systems under test. The reference test and support VNFs used for the tests are contained in Appendix B.4.

Each feature verification test is introduced by a description and pre-conditions for the test. The structure of the test descriptions is outlined in detail at the end of the preamble of this introduction. Then, a sequence diagram showing the process execution flow between the systems under test is presented. Finally, the sequence of the tests and verifications conducted are presented alongside with the verdict of the test. The test

verification steps carried out in the test sequence were both visual and API based. For visual based test results, the expected system behaviour (outlined as the test purpose in the test description) was used to adjudicate the outcome of the test result. For API based tests, a series of REST API calls made during the test sequence were identified in the log messages of the Fog MANO. These REST APIs are standard interfaces defined in the 'NFV Protocols and Data Models; RESTful protocols' specification document [58].

The verdict of each test result was reported in the results as either of the following:

- **OK**: The feature verification check was successful.
- **NOK**: At least one feature verification check failed. A comment was requested.

Test Descriptions

For each test purpose, one or several test descriptions can be specified. Test descriptions compile all the information required to execute a test. They describe all the steps required to achieve a test purpose. The following information is provided in each test description:

I. **Test Purpose:** A concise summary of the test reflecting its purpose. This allows readers to easily distinguish this test from any other tests in the document .

II. **Pre-test conditions (Optional):** Specific conditions that need to be met by the System Under Test (SUT) prior to the start of execution of the test sequence. It can include information about configuration, and/or initial state of the SUT.

III. **References:** List of references to the base specification clauses, use cases or requirements which are either used in the test or define the functionality being tested.

IV. **Test Sequence:** Detailed description of the steps that are to be followed in order to achieve the stated test purpose.

5.1.1 Test Capability of Fog MANO to Attach-To and Orchestrate Fog Node Host Infrastructure

The scope of this test was to check the registration and access rights of the Fog MANO to infrastructure resources of Fog Node Hosts. This test aims to verify the following advertised functionalities of the OVAFF; plug-in design and ability to orchestrate across heterogeneous fog node data centre infrastructures. This test also serves to validate the primary technical precepts of the feasibility of fog business models like an on-demand fog market place and federated infrastructure sharing agreements that have been proposed to be realizable using the OVAFF architecture.

Table 5.1 shows a test description for this verification. Figure 5.1 shows the sequence diagram of the end-to-end process execution sequence amongst the actors in the system.

Table 5.1: Test description for verification of registration of Fog Node Hosts in Fog MANO

Test Purpose	To verify the connectivity of the Fog MANO to Fog Node Hosts.
References	ETSI GS NFV-TST 001 V1.1.1 (2016-04) [56] ETSI Plugtests Report V1.0.0 (2017-03) [43] ETSI Plugtests Test Plan V1.0.0 (2018-02) [59]
Pre-test Conditions	- Running instance of a fully installed and configured Fog MANO. - One or more Point of Presences (PoPs) (i.e. Fog Node Hosts) have been registered in the Fog Mano.

The recipe followed by the Fog MANO to subscribe and register as a tenant of the two Fog Node Hosts in the test bed via the NFVO API is documented in Appendix D.2. Upon successful registration, a record identifying the Fog Node Host is persisted in a database of the Fog MANO. The part of verifying the tenant registrations from the Fog MANO uses a standard NFVO API query that contacts all fog nodes subscribed to. This API query run and payload it returned is documented in Appendix D.3. The returned payload contains the latest virtual Infrastructure-as-a-Service (IaaS) metadata in each the tenant.

From the results of this verification test, the returned payload data shows that the Fog MANO has tenant subscriptions to two Fog Node Hosts identified as InP1 and InP2. The IaaS meta data contains information about the networks, images, keys, image flavours and

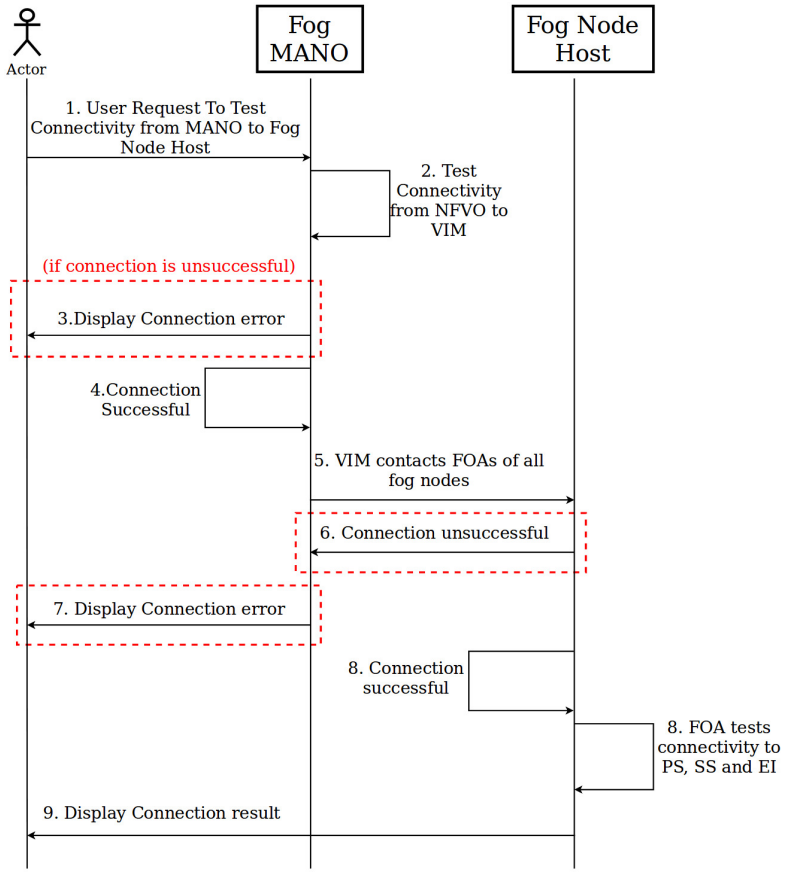


Figure 5.1: Execution plan for verification of registration of Fog Node Hosts in Fog MANO.

security groups in each tenant. The test was repeated to ascertain whether the Fog MANO would update its list of tenant subscriptions if an existing subscription/registration was unilaterally cancelled, expired or terminated by the fog node infrastructure provider. The subscription in InP1 was deleted and the test API query in Appendix D.3 re-run. The payload returned contained only the metadata for the InP2 host. This shows that the subscription function is elastic, which is a critical requirement in the highly dynamic fog market place. Finally, for all verifications carried out in this test, the verdict of the results presented above for the feature verification were all labelled **OK**.

5.1.2 Upload Network Service Descriptor Packages to Fog MANO

The scope of this test was to check the capability of the MANO to upload and manage the network service descriptors and software images of fog applications. This test assumes that the descriptors for the VNFs and NSs have already been developed. The source codes

of the developed packages for the prototype industrial IoT application are contained in Appendix B.3. This test verifies whether the descriptors were uploaded and persisted in the Fog MANO for re-use after the initial upload.

Table 5.2 shows a test description for this verification. Figure 5.2 shows the sequence diagram of the end-to-end process execution sequence amongst the actors in the system.

Table 5.2: Test description for uploading NSDs to Fog MANO

Test Purpose	To verify that a VNF Package can be on-boarded.
References	ETSI GS NFV-TST 001 V1.1.1 (2016-04) [56] ETSI Plugtests Report V1.0.0 (2017-03) [43] ETSI Plugtests Test Plan V1.0.0 (2018-02) [59]
Pre-test Conditions	- VNFD Package resides in a location reachable by MANO. - VNFD Package is complete and consumable by MANO - VNFD format is valid. - VNFD contains all the mandatory elements.

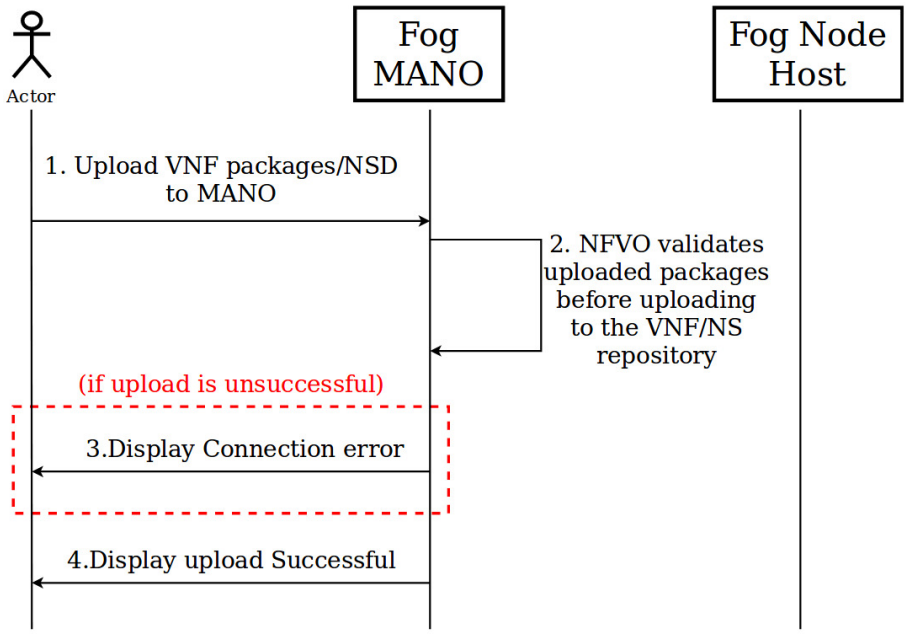


Figure 5.2: Upload Network Service Descriptors to Fog MANO.

The recipe used to upload the NSD of the IoT application to the Fog MANO is documented in Appendix D.4. A series of verification tests were conducted afterwards to ensure the success of the operation as well as the integrity of the NSD package information. To verify that the NSD packages were uploaded, the `NFVO GET /api/v1/ns-descriptors` API query documented in Appendix D.5 was run several times to retrieve all user uploaded NSD packages. The query was also run in separate user sessions to verify the persistence of network service record until otherwise deleted.

The integrity of the NSD package information was verified by comparing the NSD data uploaded in the step documented in Appendix D.4 versus the data returned in the payload in the NFVO API query in Appendix D.5. The contents in the two data packages exactly matched, confirming that the integrity of the NSD package information was maintained. Table 5.3 below summarizes the test execution sequence and the verdict of each verification test carried out.

Table 5.3: Test sequence for uploading Network Service Descriptor to Fog MANO

Step	Description	Result
1	Trigger VNF/NS Package on boarding on NFVO.	
2	Verify that the NSD has been successfully on-boarded in the Fog MANO. `NFVO POST /api/v1/ns-descriptors` API call should return a 201 response.	OK
3	Verify the VNF Package information (e.g. release date, vendor info, manifest, VNFD, SW image meta-data, files contained in the VNF Package) is correct and complete on Fog MANO. NSD data returned in the `NFVO GET /api/v1/ns-descriptors` API query should match data supplied in `NFVO POST /api/v1/ns-descriptors` API call.	OK

5.1.3 Deploy Network Service Across Fog Node Host Infrastructure

The scope of this test was to test the instantiation of a network service from the network service descriptors that were on-boarded in Section 5.1.2. Once an on-boarded NSD is deployed, a Network Service Record (NSR) will be created and virtual machines running the encapsulated fog application are launched. The recipe used to instantiate the network service from the previously uploaded NSD packages is documented in Appendix D.6.

Table 5.4 shows a test description for this verification. Figure 5.3 shows the sequence diagram of the end-to-end process execution sequence across the actors in the system.

When the NSR instantiation process was complete, a VM running the industrial IoT application was launched on Fog Node Host 1, InP1. The VM was allocated a floating address 137.158.126.27 and the TICK services could be accessed by navigating to `http://137.158.126.27:8888` in a web browser. This web page load result is shown in Figure 5.4. Figure 5.5 shows the topology of the InP1 testbed network and the instance launched on the dummy network.

Table 5.4: Test description for instantiating VNFs/NSs on Fog Node Host from MANO

Test Purpose	To verify that an NS can be successfully instantiated.
References	ETSI GS NFV-TST 001 V1.1.1 (2016-04) [56] ETSI Plugtests Report V1.0.0 (2017-03) [43] ETSI Plugtests Test Plan V1.0.0 (2018-02) [59]
Pre-test Conditions	- The VNFD/NSD packages for the application have already been successfully uploaded. - The software image repository (Glance) is reachable by the VIM. - The required resources are available on the Fog Node Host.

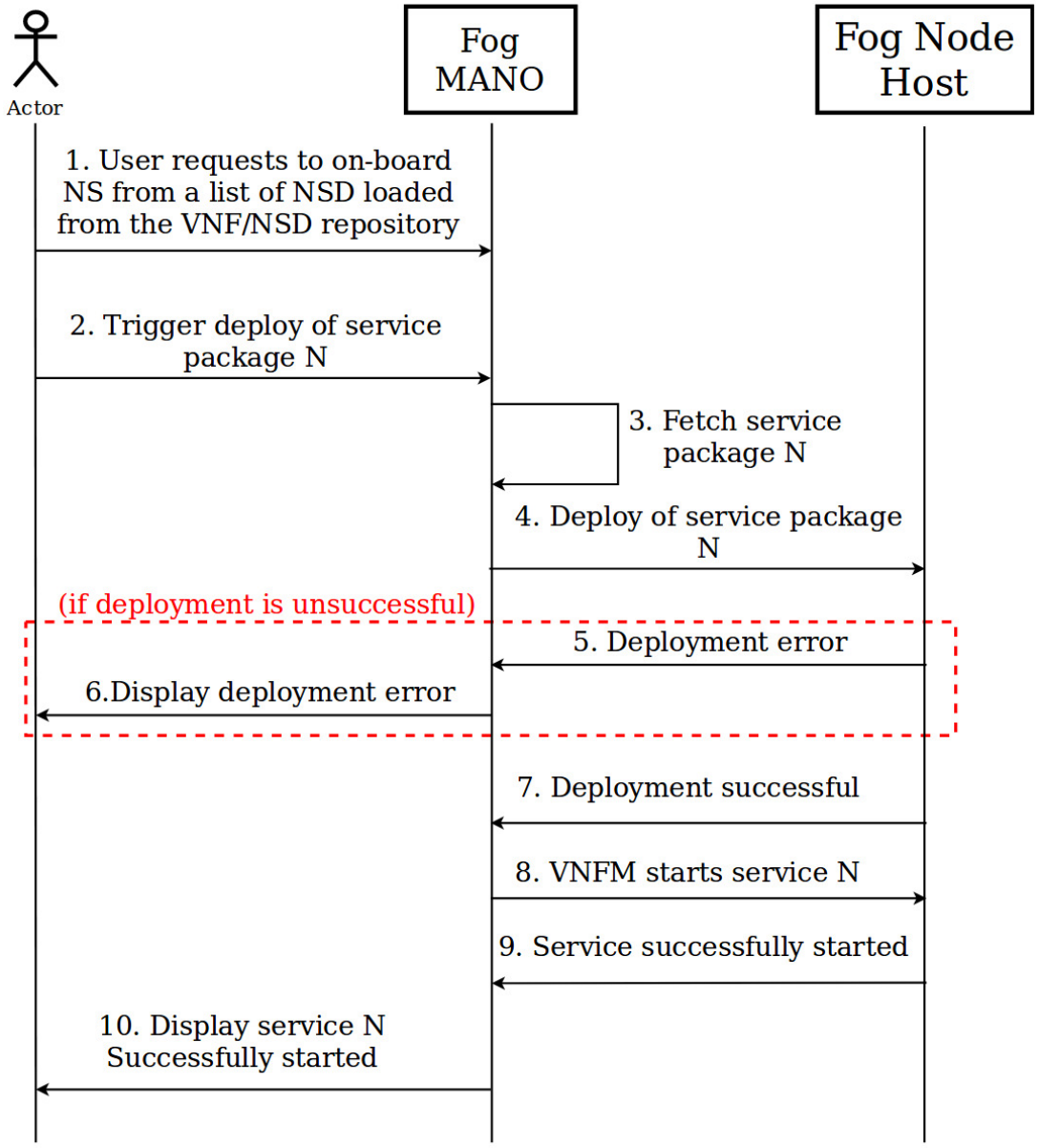


Figure 5.3: Onboard and Instantiate Network Service from Fog MANO

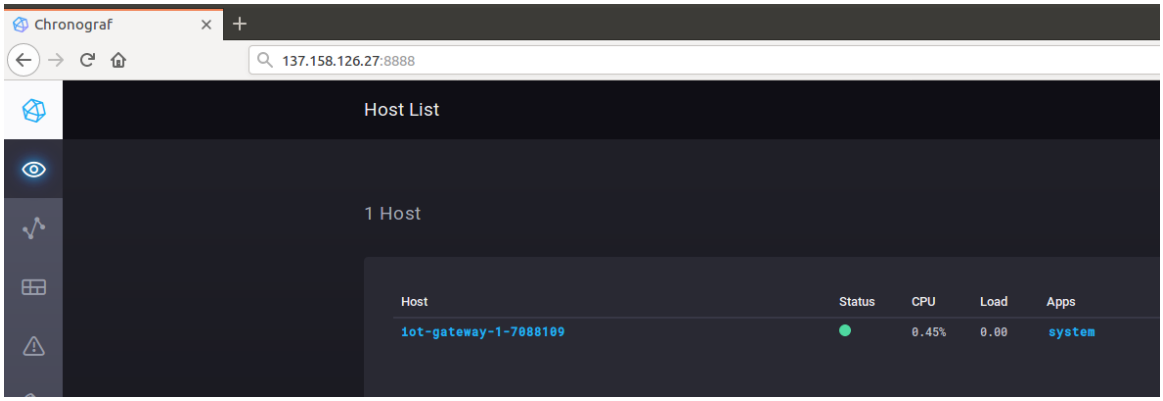


Figure 5.4: Welcome page of instantiated industrial IoT application.

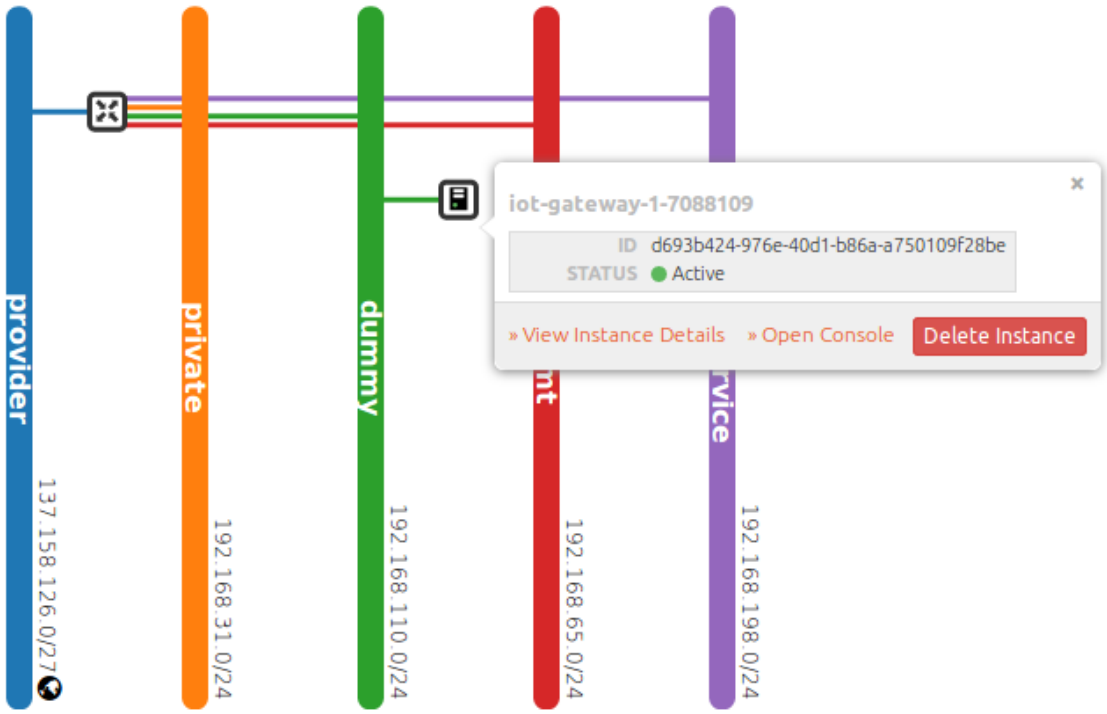


Figure 5.5: Representation of launched instance in the network topology of Fog Node Host 1.

Table 5.5 summarizes the test execution sequence and the verdict each verification step done.

Next, tests were conducted to evaluate the time to orchestrate and opportunity losses metrics. The objectives of these tests are to gather insights into the non-functional aspects in the network service instantiation. These insights will be used in optimizations on the system as well as for benchmarking against other available solutions.

The time to orchestrate metric is divided into two parts; i. Time to deploy a VM of

Table 5.5: Test sequence for instantiating VNFs/NSs on Fog Node Host from MANO

Step	Description	Result
1	User triggers the NS deployment operation in the Fog MANO.	
2	Verify that the requested resources have been allocated by the VIM according to the descriptors	OK
3	Verify that the VNFs have been deployed according to the descriptors	OK
4	Verify that the VNFs are running and reachable through the management network	OK
5	Verify that the VNFs have been configured according to VNFD(s) (i.e by obtaining a result from the management interface)	OK
6	Verify that the NS is successfully instantiated by running the end-to-end functional test	OK

the NSR and ii. Time to instantiate and configure the Virtual Network Function Record (VNFR) of the actual fog application to be deployed. Time to deploy a VM of the NSR basically tells how long it took to boot the VM, mount volumes and pull down image data - which is the point at which the VNFR of the NS starts to initialize themselves. This time to deploy a VM of the NSR is measured as the time difference between the first user API request to instantiate the NS, NFVO POST /api/v1/ns-records/ API request, and the appearance of the NFVO log message 'Finished deploying VMs with external id'. The time to instantiate and configure the VNFR of the actual fog application is measured as the time difference between the first INSTANTIATE_START log message that appears after the NFVO log message 'Finished deploying VMs with external id' and the INSTANTIATE_FINISH log message that appears after the NSR has been successfully launched. The opportunity loss metric will be measured as the number of times the experiment failed and the reasons for each failure.

Due to the random nature of practical implementations and the varying response times in the elements involved, the results were obtained over 15 test runs to ensure an accurate representation. The raw data collected during the test runs is contained in Appendices D.9 and D.10. Three statistical methods, mean, minimum and standard deviation were applied on the results to extract insights from the data.

The mean metric is useful for providing an overall trend of the data set and providing a rapid snapshot of the data. The standard deviation metric is useful for determining the dispersion of data points around the mean. Finally, the minimum metric was used to provide a measure of the best performance that was achieved in the test runs conducted.

The statistically analysed results for test cases i. and ii. in the time to orchestrate metric testing are shown in Table 5.6 and Table 5.7 respectively.

Table 5.6: Time to deploy a VM of the NSR

	s
Minimum	13
Mean	26.3
Standard deviation	7.89

Table 5.7: Time to instantiate and configure the VNFR

	hh:mm:ss
Minimum	00:02:22
Mean	00:06:12
Standard deviation	00:03:44

Discussion

The results show that the time to deploy a VM is significantly lower than the bootstrap time to instantiate and configure the VNFR. The difference is in magnitudes of over tenfold. There are three factors that contribute to this high VNFR bootstrap time; i. network latency to download VNF packages and application dependencies ii. time taken to run the installation of the fog application iii. time to provision and setup inside the instantiated VM some NFV elements of the VNF such as the Element Management System (EMS) and Virtual Network Function Container (VNFC). The conclusion from this is that the Network Service deployment process is a critical path. Since the NFV elements initialize in linear order, the time spent executing one can delay the start-up time of others. A single bad-actor can invoke significant time burdens to start-up time. As such, there's a significant trade-off to be considered with respect to how much work executes in the initialization phase of the Network Service, versus later on, in the parallel phase at runtime.

The standard deviations from the results are about 30% and 50% respectively of the mean values. This indicates that the data collected from the experiments is highly skewed and widely spread around the mean. Such an indication can be easily attributed to an unstable and unpredictable system. However, from the test results, it is clear that the major contributor of the skewed distribution are outlier measurements. When the system has been inactive for a long time, the results suggest that the time to boot up the VM

and VNFR to a healthy state is much higher and this contributes to the outliers. This cold-start behaviour happens all the time. If the orchestration components have not been used in a long time, they are put to sleep to free up resources in the NFVI and Fog MANO elements.

Finally, when the test runs were conducted, 3 out of the 18 deployments of the same NSD failed. The errors had to do with timeouts in the EMS where the fog application's dependencies were not downloaded on time and the system hung. Such a problem can be easily fixed by increasing the EMS timeout. However, this change can result in indefinite NSR boot times. In future work, a package repository for fog applications is proposed on the local OVAFF platform such that launched Network Services can publish their artefacts and packages in this OVAFF repository. The other cause for the failures was the long distance separation between the Fog MANO and OVAFF. A VNFR launched on the OVAFF communicates with the orchestration tools of the Fog MANO via a message queue. If this exchange of messages is conducted over an unreliable network, it was observed that in some cases, the NSR instantiation process would hang or fail prematurely.

5.1.4 Terminate a NS from Fog MANO

The scope of this test was to check the capability of the MANO to terminate a network service and VNFs running on a Fog Node Host. The test network service that was terminated is the one instantiated in Section 5.1.3. The recipe used to terminate the network service is documented in Appendix D.7.

Table 5.8 shows a test description for this verification. Figure 5.6 shows the sequence diagram of the end-to-end process execution sequence across the actors in the system. The Virtual Network Function Manager (VNFM) of the Fog MANO initiates service termination request to the VNFR running in a Fog Node Host.

Table 5.8: Test description for terminating a NS from Fog MANO

Test Purpose	To verify that a VNF running in a NS can be successfully stopped by the Fog MANO.
References	ETSI GS NFV-TST 001 V1.1.1 (2016-04) [56] ETSI Plugtests Report V1.0.0 (2017-03) [43] ETSI Plugtests Test Plan V1.0.0 (2018-02) [59]
Pre-test Conditions	- NS was instantiated. - VNF instance(s) in the NS are running.

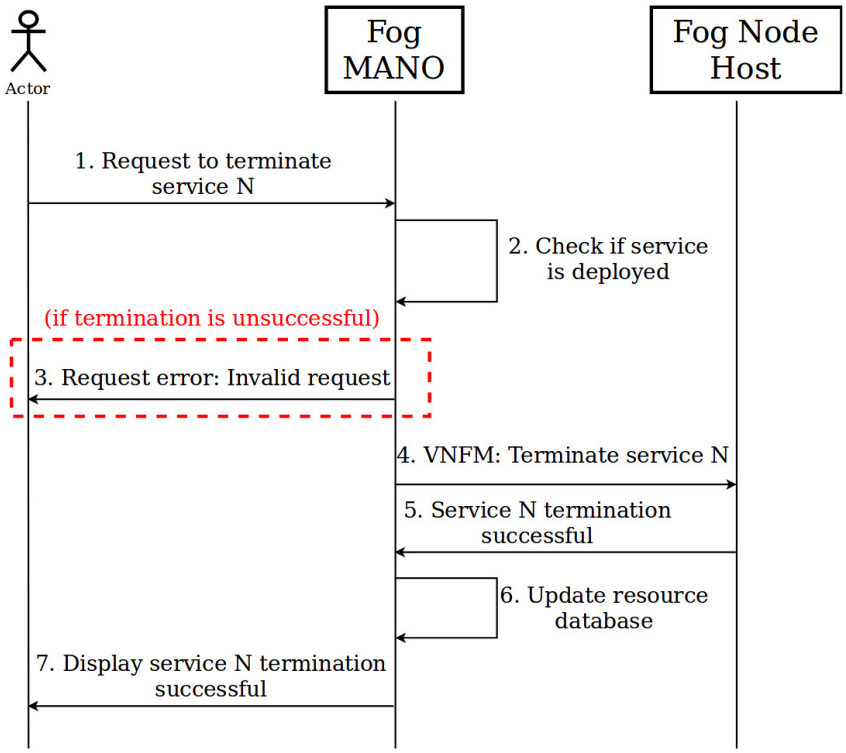


Figure 5.6: Terminate a NS from MANO.

A series of verifications were carried out afterwards to ensure the success of the operation as well as the freeing up of the Fog Node Host resources. To verify that the NSR was deleted, the NFVO GET /api/v1/ns-records/ API query documented in Appendix D.8 was run afterwards to retrieve all active NSRs. The call returned an empty payload, suggesting that the NSR had been deleted. Next, a verification was done on the Fog Node Hosts to ensure that the VMs created for the NS were also deleted at service termination. This was done by navigating to the Instances tab on the OpenStack dashboard screen of the Fog Node Host. The visual check showed that the previously running instance had been terminated. Table 5.9 below summarizes the test execution sequence and the verdict each verification test carried out.

Table 5.9: Test sequence for terminating a NS from Fog MANO

Step	Description	Result
1	User triggers the VNFs stop operation in the Fog MANO.	
2	Verify that the VNFs state inside the NS is "Stopped" on Fog MANO (query or on display)	OK
3	Verify that individual VMs inside the VNFs are stopped on VIM (query, display state from VIM)	OK

5.2 Evaluation of the Prototype Industrial IoT Application

This section presents a practical evaluation of the industrial IoT application that was run as a proof of concept on the OVAFF. The evaluation will be based on a comparison of the legacy cloud native implementation against the optimized OVAFF solution. This evaluation will demonstrate the capabilities of the real-time fog application as well as the performance enhancements introduced by running this service in the fog as opposed to the cloud.

5.2.1 Experimental Setup

In order to determine the performance of the proposed OVAFF architecture in terms of response delay, packet loss and jitter; three scenarios were evaluated depending on where the industrial IoT services described were provided. This resulted in the three architectures depicted in Figure 5.8, Figure 5.9 and Figure 5.10 (for remote AWS cloud, in-country remote data centre and OVAFF-based architecture respectively). The data source of the experiment was physically located in the Communications Research Group (CRG) test bed in Cape Town, South Africa, with remote access to an local server in Johannesburg, South Africa and an AWS cloud data centre in Dublin, Ireland. Johannesburg is 1460km from Cape Town whereas Dublin is 13884km away as shown in geographical representation of the locations in Figure 5.7.

Figure 5.7: Geographical locations of the servers.

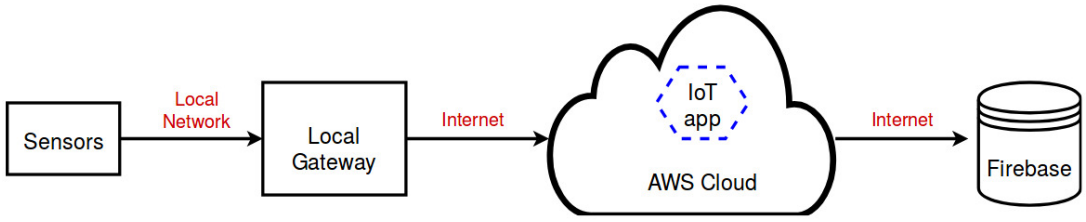


Figure 5.8: Implemented inter-continental cloud architecture.

In the case of Figure 5.8, the different sensors are connected to a local gateway in the CRG testbed that forwards the sensor data to a VM instance running in an AWS cloud situated in Dublin. The local gateway connects to the AWS cloud via the public internet. The VM instance runs the industrial IoT application. At present, there are no AWS cloud data centres in South Africa and Dublin is the closest AWS Point of Presence (PoP) in terms of latency. Therefore, users of AWS cloud data centres from South Africa host their applications in these inter-continental data centres. However, AWS has a dedicated network infrastructure that interconnects inter-continental geographical regions. Therefore, the impact of network latency due to the inter-continental geography is thus greatly alleviated. This scenario is a representation of a typical practical use case of the public cloud in the enterprise at present.

In contrast, in Figure 5.9, industrial IoT application runs in a cloud testbed located in Johannesburg. The data source resides in the CRG testbed. This scenario represents a use case where the cloud data centres are located in country. The long distance separation between the sensors and the data centre is intended to evaluate the impact of the public internet on network latency to the nearest edge location of the cloud data centre.

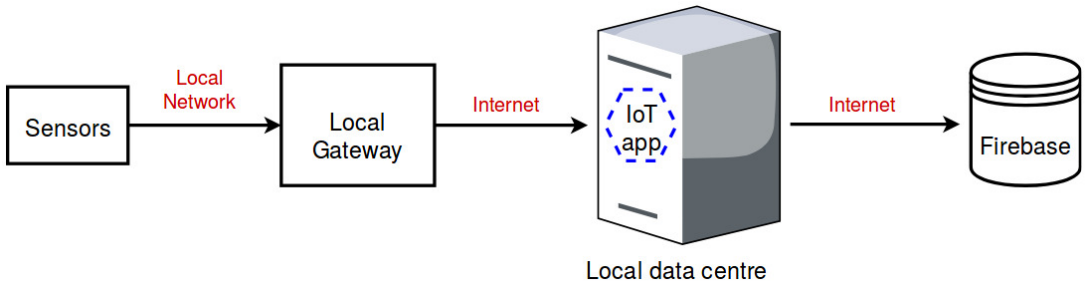


Figure 5.9: Implemented in-country data centre architecture.

In Figure 5.10, an OVAFF data centre in the same local network with the sensors is responsible for executing the mentioned services. All services used in this scenario reside in the CRG testbed. Instead of having a dedicated hardware gateway device as in the other scenarios, this scenario makes use of the solution description in Section 4.5.1 where each gateway device is realised as a virtual resource running in the OVAFF.

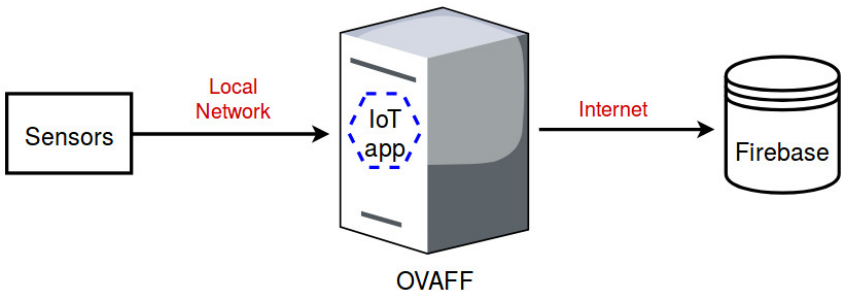


Figure 5.10: Implemented fog computing based architecture.

All network data exchanged in the experiments was sniffed using Wireshark [60] to determine the average data speeds and payloads. The network access ways used were the wireless network (Wi-Fi) and an Ethernet wired network. A Gigabit Ethernet switch and a wireless router using the 802.11n standard were provided. The 802.11n standard is at a maximum speed of 300Mbps and can be used in either the 2.4GHz and 5GHz band. There is more interference on the 2.4GHz than in the 5GHz band This is because the 2.4GHz band is where most wireless equipment like speakers and cordless phones operate. Therefore, the 5GHz band was used for the experiments.

In all three scenarios, the data exchanges between sensors and the gateway were conducted inside the local network of the test environment without needing to reach the internet. In the first scenario, the performance of the inter-continental cloud computing architecture was evaluated, which, for the sake of fairness, makes use of a dedicated network backbone with typical round-trip times outlined in Table 5.10. In the second and third scenarios, the performance of the in-country cloud computing data centre and fog computing model were evaluated, and showed the typical round-trip times outlined in Table 5.10.

Table 5.10: Typical latencies under regular network traffic conditions.

	Minimum (ms)	**Mean (ms)**	**Maximum (ms)**
Scenario 1	75.25	150.3	400.87
Scenario 2	10.25	25.3	60.87
Scenario 3	0.935	1.034	1.695

The application software used in the experiments was configured in the same way across all the test cases and scenarios.

5.2.2 Functional Capabilities of the Prototype IoT Application Under Evaluation

The data structure of the sensor data sent to the real time TICK database looks like this:

```
sensor_fusion temperature=219.871388,pressure=1099.939182,humidity=70.020351
    1541094343
```

There is a measurement, sensor_fusion, which represents a set of points, called the time series. The last value is always the time in Unix epoch time format. The 1541094343 time shown above represents Thursday, 1 November 2018 19:45:43 in human readable format. The data in-between the measurement and time represents the data fields, which are a key value store of the sensor data that was streamed from the sensors. In this evaluation, the fields contain temperature, pressure and humidity readings.

Figure 5.11 shows the results of the top 15 sensor data points queried from the real-time TICK database. This data is a sample space of the 24 hours worth of data points that were streamed from the sensors.

```
tasimba-2@ubuntu:~/generate-weather-data-master$ influx
Connected to http://localhost:8086 version 1.6.3
InfluxDB shell version: 1.6.3
>
> use iot1
Using database iot1
>
> select * from sensor_fusion limit 15
name: sensor_fusion
time                humidity  pressure    temperature
----                --------  --------    -----------
1541094343000000000 70.020351 1099.939182 219.871388
1541094344000000000 69.980735 1099.925367 220.015566
1541094345000000000 70.040064 1100.018253 220.12987
1541094346000000000 70.025416 1100.074092 219.991672
1541094347000000000 69.98027  1099.978669 219.989411
1541094348000000000 69.872455 1099.975423 219.970032
1541094349000000000 69.986152 1099.966317 220.035601
1541094350000000000 69.970415 1100.03543  219.960787
1541094351000000000 70.110503 1099.914668 220.205641
1541094352000000000 70.016589 1100.101058 220.104084
1541094353000000000 70.009404 1099.924093 220.021002
1541094354000000000 69.971175 1100.192363 219.861102
1541094355000000000 70.195486 1100.052992 220.012439
1541094356000000000 70.140947 1099.969408 220.052976
1541094357000000000 70.019178 1100.095101 219.904433
>
```

Figure 5.11: Sample representation of the structure of the streamed sensor data.

Figure C.1 in Appendix C.1 shows a dashboard of real-time visualization charts of the process being monitored. The dashboard view is composed of dashboards for temperature,

pressure and humidity readings.

For the emulated 24 hour IoT process created by the source code in Section B.1 of Appendix B, the temperature, pressure and humidity values collected by the sensors were emulated to fluctuate at certain time intervals. Section C.1 in Appendix C provides an overview of these test configurations and the results observed on the monitoring dashboards for each configured anomaly event.

In all scenarios experimented, 86,400 data points were collected by the sensors in the 24 hour period. For the scenarios shown in Figure 5.9 and Figure 5.10, only 288 data points were forwarded to the cloud data centre by the gateway device due to the down sampling work done at the gateway. As opposed to the cloud native approach in the scenario shown in Figure 5.8, this equates to a 99.667% reduction in data points sent across public networks.

5.2.3 Evaluation of Quality of Service (QoS) Aspects of the Prototype IoT Application Under Evaluation

Quality of Service (QoS) refers to the characteristics of a data link as observed between the connected endpoints. It is measured by metric parameters such as bandwidth, latency, jitter and packet loss rate. To compare the differences in QoS aspects between the fog and legacy cloud-only architectures, Ping [61] command was used to measure latency and the Iperf [62] tool measured the other QoS parameters.

One of the main challenges in industrial IoT deployments is that sensors, actuators and gateways devices should be able to obtain, actuate and display the required information fast enough for seamless operation. This speed limitation is determined by different factors such as the involved hardware, communication technology and scenario under evaluation. During the experiments conducted, most of these variables were fixed thereby reducing bias in the measurements obtained. The only exception was the network traffic since the experiments were carried out on local and Internet-connected networks shared with other users. The influence of the network on the results was alleviated by conducting 10 tests per scenario and averaging the results.

Bandwidth

As shown in Figure 5.12, the 802.11n wireless connection provides in the 66.5Mbps range when connecting across the public internet. When the access speed increases, it can be observed that the throughput of the fog server also increases. However, no matter how much speed is provided, the bandwidth of remote server limits around the 83Mbps rate. With the 1Gbps Ethernet access, the throughput for the Dublin server is 79.1Mbps and that for local data centre is 83.1Mbps. It can therefore be concluded that there is a bottleneck in the core network.

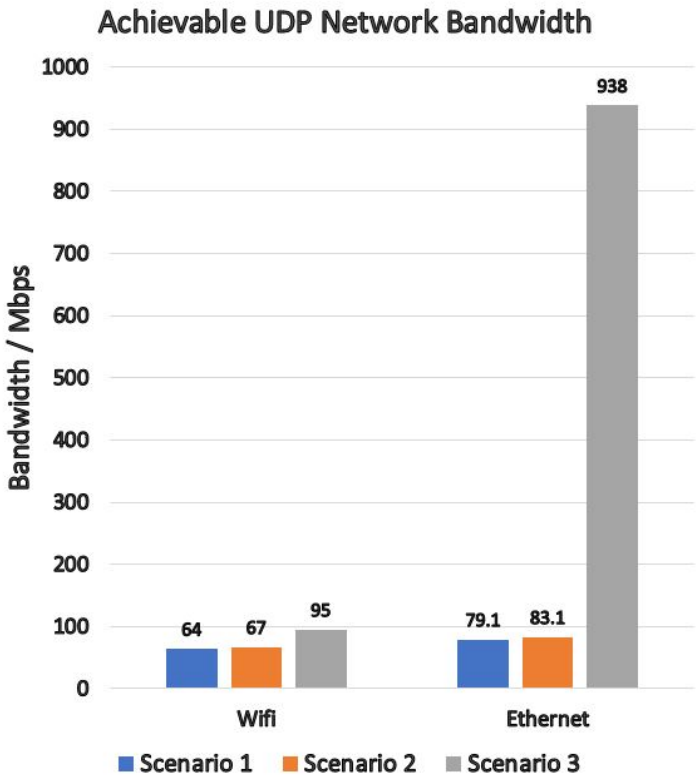


Figure 5.12: Comparison of available bandwidth for data transfers in the three scenarios evaluated.

Next, the available throughput in the cloud-only implementation in scenario 1 and the long distance in-country data centre scenario 2 were compared. Firstly, since the access speed is restricted by the core network in the 802.11n wireless and 1Gbps wired networks, there is only a 24.03% throughput growth with this improvement in access network. Lastly, the immensely increased access speed of the 1Gbps Ethernet network in the fog implementation in scenario 3 results in a bandwidth growth rate that is up-to tenfold.

From these results, it can be concluded that the higher the access speed, the larger the

throughput gap. Edge located applications can immediately acquire local fog computing resources which are isolated from the core network. In daily life, access speeds are generally faster and the introduction of the OVAFF solution can steer the process towards elimination of the major bottlenecks of in core network, which is under the tremendous pressure from greater demand. Applications such as ultra high definition video, augmented and virtual reality and artificial intelligence can make full use of this higher throughput in fog computing.

Latency

Regardless of the network access method used, the latency in the fog computing implementation is conspicuously lower than in the other 2 implementations and the gap can be at least up to 5 times as shown in Figure 5.13. Scenario 1's latency is always longer than scenario 2's. Factually, latency is in proportion to the geographical distances depicted in Figure 5.7. The geographical distance keeps users and remote servers far apart. Therefore, users are closest to the fog computing server, then second to the in-country data centre and third, to the remote AWS cloud. Fog computing can take advantage of ultra-low latency to improve the user experience and reduce congestion as much as possible for the whole network.

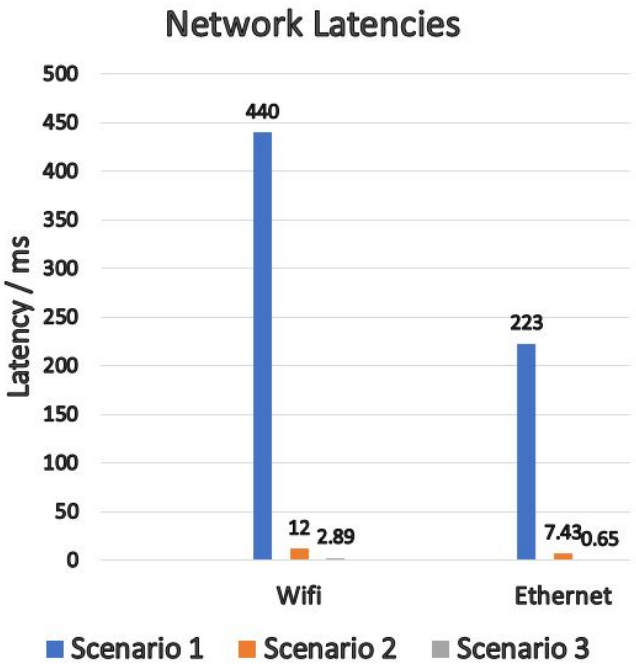


Figure 5.13: Comparison of network latencies for data transfers in the three scenarios evaluated.

Jitter and Packet Loss

Jitter and the packet loss rate in the fog computing server are also lower than those in the remote servers of scenarios 1 and 2, which is important for the User Datagram Protocol (UDP) protocol. As for jitter, the observed growth rate of up to 100% across the different access network technologies. As for packet loss rate, it is nearly 0% for all access technologies in the fog servers. However there are always some packets lost in remote servers. It reflects the congestion in core network and the suggest that there is more congestion and packet loss in process of transferring data to the Johannesburg server than the Dublin server.

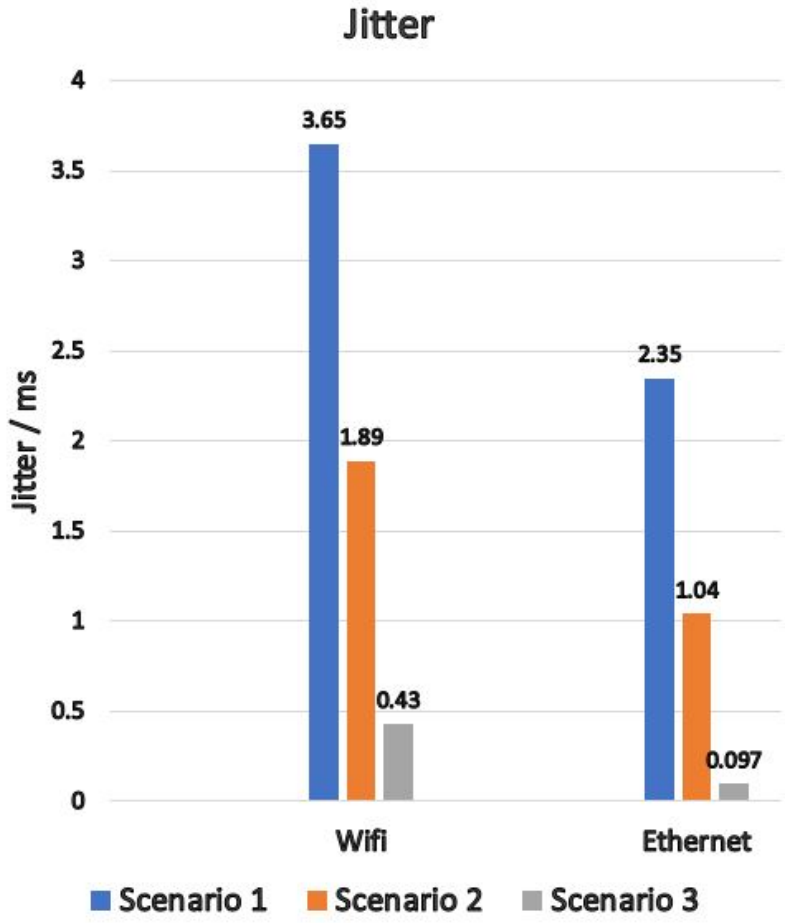


Figure 5.14: Comparison of network jitter in data transfers for the three scenarios evaluated.

Since the industrial IoT application is delivered over HTTP streaming using the Transmission Control Protocol (TCP) protocol to ensure reliable data delivery, the QoS parameters of jitter and packet loss do not directly affect the quality of experience. However, this is inevitable in applications that use the UDP protocol, such as, voice over IP (VoIP) and video streaming.

Discussion on QoS Evaluation of Industrial IoT Application

Based on the analysis above, it can be concluded that compared with wired networks, the characteristics of wireless networks are low throughput, high jitter and high transmission delay. And, when compared with the remote cloud servers, the OVAFF based fog computing implementation guarantees a highly reliable operation of the network and low ratio of packets delayed and discarded. Moreover, it offers other enhancements such as: i) higher reliability and faster response speed ii) it greatly helps ease the burden of network demand and making full use of existing bandwidth; iii) it aids in expanding the scope of business applications, which improves efficiency and drive profits.

5.3 Summary

This chapter described a range of testbed evaluations, demonstrated a proof of concept and identified some critical performance attributes of both the OVAFF platform and prototype application use case. The modular design and implementation allowed each concept to be evaluated separately and also enabled incremental test cases to be incorporated.

The results show that the testbed was implemented successfully and few technical shortcomings were experienced. Moreover, the results demonstrate the viability of the proposed concepts. Most importantly, it has been shown that overheads introduced as a result of the intermittent slow cold-boot of the virtualization and orchestration elements at service instantiation do not affect the end-user experience. There is no current benchmark of acceptable criteria to compare with. To the best of my knowledge, this is the first work to publish results of these overheads in the context of fog computing. Therefore, it can be concluded that this thesis has set a benchmark of the acceptable criteria for future work in this research space. The next chapter concludes the thesis and recommendations for future work are proposed.

Chapter 6

Conclusions and Recommendations

We are still in the early stages of the development of fog computing solutions and, there are still numerous efforts focused on the standardization of these technologies. This is of paramount importance as otherwise, there will exist multiple and not necessarily compatible solutions which could lead to a fragmented marketplace that would fail to grow.

This thesis explored ways to overcome this challenge by presenting an open vendor agnostic fog computing framework for data dense and mission critical application use cases of fog computing. The previous chapters have presented an in-depth analysis of the challenges inherent in the existing fog systems. This thesis presented solutions to this by developing a performance enhancement cooperation framework that is based on the alignment of the efforts of a leading industry consortium and a leading standards setting organization. Firstly, an architecture for infrastructure management and orchestration for fog computing was presented. Then, through the implementation of an initial prototype, the core functional aspects of the proposed architecture were validated. Finally, a proof of concept use case from industry was analysed on the platform to provide insights on the impact of the platform on application performance.

This chapter presents conclusions drawn from the thesis and summarises the contributions that were made. This chapter additionally presents recommended areas for further study that have been identified.

6.1 Conclusions

6.1.1 Need for a Consistent Fog Computing Ecosystem

The first chapter of this thesis discusses the growing digital innovation that will soon disrupt traditional business models. This new era will be characterized by data dense and mission critical application use cases that will need to be handled closer to the data sources for various reasons discussed in the literature. However, this thesis found that at present, there exists a fragmented fog computing ecosystem whose solutions are expensive, proprietary and inflexible. Faced with these pain points, this thesis embarked on developing something new, rooted on a solid foundation of a cooperation framework based on leading efforts from academia, industry and a standards setting body. The solution proposed is based on Network Function Virtualization (NFV) and uses standard IT virtualization technology to run applications on standard servers. This technology is key for consistency and reducing the negative impacts of hardware, feature and platform heterogeneity.

6.1.2 Open Fog Computing Ecosystems Are Broad Subject Matters

Fog computing is relevant across the whole stack from the hardware to the applications. The scope of an open fog ecosystem is therefore extremely large. It is hence crucial that focus is placed on each of the layers, identifying use cases, architectural options, and filling all the gaps in all these layers. For this reason, the Open Vendor Agnostic Fog Framework (OVAFF) solution presented in this thesis pre-dominantly focussed on the Infrastructure-as-a-Service layer. The hardware layer was not explored. Insights and best practices for the other layers of the stack are documented in the OpenFog reference architecture specification document.

6.1.3 Open Source Testbed Implementation

The proposed OVAFF architecture was implemented in a practical testbed to facilitate proof of concept and provide a platform for evaluations. The setup was achieved using Free and Open Source Software (FOSS) for all testbed components.

This enabled a rapid implementation of the proposed architectures in a practical setting and ensured that all subsequent evaluations were reproduced and verified; hence providing a convenient point of departure for future work and innovation. Therefore, the findings show that a comprehensive OVAFF implementation can be achieved through the use of OpenStack cloud software, a mature and field-proven platform which is now in its 16th release and has been deployed in many documented NFV and edge computing scenarios.

6.1.4 Management of Shared Infrastructure

To address the challenge of coordination of resources across heterogeneous fog data centres, this thesis proposed a logically centralized Management and Orchestration (MANO) architecture that can orchestrate these distributed data centres. By incorporating the high-level ETSI NFV architecture, the OVAFF framework emphasised ensuring the isolation of resources to allow for secure separation between multiple fog application providers that utilize the same shared infrastructure.

6.1.5 Performance of Platform and Prototype Application Use-case

The major bottleneck identified in the OVAFF platform was the slow cold-boot when instantiating a network service immediately after the system had been idle for a long period of time. This delay is attributed to the slow wake up of the virtualization and orchestration tools from sleep mode. However, these overheads do not affect the end-user experience and they fall within acceptable criteria when compared to the public clouds. Moreover, in other orchestration systems like Docker Swarm, Kubernetes and OpenStack Heat, the issue of cold-boot is common and well documented. The conclusion is that this bottleneck makes the solution no worse than what is currently on the market. Instead, this thesis is the first work to publish metrics of time to orchestrate and opportunity loss metrics for virtualized fog computing systems.

A proof of concept industry application was deployed on this platform to evaluate the quality of service in comparison to remote servers. The results show that the NFV-based OVAFF can extremely provide up to tenfold throughput and ultra-low latency, jitter and packet loss rate. Additionally, there is more superiority exhibited by the OVAFF for none

performance-based attributes like data reduction, compliance and geographical locality of control.

6.2 Future Work

6.2.1 Evaluation of Other Layers of the Fog Computing Stack

As mentioned earlier, the scope of fog computing is extremely large. It is an entirely new delivery model and infrastructure space. It is comparable to the web and mobile in figurative terms. It is that big in terms of the scope and the OVAFF solution presented in this thesis filled in one of the spaces in that stack. In future work, the application, security and hardware layers of the fog computing stack will need to be explored. Moreover, other Infrastructure-as-a-Service platform services such as providing a Container-as-a-Service multi-tenant platform based on the OVAFF design will need to be evaluated.

6.2.2 OVAFF Performance Enhancement

Fog environments have stringent requirements in the areas of scaling, performance, faster and more deterministic responses to failures, as well as many more other specific features. The requirement for deterministic responses refers to predictable performance, including how to avoid the vagaries of the hypervisor and host Operating System (OS) scheduler affecting performance-sensitive applications. The requirement for performance in the virtualized environment of the OVAFF is largely about high-performance packet processing. This refers to how to get a packet off the network, into a Virtual Machine (VM), processed quickly and back out again on the network. There are several techniques to give VMs direct physical access to the network such as Single-root Input/Output Virtualization (SR-IOV), Data Plane Development Kit (DPDK)-accelerated Open Virtual Switch (vSwitch) for fast packet processing or multi-queue virtual host-users with accelerated virtual switches. These enhancements are outside the scope of this thesis and are crucial in carrier grade virtualization environments. In future work, the effect of these optimization enhancements for robust performance in the virtualized OVAFF environment will be evaluated.

6.2.3 Scalability, High Availability and Resiliency of Deployed Fog Applications

One of the motivations for the OVAFF was its capability to provide 'carrier grade' infrastructure that can ensure high reliability of the platform. The architectural tenet employed in the OVAFF design is to achieve both scalability and reliability using massive horizontal scale. This approach has the effect of pushing the high availability requirement even up to the application level.

However, further evaluations for ability of the OVAFF to support high availability at the application layer needs to be further explored. In particular, the auto-scaling and fault management orchestration services of the Fog MANO were not evaluated, mainly due to their implications of a requirement for an in-built load balancing scheme which is outside the scope of the objectives of this thesis. The auto-scaling service can scale the number of Virtual Network Functions (VNFs) based on a KPI from the monitoring data. The fault management service migrates VNFs across different Fog Node Hosts in the event of alarms received from the Fog MANO's Virtualization Infrastructure Manager (VIM). In addition, other than validating the functional aspects of these two services, an evaluation of their non-functional aspects is also part of the next steps to come especially the KPIs for time to scale and time to migrate deployed Network Services and VNFs.

6.2.4 Enhanced Distributed and Multi-Geography OVAFF Infrastructure

Some vertical markets require Fog Node Hosts to be distributed across multiple geographical locations. Multiple connected OVAFF deployments and high availability amongst them is required for a distributed, multi-geography OVAFF infrastructure. Hybrid approaches and migration strategies to incorporate these requirements need to be further examined. There is therefore room for the OVAFF platform to be further improved in future work to support the following requirements of robust distributed multi-geography virtualization infrastructures:

i. Application-level redundancy across different fog node hosts.

ii. Network management across multiple sites and between physical and virtual infrastructure.

iii. Multisite image replication.

iv. Global and per-site quota management.

6.2.5 Security Evaluation of the OVAFF

Security can be approached from persona and attribute point of views. Under the persona viewpoint, the security parameter encompasses users, network operators, third-party app providers, app developers, content providers, platform vendors or edge devices. Alternatively, the attribute viewpoint encompasses privacy, integrity, trust, attestation, verification and measurement characteristics of the OVAFF platform.

The OVAFF platform environment has to be secure. Both the ETSI MEC and OpenFog reference architectures describe security requirements in detail. The scope of the security attributes of the OVAFF are broad subject matters that will need to be evaluated in dedicated surveys. Future research should investigate the trust and security implications of the architecture presented in this thesis.

6.2.6 Business Models in the OVAFF Ecosystem

Fog computing providers will need to innovate on business models and one of the first items to consider is the revenue model. Revenue could be a percentage of the third-party app revenue, charging for resources consumed or some combination of both. Resource consumption pricing could be flat per use or spot, since resources at the edge are extremely precious and likely to experience huge swings in utilization [2]. Moreover, net neutrality may or may not apply in fog computing since the fog is strictly not speaking to the internet. As a result, Communications Service Providers (CSPs) and Over The Top (OTT) vendors may end up collaborating to a much greater degree than before.

This thesis is a starting point for open and vendor-agnostic fog computing platforms for businesses to leverage on. This platform facilitates the development of favourable market conditions which aim to create sustainable business opportunities for all players in the fog value chain. However, future work should comprehensively evaluate the additional economic factors at stake in the emerging market for fog computing. Moreover, future work should also address the current challenges faced by users trying to extract economic benefits from fog computing or use fog computing to define new business models.

Bibliography

[1] OPFRA001.020817 (2017-02): OpenFog Reference Architecture. [Online]. Available: http://www.openfogconsortium.org/white-paper-reference-architecture/.

[2] SDxCentral, "Innovations in edge computing and mec report," SDxCentral, Industry Report, 2017.

[3] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of mec in the internet of things." vol. 5, pp. 84–91, 10 2016.

[4] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014. [Online]. Available: http://doi.acm.org/10.1145/2677046.2677052

[5] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," vol. 78, 02 2016.

[6] C. C. Byers, "Architectural imperatives for fog computing: Use cases, requirements, and architectural techniques for fog-enabled iot networks," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 14–20, 2017.

[7] B. Varghese, N. Wang, D. S. Nikolopoulos, and R. Buyya, "Feasibility of fog computing," *CoRR*, vol. abs/1701.05451, 2017. [Online]. Available: http://arxiv.org/abs/1701.05451

[8] S. B. Nath, H. Gupta, S. Chakraborty, and S. K. Ghosh, "A survey of fog computing and communication: Current researches and future directions," *CoRR*, vol. abs/1804.04365, 2018. [Online]. Available: http://arxiv.org/abs/1804.04365

[9] OpenFog Architecture Overview. [Online]. Available: http://www.openfogconsortium.org/white-paper-reference-architecture/.

[10] M. B. A. P. Madumal, D. A. S. Atukorale, and T. M. H. A. Usoof, "Adaptive event tree-based hybrid cep computational model for fog computing architecture," in *2016 Sixteenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, Sept 2016, pp. 5–12.

[11] F. Bonomi, R. Milito, J. Zhu, and S. 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*, ser. MCC '12. New York, NY, USA: ACM, 2012, pp. 13–16. [Online]. Available: http://doi.acm.org/10.1145/2342509.2342513

[12] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proceedings of the 2015 Workshop on Mobile Big Data*, ser. Mobidata '15. New York, NY, USA: ACM, 2015, pp. 37–42. [Online]. Available: http://doi.acm.org/10.1145/2757384.2757397

[13] OpenFog Consortium. [Online]. Available: http://www.openfogconsortium.org

[14] Fog computing vs edge computing. [Online]. Available: https://erpinnews.com/fog-computing-vs-edge-computing

[15] N. Mohamed, J. Al-Jaroodi, S. Lazarova-Molnar, I. Jawhar, and S. Mahmoud, "A service-oriented middleware for cloud of things and fog computing supporting smart city applications," in *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, Aug 2017, pp. 1–7.

[16] R. Silva, J. S. Silva, and F. Boavida, "Opportunistic fog computing: Feasibility assessment and architectural proposal," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, May 2017, pp. 510–516.

[17] AWS IoT Greengrass. [Online]. Available: https://aws.amazon.com/greengrass/

[18] Azure Stack. [Online]. Available: https://azure.microsoft.com/en-us/overview/azure-stack/

[19] Securely connect, manage and analyze IoT data with Watson IoT Platform. [Online]. Available: https://www.ibm.com/za-en/internet-of-things/solutions/iot-platform/watson-iot-platform

[20] Network Functions Virtualisation. [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/nfv

[21] A. Aljumah and T. A. Ahanger, "Fog computing and security issues: A review," in *2018 7th International Conference on Computers Communications and Control (ICCCC)*, May 2018, pp. 237–239.

[22] A. Rauf, R. A. Shaikh, and A. Shah, "Security and privacy for iot and fog computing paradigm," in *2018 15th Learning and Technology Conference (L T)*, Feb 2018, pp. 96–101.

[23] ETSI V1 (2018-09-14) Mobile-Edge Computing – Introductory Technical White Paper.

[24] Network Functions Virtualisation – Introductory White Paper. [Online]. Available: https://portal.etsi.org/NFV/NFV_White_Paper.pdf

[25] Network Functions Virtualisation – Update White Paper. [Online]. Available: https://portal.etsi.org/NFV/NFV_White_Paper2.pdf

[26] P. Tsai, H. Hong, A. Cheng, and C. Hsu, "Distributed analytics in fog computing platforms using tensorflow and kubernetes," in *2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Sept 2017, pp. 145–150.

[27] ETSI and OpenFog Consortium collaborate on fog and edge applications. [Online]. Available: https://www.etsi.org/news-events/news/1216-2017-09-news-etsi-and-openfog-consortium-collaborate-on-fog-and-edge-applications

[28] Open Edge Computing. [Online]. Available: http://openedgecomputing.org

[29] Open Edge Computing. [Online]. Available: https://github.com/openedgecomputing

[30] ETSI Multi-access Edge Computing group releases a first package of APIs. [Online]. Available: https://www.etsi.org/news-events/news/1204-2017-07-news-etsi-multi-access-edge-computing-group-releases-a-first-package-of-apis

[31] ETSI GS MEC 009 V1.1.1 (2017-07): "Mobile Edge Computing (MEC); General principles for Mobile Edge Service APIs".

[32] ETSI GS MEC 010-2 V1.1.1 (2017-07): "Mobile Edge Computing (MEC); Mobile Edge Management; Part 2: Application lifecycle, rules and requirements management ".

[33] ETSI GS MEC 011 V1.1.1 (2017-07): "Mobile Edge Computing(MEC); Mobile Edge Platform Application Enablement ".

[34] ETSI GS MEC 012 V1.1.1 (2017-07): "Mobile Edge Computing (MEC); Radio Network Information API".

[35] ETSI GS MEC 013 V1.1.1 (2017-07): "Mobile Edge Computing (MEC); Location API".

[36] ETSI GS MEC 003 V1.1.1 (2016-03): "Mobile Edge Computing; Framework and Reference Architecture".

[37] Y. Jiang, Z. Huang, and D. H. K. Tsang, "Challenges and solutions in fog computing orchestration," *IEEE Network*, vol. 32, no. 3, pp. 122–129, May 2018.

[38] Accelerating NFV Delivery with OpenStack. [Online]. Available: https://www.openstack.org/assets/telecoms-and-nfv/OpenStack-Foundation-NFV-Report.pdf

[39] X. Sun and N. Ansari, "Avaptive avatar handoff in the cloudlet network," vol. PP, pp. 1–1, 05 2017.

[40] K. Li and J. Nabrzyski, "Virtual machine placement in cloudlet mesh," *Journal of Communications and Networks*, vol. 20, no. 3, pp. 266–278, June 2018.

[41] T. M. Fernández-Caramés, P. Fraga-Lamas, M. Suárez-Albela, and M. Vilar-Montesinos, "A fog computing and cloudlet based augmented reality system for the industry 4.0 shipyard," *Sensors (Basel)*, vol. 18, no. 6, p. 1798, Jun 2018, 29865266[pmid]. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/29865266

[42] M. S. de Brito, S. Hoque, T. Magedanz, R. Steinke, A. Willner, D. Nehls, O. Keils, and F. Schreiner, "A service orchestration architecture for fog-enabled infrastructures," in *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, May 2017, pp. 127–132.

[43] ETSI Plugtests Report V1.0.0 (2017-03): "PLUGTESTS INTEROP EVENTS".

[44] Riverbed Modeler: A Suite of Protocols and Technologies with a Sophisticated Development Environment. [Online]. Available: https://www.riverbed.com/sg/products/steelcentral/steelcentral-riverbed-modeler.html

[45] R. M. Yakov Rekhter and D. Karrenberg, "Address Allocation for Private Internets," Internet Requests for Comments, RFC Editor, RFC 1918, February 1996. [Online]. Available: https://tools.ietf.org/html/rfc1918

[46] OpenStack. [Online]. Available: https://www.openstack.org/

[47] Open Baton. [Online]. Available: https://openbaton.github.io/index.html

[48] Zabbix. [Online]. Available: https://www.zabbix.com/

[49] InfluxData - Open Source Time Series Platform. [Online]. Available: https://www.influxdata.com/time-series-platform/

[50] O. Foundation, "Accelerating nfv delivery with openstack - global telecoms align around open source networking future," OpenStack Foundation, Industry Report, 2016.

[51] A quick overview of OpenStack technology. [Online]. Available: https://www.ibm.com/blogs/cloud-computing/2014/08/06/quick-overview-openstack-technology/

[52] Open Baton: A Framework for Virtual Network Function Management and Orchestration for Emerging Software-Based 5G Networks. [Online]. Available: https://sdn.ieee.org/newsletter/july-2016/open-baton

[53] QEMU. the FAST! processor emulator. [Online]. Available: https://www.qemu.org/

[54] Docker - Build, Ship, and Run Any App, Anywhere. [Online]. Available: https://www.docker.com/

[55] Firebase Realtime Database. [Online]. Available: https://firebase.google.com/docs/database/

[56] ETSI GS NFV-TST 001 V1.1.1 (2016-04): "Network Functions Virtualisation (NFV); Pre-deployment Testing; Report on Validation of NFV Environments and Services".

[57] ETSI GS NFV-TST 002 V1.1.1 (2016-10): "Network Functions Virtualisation (NFV); Testing Methodology; Report on NFV Interoperability Testing Methodology".

[58] ETSI GS NFV-SOL 005 V2.4.1 (2018-02): "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point".

[59] ETSI Plugtests Test Plan V1.0.0 (2018-02): "PLUGTESTS INTEROP EVENTS - 2nd ETSI NFV Plugtests".

[60] Wireshack - Go Deep. [Online]. Available: https://www.wireshark.org/

[61] N. W. Group, "INTERNET CONTROL MESSAGE PROTOCO," Internet Requests for Comments, RFC Editor, RFC 792, September 1981. [Online]. Available: https://tools.ietf.org/html/rfc792

[62] iPerf - The ultimate speed test tool for TCP, UDP and SCTP. [Online]. Available: https://iperf.fr/

[63] Y. Simmhan, "Big data and fog computing," *CoRR*, vol. abs/1712.09552, 2017. [Online]. Available: http://arxiv.org/abs/1712.09552

Appendix A

Additional Related Content and Schematics

A.1 Interaction Models in Fog Computing

A defining characteristic of fog computing is its smaller network distance from the edge, but multiple network topology architectures exist, as shown in Figure A.1.

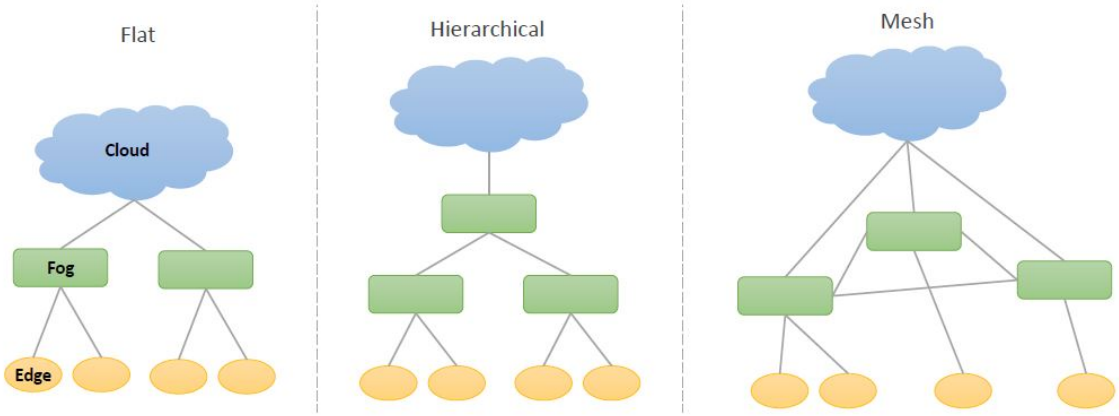


Figure A.1: Different interaction models in fog computing [63].

Some models consider there to be a flat fog layer that sits, say, at one hop from the edge devices that allows connectivity to the cloud for a zone of responsibility based on logical groupings or spatial regions [63]. Others consider a hierarchical approach where fog devices form layers, where each layer is further away from the edge device and responsible for fog devices that fall within its sub-tree, with the cloud forming the root

[63]. Alternatively, there are mesh designs where fog resources can communicate with each other as well as the cloud, with edge devices assigned to distinct fog nodes [63].

Most fog deployments require computational and system hierarchy to leverage the benefits offered by fog computing. Fog computing resources can be seen as a logical hierarchy based on the end-to-end functional requirements of a fog system [36]. Figure A.2 presents the logical view of the fog cluster hierarchy for OVAFF deployments.

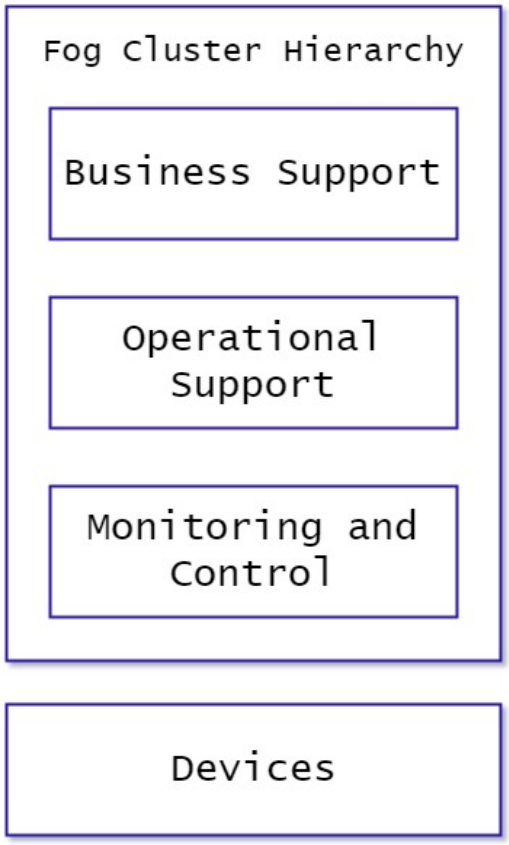


Figure A.2: Layered architecture view of the fog cluster hierarchy.

This dissertation references from OpenFog, five hierarchical service layers that can address specific concerns of a fog system in the fog cluster hierarchy. It should be noted that these layers are mutually exclusive and should be applied on a case by case basis. These service layers are explained in-depth below.

Devices

These are the physical sensors and actuator devices that produce the telemetry data to be consumed by the monitoring and control layer.

Monitoring and Control

The main responsibility of this layer is to execute control logic through stateful inspection of the sensor telemetry. This can involve computing alarms and generating events which

may trigger workflows through machine-to-machine or human intervention.

Operational Support

This layer provides a composite picture of the real-time operations with some short-term history. It is responsible for analysing streaming telemetry data and storing operationally oriented analytics. These analytics may be presented through interfaces like control room dashboards and mobile applications. At this layer, the scope of the analytics is narrow - it mainly focuses on the operational aspects of the physical environment for which the system is responsible.

Business Support

The primary responsibility of this layer is to store and analyse the entire history of IoT operations that span multiple systems. This is the system of record for IoT operations as governed by the compliance and record retention policies. Using machine learning and artificial intelligence models on this data, these Petabyte scale analytics will help in mining insights, business planning, operational efficiency and optimization of processes. Additionally, other secondary aspects of this layer include meta-data and reference data management, business rule and operational health management of lower layers.

Enterprise Systems

These are the ERP, CRM and other enterprise systems for an organization using fog deployments.

A.2 Fog Deployment Models

The size of the fog cluster hierarchy is flexible. It can range from a single small fog node to large fog systems depending on the needs of the application use case. Fog envisions a seamless a seamless cloud-fog-thing architecture to enable computing anywhere along the cloud to thing continuum.

Figure A.3 shows a subset of the combination of fog and cloud deployments to address the various domain scenarios framed in the layered view of fog systems outlined earlier in Section A.1.

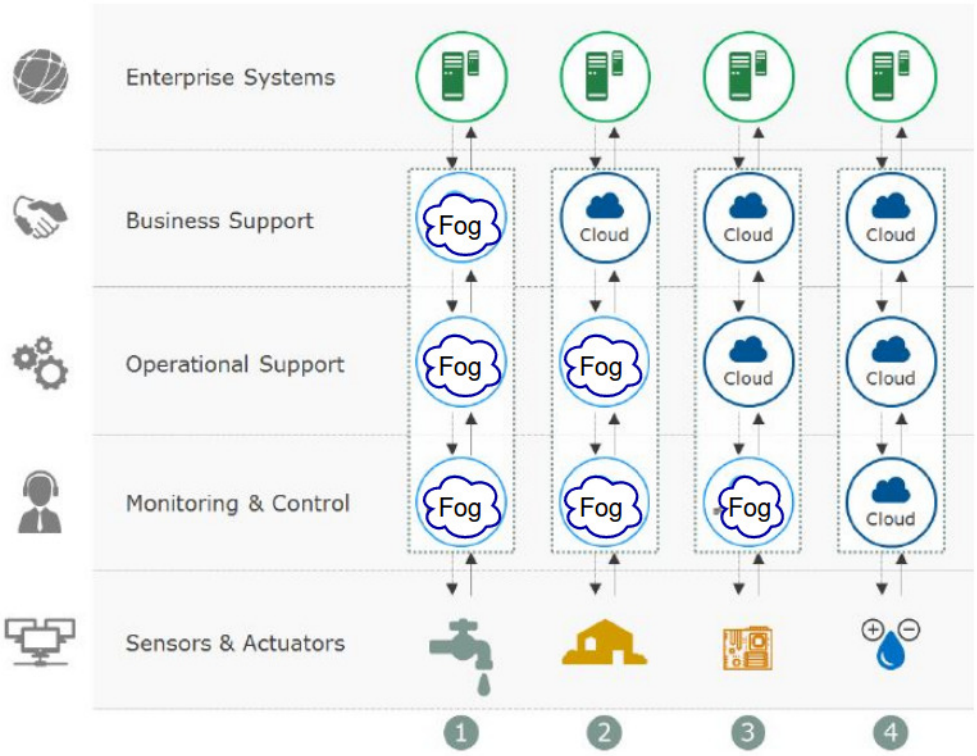


Figure A.3: Fog deployment combinations for various domain scenarios [1]

Each fog layer may represent a hierarchy of fog clusters fulfilling the same functional responsibilities [1]. Depending on the scenario, multiple fog and cloud elements may collapse into a single physical deployment [1]. Moreover, each fog element may also represent a mesh of peer fog nodes where fog nodes may securely discover and communicate with each other for exchanging context-specific intelligence [1]. The enterprise support systems integrate with the cloud for business operations.

The rest of this section outlines the four scenario-based deployment models for the OVAFF. Use case scenarios falling under each deployment model are also outlined.

A.2.1 Model 1: Cloud Independent Model

Figure A.4 represents a fog deployment hierarchy that is independent of the cloud. This model is applicable for use cases where the cloud can't be used for reasons such as regulatory compliance, unavailability of a central cloud in a particular geography, low event to action time window and military grade security and privacy.

Example use cases include ATM banking systems, armed forces IT systems, drone operations and some health care systems.

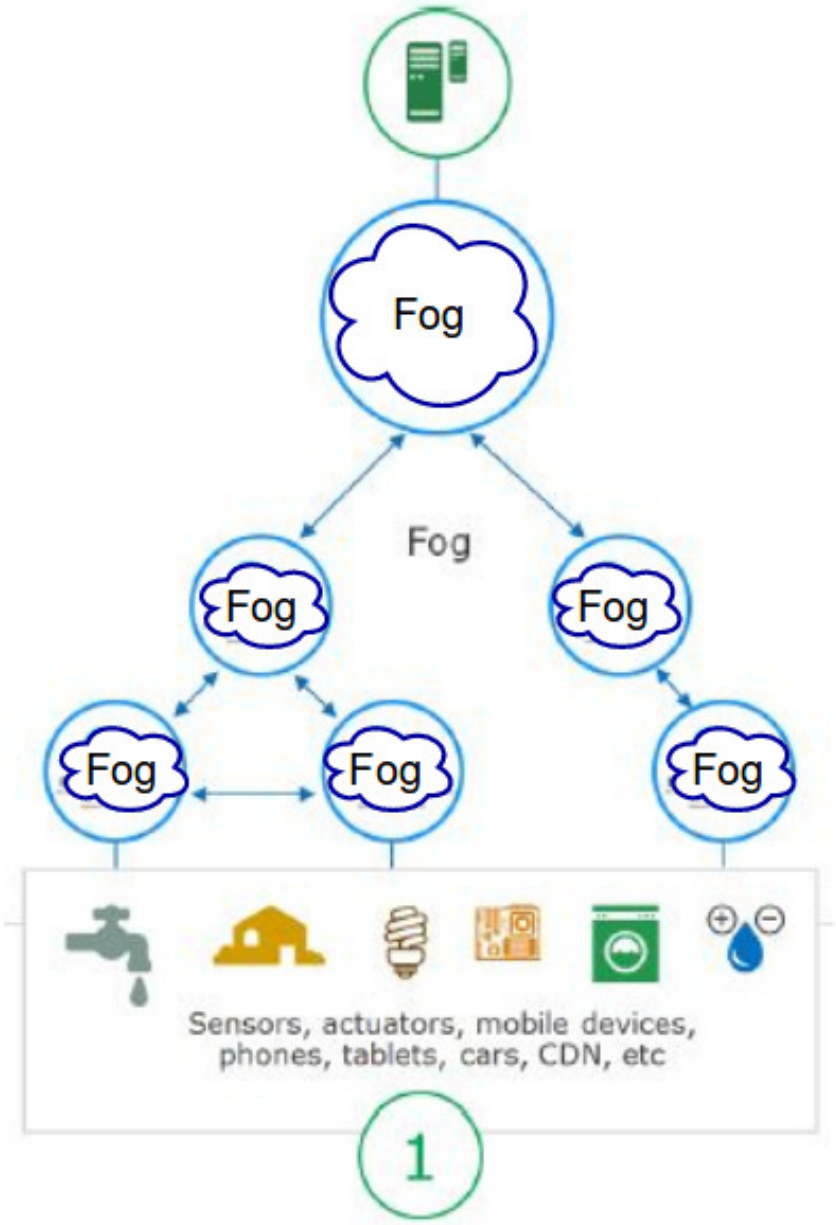


Figure A.4: Cloud independent fog deployment model [1]

A.2.2 Model 2: Time Independent Model

Figure A.5 represents a fog deployment hierarchy with cloud and fog nodes in hierarchy. This model is applicable for use cases where the cloud is used for information processing related to decision making that may have event-to-action time window ranging from hours to days to months. Operation-centric information processing is done by fog nodes that are located to the infrastructure or process being managed. Example use cases include retail and surveillance systems.

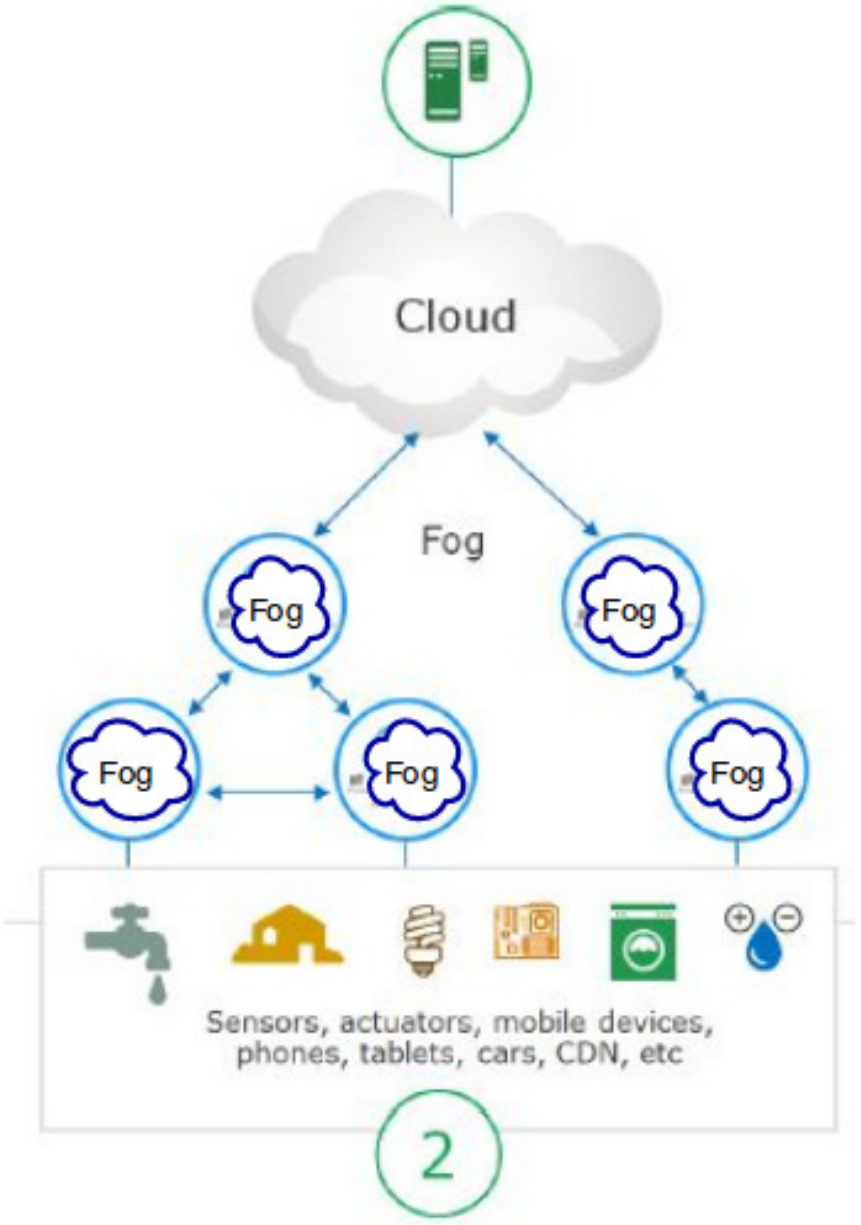


Figure A.5: Fog deployment model for delay tolerant and time independent applications [1]

A.2.3 Model 3: Time Sensitive Model

Figure A.6 represents a fog deployment hierarchy for time-sensitive use cases. Local fog infrastructure is used for time-sensitive computation whereas the cloud is used for balancing operational and business-related information processing.

Example use cases include mobile network acceleration, Content Delivery Network (CDN)s for internet acceleration and uninterruptible power supply device monitoring systems.

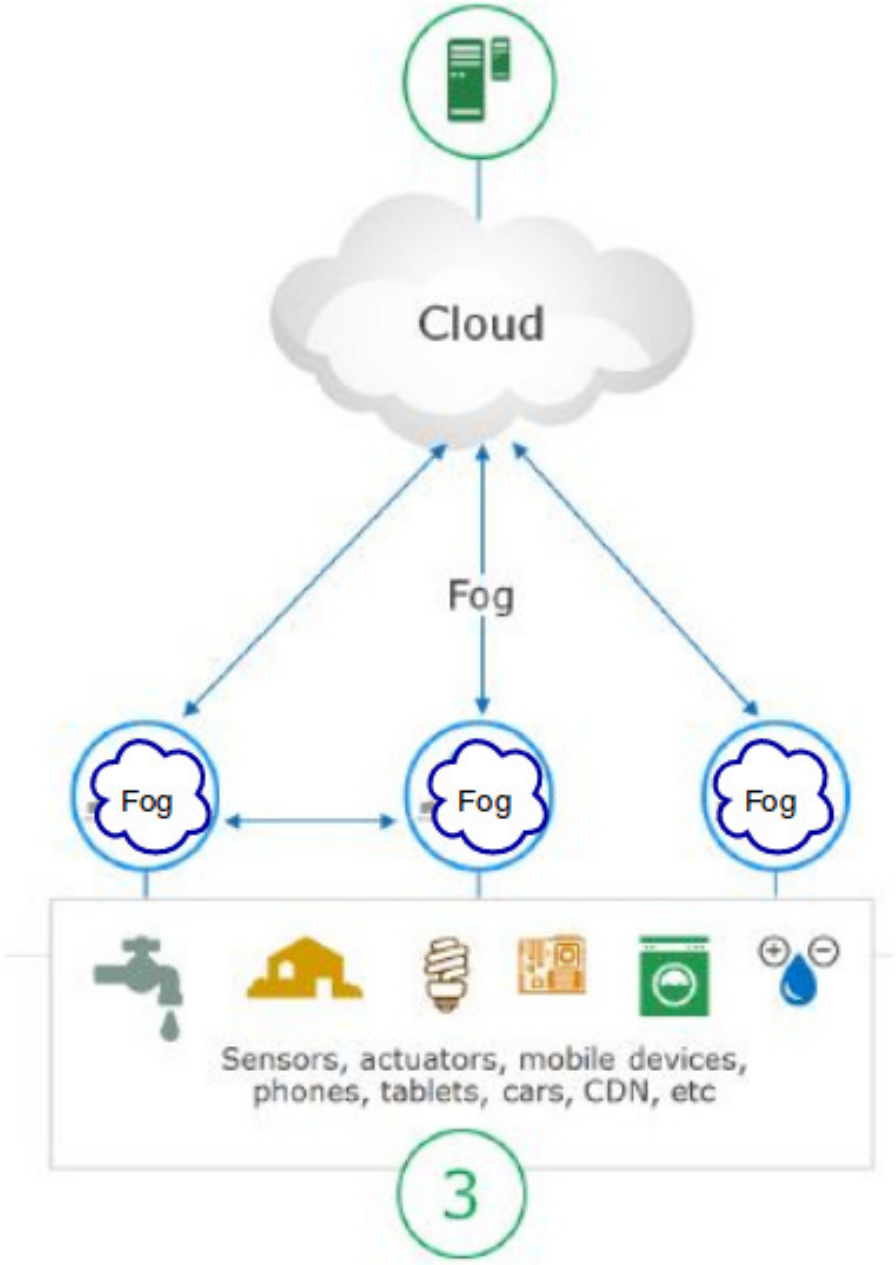


Figure A.6: Fog deployment model for sensitive applications [1]

A.2.4 Model 4: Cloud-Only Model

Figure A.7 represents a fog deployment where the cloud is used for the entire stack of the hierarchy. This may be due to constrained environments in which the deployment of fog infrastructure may not be feasible or economical.

Example use cases are remote weather stations, agriculture and connected cars.

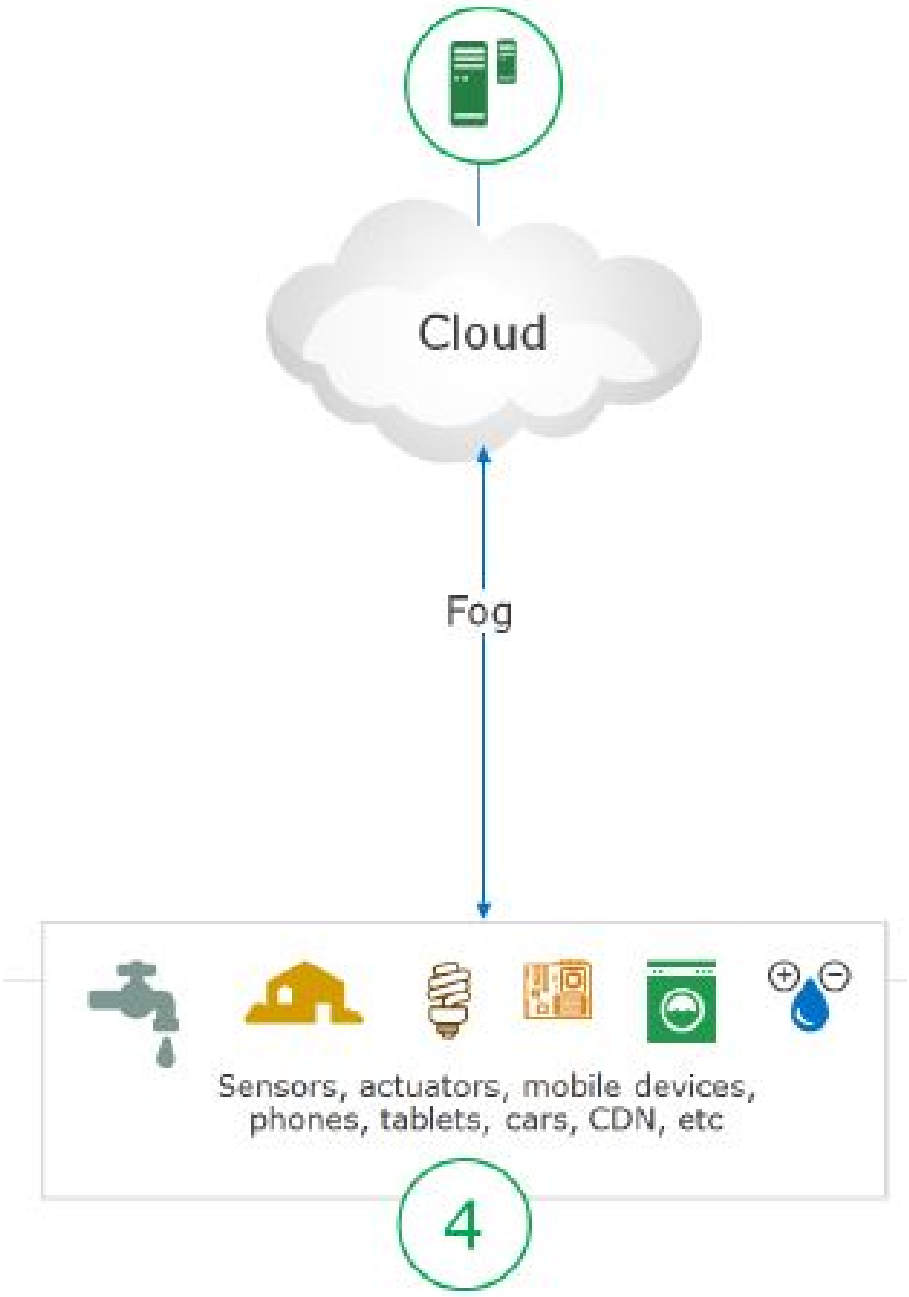


Figure A.7: Cloud only fog deployment model [1]

A.3 Impact of Virtualization on Testing Methods

To understand the impact of virtualization on testing methods, it is instructive to revisit the new components, interfaces and capabilities introduced in the NFV architecture [56]. Some of these new capabilities that change the way systems are tested are:

i. In traditional systems, applications run on dedicated physical devices with dedicated resources. In a virtualised environment, VNFs run in a shared compute, storage and network environment and may contend for the same resources.

ii. The Virtualization Layer, consisting of a hypervisor, OS container and/or vSwitch, abstracts the resource details from the VNFs. The performance of the NFV Infrastructure is influenced by the type of load (network versus IT workload, CPU intensive, memory intensive or storage intensive) and number of VNFs executing.

iii. Special data plane acceleration mechanisms may be used for IO intensive applications.

iv. NFV allows for service chaining, where a forwarding graph may be designed to optimize the path a packet flow will take from its source to its destination. The path may consist of one or multiple VNFs, which may or may not be present on the same NFVI.

While these concepts are largely independent of the core function accomplished by VNFs, the overall performance of the VNFs maybe influenced. For this reason, it is prudent to identify the main configuration items (variables) in virtualized systems. These are listed below [56]:

i. Resources allocated to the VNF, for example compute cores and memory allocation.

ii. Resources allocated to the vSwitch.

iii. Virtualization layer (hypervisor) used.

iv. The Hardware resources, including compute, networking (NIC) and storage used.

v. The usage (or not) of data plane acceleration techniques.

vi. The presence or absence of other VNFs on the same NFVI (multi-tenancy), and the function of these VNFs.

Appendix B

Source Code

B.1 Source Code for Generating Emulated Sensor Measurements

```
#!/usr/bin/python2

from numpy import random
from datetime import timedelta, datetime
import sys
import time
import requests


# Target temperatures in C
temperature = 220
pressure = 1100
humidity = 70

# Connection info
write_url = 'http://localhost:9092/write?db=printer&rp=autogen&precision=s'
measurement = 'sensor_fusion'

def temp(target, sigma):
    """
    Pick a random temperature from a normal distribution
```

```
    centered on target temperature.
    """
    return random.normal(target, sigma)

def main():
    temperature_sigma = 0
    pressure_sigma = 0
    humidity_sigma = 0
    temperature_offset = 0
    pressure_offset = 0
    humidity_offset = 0

    # Define some anomalies by changing sigma at certain times
    # list of sigma values to start at a specified iteration
    temperature_anomalies =[
        (0, 0.1, 0), # normal sigma
        (3600, 3.0, -1.5), # at one hour the temperature goes bad
        (3900, 0.1, 0), # 5 minutes later recovers
    ]
    pressure_anomalies =[
        (0, 0.1, 0), # normal sigma
        (28800, 5.0, 2.0), # at 8 hours the pressure goes bad
        (29700, 0.1, 0), # 15 minutes later recovers
    ]
    humidity_anomalies = [
        (0, 0.1, 0), # normal sigma
        (10800, 5.0, 0), # at 3 hours humidity starts to fluctuate more
        (43200, 10.0, -5.0), # at 12 hours humidity goes really bad
        (45000, 5.0, 0), # 30 minutes later recovers
        (72000, 0.1, 0), # at 20 hours goes back to normal
    ]

    # Start from 2018-10-14 00:00:00 UTC
    # This makes it easy to reason about the data later
    now = datetime(2018, 10, 14)
    second = timedelta(seconds=1)
    epoch = datetime(1970,1,1)

    # InfluxDB header
```

```
ddl = '# DDL'
createScript = 'CREATE DATABASE iot1'
dml = '# DML'
context = '# CONTEXT-DATABASE: iot1'

# 24 hours of temperatures once per second
points = []
pointsDb = []

# Add influxDb headers
pointsDb.append(ddl)
pointsDb.append(createScript)
pointsDb.append(dml)
pointsDb.append(context)

# 24 hours of temperatures once per second
for i in range(60*60*24+2):
    # update sigma values
    if len(temperature_anomalies) > 0 and i == temperature_anomalies[0][0]:
        temperature_sigma = temperature_anomalies[0][1]
        temperature_offset = temperature_anomalies[0][2]
        temperature_anomalies = temperature_anomalies[1:]

    if len(pressure_anomalies) > 0 and i == pressure_anomalies[0][0]:
        pressure_sigma = pressure_anomalies[0][1]
        pressure_offset = pressure_anomalies[0][2]
        pressure_anomalies = pressure_anomalies[1:]

    if len(humidity_anomalies) > 0 and i == humidity_anomalies[0][0]:
        humidity_sigma = humidity_anomalies[0][1]
        humidity_offset = humidity_anomalies[0][2]
        humidity_anomalies = humidity_anomalies[1:]

    # generate temps
    currentTemperature = temp(temperature+temperature_offset,
        temperature_sigma)
    currentPressure = temp(pressure+pressure_offset, pressure_sigma)
    currentHumidity = temp(humidity+humidity_offset, humidity_sigma)
```

```
        points.append("%s temperature=%f,pressure=%f,humidity=%f %d" %
            (measurement, currentTemperature, currentPressure, currentHumidity,
            time.time() + i))
        pointsDb.append("%s temperature=%f,pressure=%f,humidity=%f %d" %
            (measurement, currentTemperature, currentPressure, currentHumidity,
            time.time() + i))
        #now += second

    # Write data to file
    printerDataFile = open("printer_data.dat", "w")
    printerDataFile.write('\n'.join(pointsDb))
    printerDataFile.close()



    # Write data to Kapacitor
    r = requests.post(write_url, data='\n'.join(points))
    print r.status_code
    if r.status_code != 204:
        print >> sys.stderr, r.text
        return 1
    return 0

if __name__ == '__main__':
    exit(main())
```

B.2 Kapacitor TICK Script for Downsampling Streaming Data

```
dbrp "iot1"."autogen"
var temperatureCritical = 220

var pressureCritical = 1100

var humidityCritical = 70

var data = batch
    |query('SELECT * FROM "iot1"."autogen".sensor_fusion WHERE time > now() -
       60s')
        .period(5m)
        .every(5m)

// Calculate and POST 5 minute mean of temperature data
data
    |mean('temperature')
    |httpPost('https://influxdata-dff87.firebaseio.com//SensorFusion/temperature.json')

// Report in real-time all data points where an anomaly in temperature is
   detected
data
    |alert()
        .warn(lambda: "temperature" > temperatureCritical - 2)
        .post('https://influxdata-dff87.firebaseio.com//SensorFusion/temperatureWarn.json')

// Calculate and POST 5 minute mean of pressure data
data
    |mean('pressure')
    |httpPost('https://influxdata-dff87.firebaseio.com//SensorFusion/pressure.json')

// Report in real-time all data points where an anomaly in pressure is
   detected
data
    |alert()
        .warn(lambda: "pressure" > pressureCritical - 2)
        .post('https://influxdata-dff87.firebaseio.com//SensorFusion/pressureWarn.json')
```

```
// Calculate and POST 5 minute mean of humidity data
data
    |mean('humidity')
    |httpPost('https://influxdata-dff87.firebaseio.com//SensorFusion/humidity.json')

// Report in real-time all data points where an anomaly in humidity is
    detected
data
    |alert()
        .warn(lambda: "humidity" > humidityCritical - 2)
        .post('https://influxdata-dff87.firebaseio.com//SensorFusion/humidityWarn.json')
```

B.3 Scripts Executed in the VNF Life-cycle

Scripts contain all the required shell scripts for instatiating, configuring, and starting a VNF on the Virtual Machines or containers instantiated during the lifecycle. The execution order is defined by the lifecycle_events inside the VNFD.

install.sh

```
#!/bin/bash
#sudo apt-get update
sudo apt-get install -y python2.7
#install pip
apt-get install -y python-pip
sudo python -m pip install --upgrade pip
sudo python -m pip install --upgrade setuptools
sudo pip install scipy
sudo pip install requests
```

install-tick.sh

```
#!/bin/bash
#sudo dpkg -i telegraf_1.8.1-1_amd64.deb
sudo dpkg -i influxdb_1.6.3_amd64.deb
sudo dpkg -i chronograf_1.6.2_amd64.deb
```

```
sudo dpkg -i kapacitor_1.5.1_amd64.deb
#start kapacitor daemon
sudo kapacitord &
kapacitor define sensor_fusion -tick sensor_fusion.tick
#generate day's worth of data
chmod +x ./generate_data.py
./generate_data.py
#insert data into influx database
mycurrentdir=$(pwd)
datafile="$mycurrentdir/sensor_data.dat"
echo $datafile
influx -import -path=$datafile -precision=s
```

B.4 NSD for Prototype Industrial IoT Application

```
{
  "name":"IoTGateway",
  "vendor":"Tasimba Chirindo",
  "version":"1.0.0",
  "vld":[
    {
      "name":"dummy"
    }
  ],
  "vnfd":[
    {
      "name":"iot-gateway-1",
      "vendor":"Tasimba Chirindo",
      "version":"1.0.0",
      "lifecycle_event":[
        {
          "event":"INSTANTIATE",
          "lifecycle_events":[
            "install.sh"
          ]
        }
      ],
```

```
      "vdu":[
        {
          "vm_image":[
            "ubuntu-gateway"
          ],
          "scale_in_out":1,
          "vnfc":[
            {
              "connection_point":[
                {
                  "floatingIp":"random",
                  "virtual_link_reference":"dummy",
                  "interfaceId":0
                }
              ]
            }
          ],
          "vimInstanceName":[]
        }
      ],
      "virtual_link":[
        {
          "name":"dummy"
        }
      ],
      "deployment_flavour":[
        {
          "flavour_key":"m1.slice"
        }
      ],
      "type":"client",
      "endpoint":"generic",
      "vnfPackageLocation":"https://nyatsimbamutota1@bitbucket.org/nyatsimbamutota1/pushing-
    }
  ]
}
```

Appendix C

Evaluation of Prototype Industrial IoT Application

C.1 Analytics Results from Prototype Industrial IoT Application

This section explains the results obtained from running the IoT POC over a 24 hour time period. The expected behavior of the experiment is inscribed in the source code pasted in Appendix B.1.

Figure C.1 shows an overview of the dashboard analytics visible at the gateway device. The dashboard shows the temperature, pressure and humidity values of the process in real-time as the data is streamed from the sensors. The start time of the experiment was 07:43. From the source code, the the temperature is expected to go bad for a period of 5 minutes an hour from the start of the experiment. This result is shown on the dashboard view in Figure C.2.

The pressure process was programmed to go unstable 8 hours after the start of the experiment for a period of 15 minutes. This result is shown on the dashboard view in Figure C.3. Similary, the humidity process was expected to go unstable 3 hours from start of experiment for a period of 17 hours. Figures C.4, C.5, C.6 and C.7 show this.

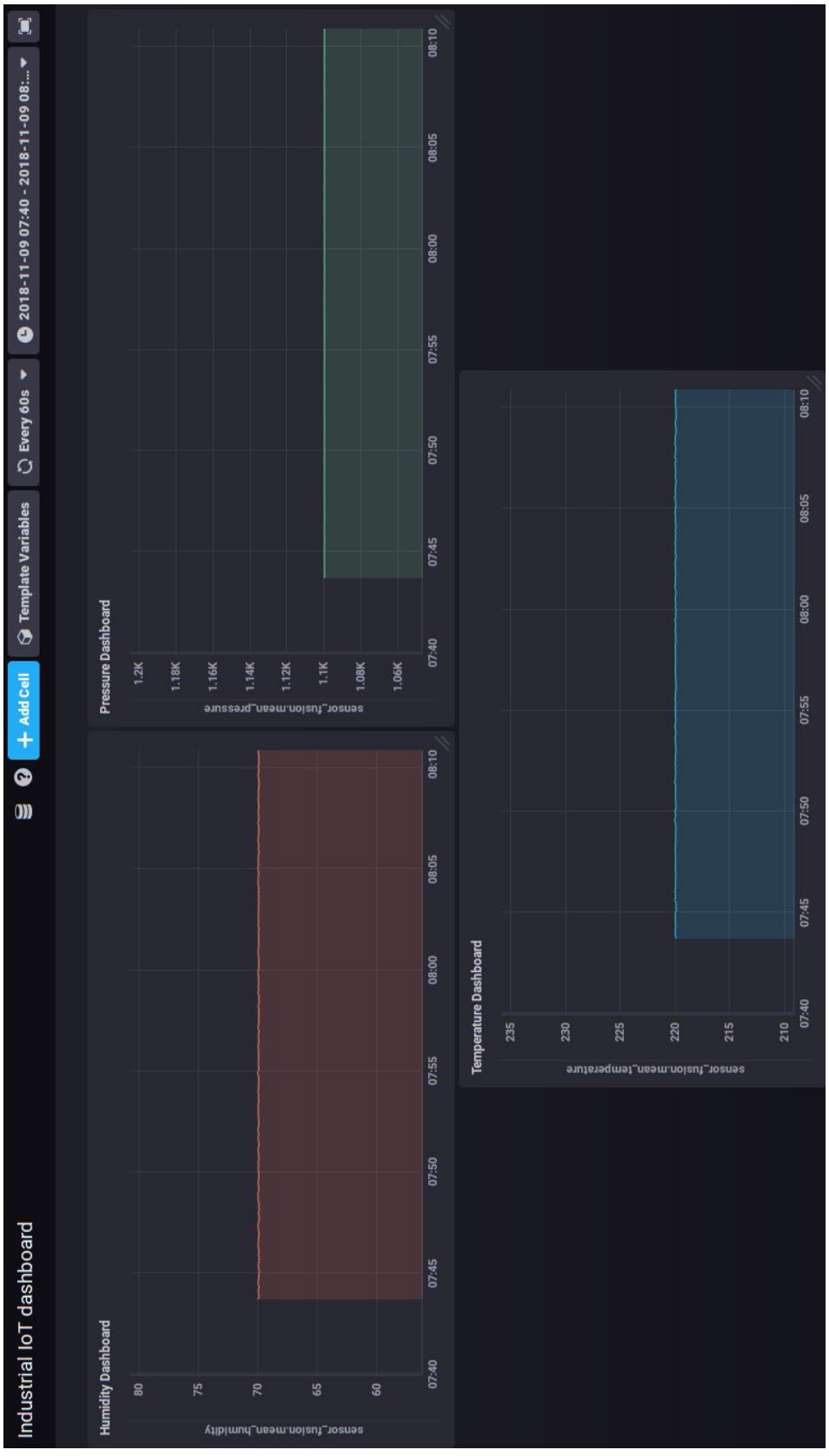


Figure C.1: Overall dashboard view

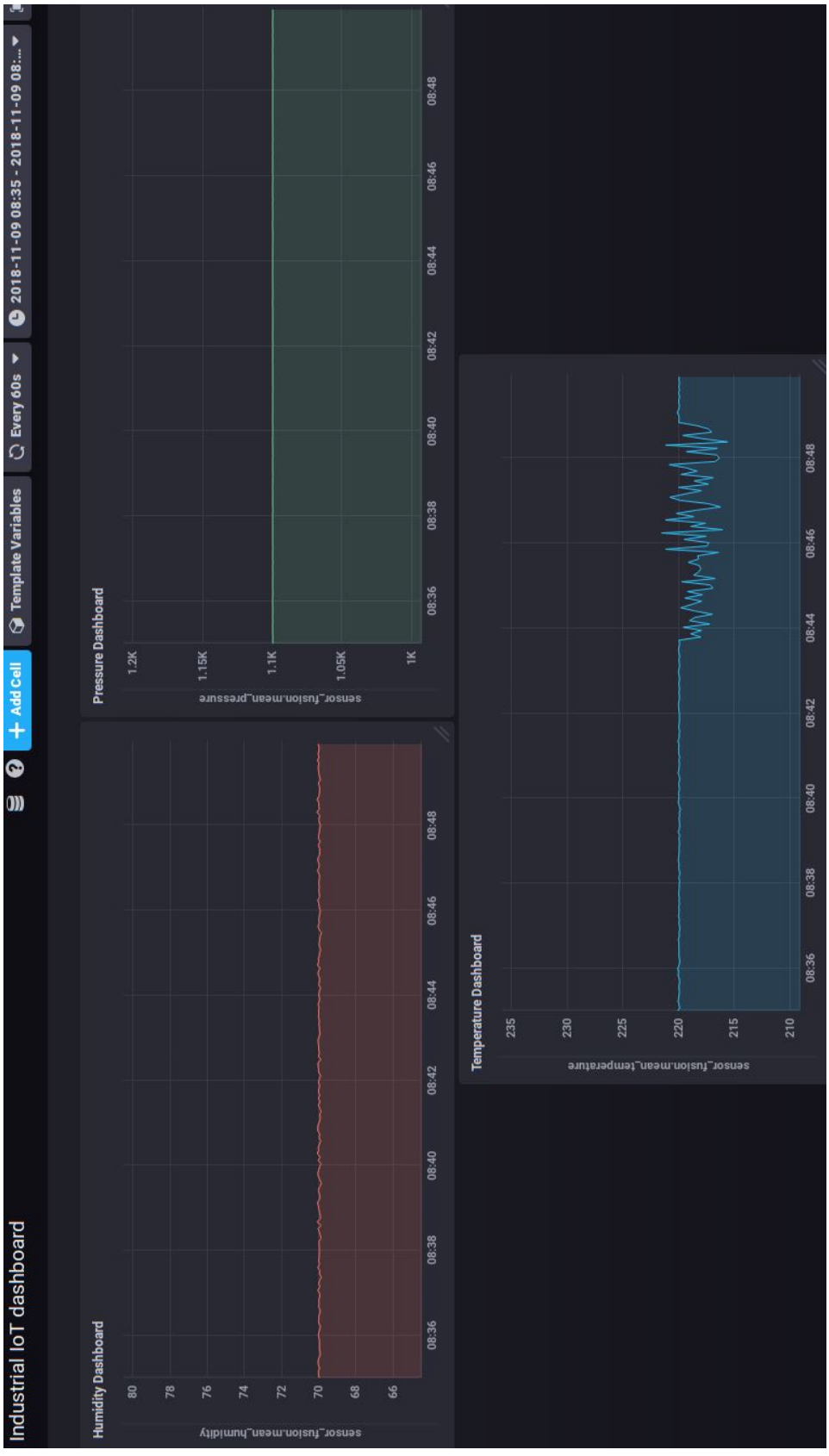


Figure C.2: Temperature anomaly in the monitored process after 1hr

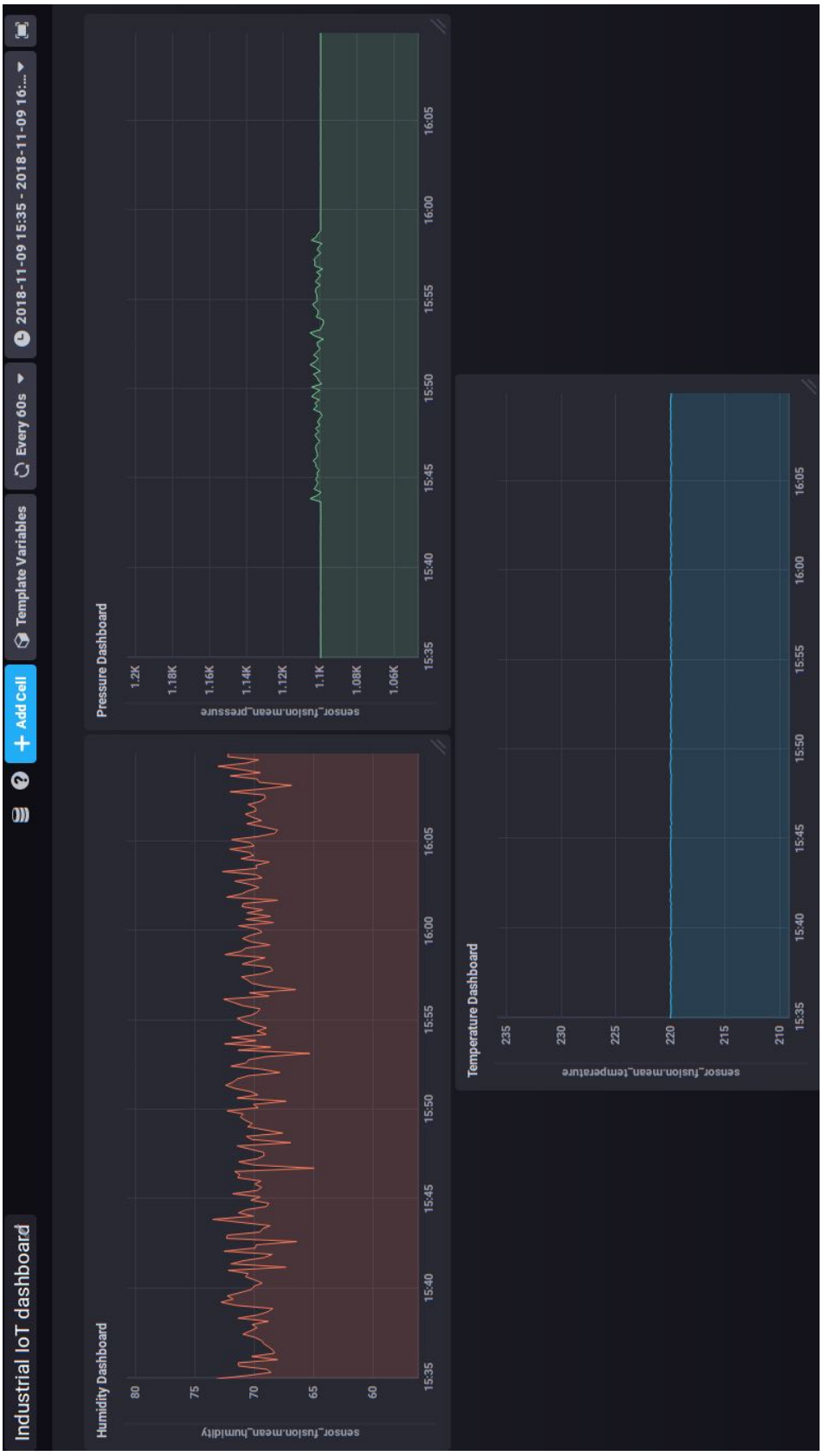


Figure C.3: Pressure anomaly in the monitored process after 8hr

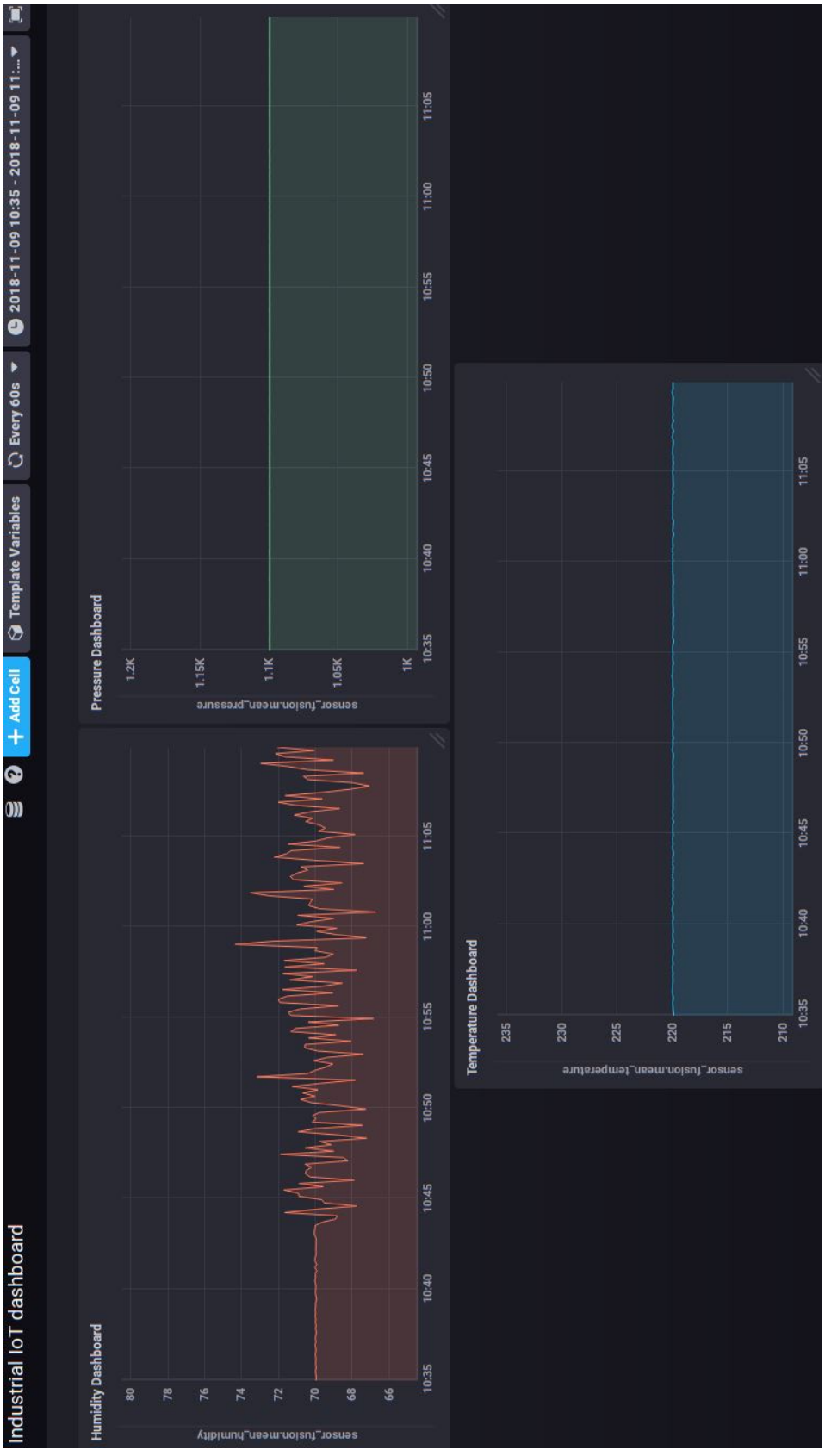


Figure C.4: Start of humidity process going unstable after 3hr

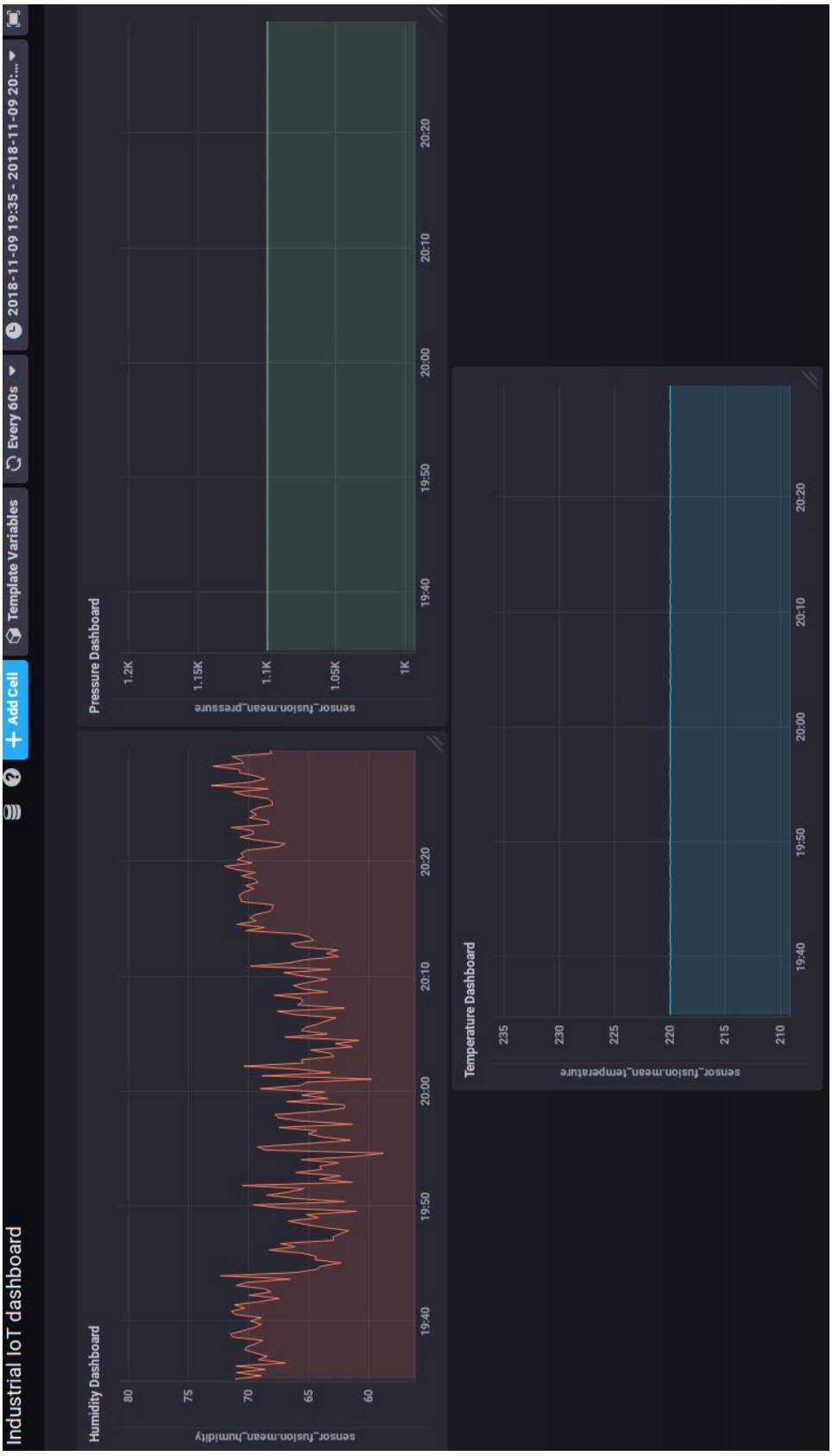


Figure C.5: Humidity process goes very unstable after 12hr

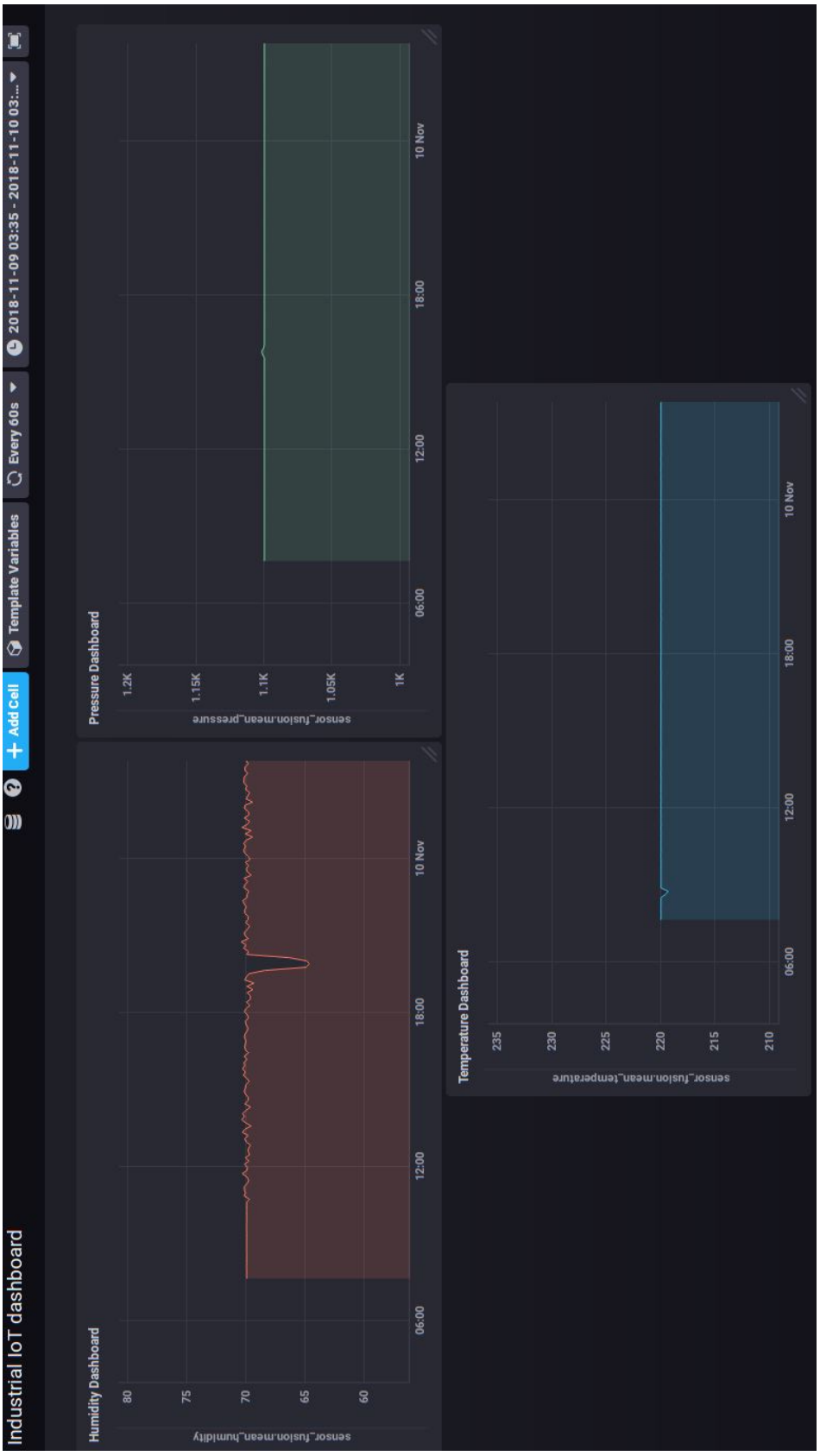


Figure C.6: Overall view of the humidity process for entire time window

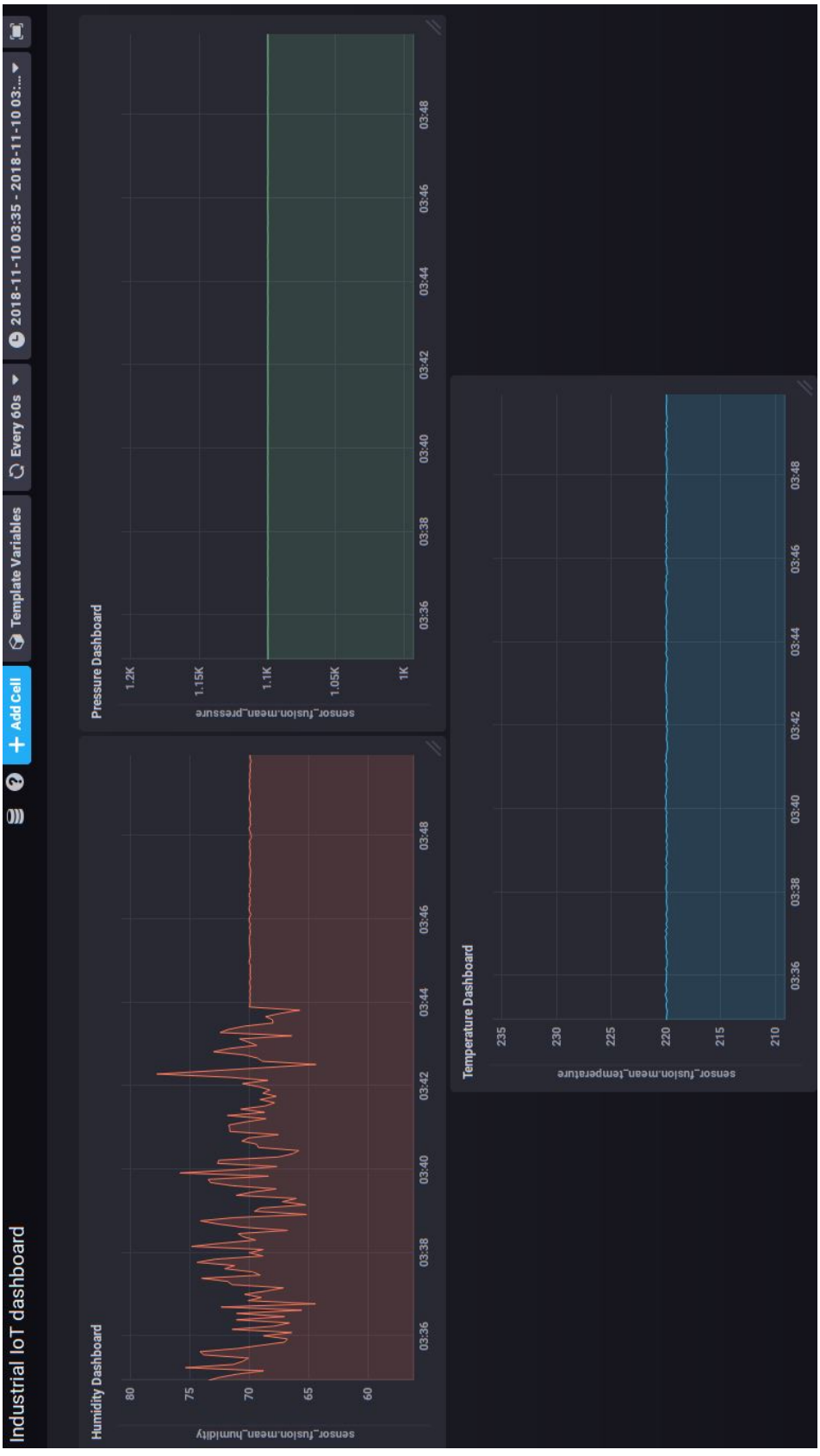


Figure C.7: Humidity process eventually stabilizes after after 12.5hr

Appendix D

Results from Feature Verification of the OVAFF

D.1 Obtaining Authentication Token

To send REST requests to the NFVO, you first have to get a token which you then have to pass in the header of every request. The token used in the calls in the rest of Appendix D was retrieved by executing this curl request:

```
curl -v -u openbatonOSClient:secret -X POST http://localhost:8080/oauth/token
    -H "Accept:application/json" -d
    "username=admin&password=openbaton&grant_type=password"
```

The answer from the orchestrator will look something like this:

```
{
  "value": "ac85d1ab-03c8-4633-92d9-a8edb509eb9c",
  "expiration": "Nov 13, 2018 7:12:43 PM",
  "tokenType": "bearer",
  "refreshToken": {
    "expiration": "Dec 7, 2018 2:25:18 PM",
    "value": "99ccc868-3694-48ee-b37e-4b801640a27e"
  },
```

```
  "scope": [
    "read",
    "write"
  ],
  "additionalInformation": {}
  * Connection #0 to host localhost left intact
}
```

The value field contains the token. In this case ac85d1ab-03c8-4633-92d9-a8edb509eb9c

D.2 Fog MANO Subscription Process to Fog Node Hosts

The Fog MANO in the testbed implementation subscribed to the two Fog Node Hosts in the testbed. These Fog Node Hosts are identified as InP1 and InP2 in the test bed. To subscribe to these data centres, a POST request with the Fog Node Host instance meta data as JSON content in the request body to the endpoint host/api/v1/datacenters were made.

The POST call to subscribe to Fog Node Host 1 is structured as below:

```
$ curl 'http://localhost:8080/api/v1/datacenters' -i -X POST -H 'project-id:
    04aec32d-fb4d-4734-a97d-d85bfa84dbd4' -H 'Authorization: Bearer
    ac85d1ab-03c8-4633-92d9-a8edb509eb9c' -H 'Content-Type: application/json'
    -d '{
  "version": 0,
  "name": "InP1",
  "authUrl": "http://controller:35357/v3",
  "tenant": "demo",
  "username": "demo",
  "password": "demo",
  "keyPair": "key",
  "location": {
    "version": 0,
    "name": "RegionOne",
```

```
    "latitude": "52.525876",
    "longitude": "13.314400"
  },
  "type": "openstack",
  "active": true
}'
```

The POST call to subscribe to Fog Node Host 2 is structured as below:

```
$ curl 'http://localhost:8080/api/v1/datacenters' -i -X POST -H 'project-id:
    04aec32d-fb4d-4734-a97d-d85bfa84dbd4' -H 'Authorization: Bearer
    ac85d1ab-03c8-4633-92d9-a8edb509eb9c' -H 'Content-Type: application/json'
    -d '{
  "version": 0,
  "name": "InP2",
  "authUrl": "http://controller1:35357/v3",
  "tenant": "demo",
  "username": "demo",
  "password": "demo",
  "keyPair": "key1",
  "location": {
    "version": 0,
    "name": "RegionTwo",
    "latitude": "52.525876",
    "longitude": "13.314400"
  },
  "type": "openstack",
  "active": true
}'
```

These POST calls returned 201 response codes indicating that the Fog MANO had successfully subscribed to the fog nodes and was now ready to launch and orchestrate services on them.

```
HTTP/1.1 201 Created
```

D.3 Test Fog MANO Subscriptions to Fog Node Host Infrastructure

To query the subscriptions and access rights established in the process defined in Appendix D.2, a GET request to the endpoint host/api/v1/datacenters was made.

```
curl 'http://localhost:8080/api/v1/datacenters' -i -H 'project-id:
    5d283732fa4c484bb268f1e260dfcc52' -H 'Authorization: Bearer
    ac85d1ab-03c8-4633-92d9-a8edb509eb9c'
```

The request returned a 200 response code with the following payload:

```
4-a97d-d85bfa84dbd4' -H 'Authorization:Bearer
    ac85d1ab-03c8-4633-92d9-a8edb509eb9c'
HTTP/1.1 200
X-Application-Context: NFVO OpenBaton:mysql
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Tue, 13 Nov 2018 18:37:11 GMT
[
  {
    "tenant": "61d4887c99514c92a54f78c6699bb20e",
    "username": "admin",
    "password": "**********",
    "keyPair": "mykey",
    "securityGroups": [
      "admin-group"
    ],
    "keys": [
      {
        "name": "mykey",
        "publicKey": "ssh-rsa
```

```
        AAAAB3NzaC1yc2EAAAADAQABAAABAQDNuRHOCtb+v3xLq89Qsqb32vx+HMHs8m6ZZqGmJTw8zhbdBNweN
        stack@controller\n",
      "fingerprint": "b0:b5:38:1d:bd:28:72:9a:c5:22:8d:94:45:4d:c2:28",
      "id": "fe9b60d5-4e78-40ef-8280-e55ad2364932",
      "hbVersion": 0,
      "projectId": "04aec32d-fb4d-4734-a97d-d85bfa84dbd4",
      "shared": false,
      "metadata": {}
    }
  ],
  "flavours": [
    {
      "flavour_key": "m1.miot",
      "extId": "3",
      "ram": 8192,
      "disk": 50,
      "vcpus": 2,
      "id": "ba56daf7-2e4b-4196-92e8-90432a74c999",
      "hbVersion": 0,
      "shared": false,
      "metadata": {}
    }
  ],
  "images": [
    {
      "name": "cirros",
      "minRam": 0,
      "minDiskSpace": 0,
      "isPublic": true,
      "diskFormat": "QCOW2",
      "containerFormat": "BARE",
      "updated": "Feb 16, 2018 9:35:49 PM",
      "status": "ACTIVE",
      "extId": "eeaae167-b7e4-431d-b43c-de58566326d5",
      "created": "Feb 16, 2018 9:35:49 PM",
      "id": "265e9ace-5dd1-4e6b-9e8a-d23eb98ffec7",
      "hbVersion": 0,
      "shared": false,
      "metadata": {}
```

```
    },
    {
      "name": "ubuntu-cloud",
      "minRam": 0,
      "minDiskSpace": 0,
      "isPublic": true,
      "diskFormat": "QCOW2",
      "containerFormat": "BARE",
      "updated": "Feb 16, 2018 9:36:22 PM",
      "status": "ACTIVE",
      "extId": "dfa0b48e-6f82-4365-bd78-219ed5b6376e",
      "created": "Feb 16, 2018 9:36:20 PM",
      "id": "98bf77ef-d057-40df-904f-6e1d412d4d9f",
      "hbVersion": 0,
      "shared": false,
      "metadata": {}
    }
  ],
  "networks": [
    {
      "external": false,
      "extShared": false,
      "subnets": [
        {
          "name": "private_subnet",
          "extId": "fa4ca9bb-8c72-4c33-9164-8c19e1a6b9c8",
          "networkId": "7750aff9-91a8-4b23-b267-45be666212ca",
          "cidr": "192.168.181.0/24",
          "gatewayIp": "192.168.181.1",
          "dns": [],
          "id": "9f4f3184-0607-4229-9e12-917f3c35a265",
          "hbVersion": 0,
          "shared": false,
          "metadata": {}
        }
      ],
      "name": "private",
      "extId": "7750aff9-91a8-4b23-b267-45be666212ca",
      "id": "3f4d11b9-85f8-4aa6-8fe5-7a26445bc84a",
```

```
      "hbVersion": 1,
      "shared": false,
      "metadata": {}
    },
    {
      "external": false,
      "extShared": false,
      "subnets": [
        {
          "name": "dummy_subnet",
          "extId": "3c0eddd6-ec3c-4433-834e-bb0ce0522f9a",
          "networkId": "5bbaf2d5-4306-4bd7-ab33-38ddf4e5da72",
          "cidr": "192.168.143.0/24",
          "gatewayIp": "192.168.143.1",
          "dns": [],
          "id": "c13cabcc-f72b-4059-b9fa-ea8bc735fc38",
          "hbVersion": 0,
          "shared": false,
          "metadata": {}
        }
      ],
      "name": "dummy",
      "extId": "5bbaf2d5-4306-4bd7-ab33-38ddf4e5da72",
      "id": "fdf35d3a-1974-413b-8d45-1065175a2585",
      "hbVersion": 1,
      "shared": false,
      "metadata": {}
    }
  ],
  "zones": [
    {
      "name": "nova",
      "available": true,
      "hosts": {},
      "id": "3b3bc18e-7794-44d9-acf0-d9ce34992309",
      "hbVersion": 1,
      "shared": false,
      "metadata": {}
    }
```

```
  ],
  "name": "InP2",
  "authUrl": "http://controller1:35357/v3",
  "location": {
    "name": "RegionTwo",
    "latitude": "52.525876",
    "longitude": "13.314400",
    "id": "a6b33eed-d0b7-47bb-b9c8-bd4250272082",
    "hbVersion": 0,
    "shared": false,
    "metadata": {}
  },
  "type": "openstack",
  "active": true,
  "id": "8822bfa5-e9a2-4c45-b37e-6e44db725356",
  "hbVersion": 9,
  "projectId": "04aec32d-fb4d-4734-a97d-d85bfa84dbd4",
  "shared": false,
  "metadata": {}
},
{
  "tenant": "c97b77b5068648709faf8594ffab3907",
  "username": "admin",
  "password": "**********",
  "keyPair": "mykey",
  "securityGroups": [
    "admin-group"
  ],
  "keys": [
    {
      "name": "mykey",
      "publicKey": "ssh-rsa
         AAAAB3NzaC1yc2EAAAADAQABAAABAQDfwt1OOM9Hma08C/RHQM7NBe/mPiAKRtc6Gi8O801PQUPJurocl
         stack@controller\n",
      "fingerprint": "f7:b6:71:0b:3a:dd:29:3d:2f:86:75:b9:e5:d3:bc:86",
      "id": "9b964aeb-ad5b-4154-b3b8-cf56a91660fa",
      "hbVersion": 0,
      "projectId": "04aec32d-fb4d-4734-a97d-d85bfa84dbd4",
      "shared": false,
```

```
      "metadata": {}
    }
  ],
  "flavours": [
    {
      "flavour_key": "m1.epc",
      "extId": "5f6b4a87-cda6-43a7-b86f-91c3335d2583",
      "ram": 4096,
      "disk": 40,
      "vcpus": 1,
      "id": "dc7be75a-a619-423c-bcf8-d05a07289ee2",
      "hbVersion": 0,
      "shared": false,
      "metadata": {}
    },
    {
      "flavour_key": "m1.slice",
      "extId": "1",
      "ram": 2048,
      "disk": 20,
      "vcpus": 1,
      "id": "2cfef9ac-47e3-4c3d-b1d6-607a13f0fafe",
      "hbVersion": 0,
      "shared": false,
      "metadata": {}
    },
    {
      "flavour_key": "m1.miot",
      "extId": "3",
      "ram": 8192,
      "disk": 50,
      "vcpus": 2,
      "id": "1d8edf99-f374-4611-87b5-bb58720ec1ce",
      "hbVersion": 0,
      "shared": false,
      "metadata": {}
    },
    {
      "flavour_key": "m1.urllc",
```

```
      "extId": "4",
      "ram": 4096,
      "disk": 50,
      "vcpus": 1,
      "id": "0382a098-3271-4775-904e-57aa02525bfa",
      "hbVersion": 0,
      "shared": false,
      "metadata": {}
    },
    {
      "flavour_key": "m1.embb",
      "extId": "2",
      "ram": 16384,
      "disk": 50,
      "vcpus": 4,
      "id": "7f4a0c1c-1952-4bac-b448-1b235cfdabbc",
      "hbVersion": 0,
      "shared": false,
      "metadata": {}
    },
    {
      "flavour_key": "m1.nano",
      "extId": "0",
      "ram": 64,
      "disk": 1,
      "vcpus": 1,
      "id": "5d666b7a-444d-4c03-8d4e-ab43cf503d64",
      "hbVersion": 0,
      "shared": false,
      "metadata": {}
    }
  ],
  "images": [
    {
      "name": "ubuntu-cloud",
      "minRam": 0,
      "minDiskSpace": 0,
      "isPublic": true,
      "diskFormat": "QCOW2",
```

```
      "containerFormat": "BARE",
      "updated": "Jan 24, 2018 5:59:16 PM",
      "status": "ACTIVE",
      "extId": "bf898fd4-88bc-4e65-a89d-8ea10a063af0",
      "created": "Jan 24, 2018 5:59:15 PM",
      "id": "23580f22-4aa9-4d76-8f9d-3cb7ba1bb056",
      "hbVersion": 0,
      "shared": false,
      "metadata": {}
    },
    {
      "name": "cirros",
      "minRam": 0,
      "minDiskSpace": 0,
      "isPublic": true,
      "diskFormat": "QCOW2",
      "containerFormat": "BARE",
      "updated": "Dec 8, 2017 3:06:01 PM",
      "status": "ACTIVE",
      "extId": "57e85497-b04d-42ef-9999-06910f2a2cb2",
      "created": "Dec 8, 2017 3:06:01 PM",
      "id": "948f1cf5-18be-49b4-ab11-6029862b2728",
      "hbVersion": 0,
      "shared": false,
      "metadata": {}
    }
  ],
  "networks": [
    {
      "external": false,
      "extShared": false,
      "subnets": [
        {
          "name": "dummy_subnet",
          "extId": "8a127d0c-e372-477a-91e0-85767f639f2d",
          "networkId": "7ddc7e09-e70a-41c8-a2a8-9a3703cb45f0",
          "cidr": "192.168.110.0/24",
          "gatewayIp": "192.168.110.1",
          "dns": [],
```

```
        "id": "65276925-860a-4bbd-854f-45e121d8d781",
        "hbVersion": 0,
        "shared": false,
        "metadata": {}
      }
    ],
    "name": "dummy",
    "extId": "7ddc7e09-e70a-41c8-a2a8-9a3703cb45f0",
    "id": "20c1f52c-63f6-47dc-969d-db1fd7591bb5",
    "hbVersion": 1,
    "shared": false,
    "metadata": {}
  },
  {
    "external": true,
    "extShared": false,
    "subnets": [
      {
        "name": "provider",
        "extId": "1c0e254c-ae0a-4ece-bfc2-c0913bbc1919",
        "networkId": "c4584d75-5447-4cbc-836b-f33c65f5f022",
        "cidr": "137.158.126.0/27",
        "gatewayIp": "137.158.126.1",
        "dns": [],
        "id": "2265a276-c12f-4747-ad1a-a5c8a66a3ebe",
        "hbVersion": 0,
        "shared": false,
        "metadata": {}
      }
    ],
    "name": "provider",
    "extId": "c4584d75-5447-4cbc-836b-f33c65f5f022",
    "id": "d2470379-924a-46a0-8b10-67d4fa80cce6",
    "hbVersion": 1,
    "shared": false,
    "metadata": {}
  }
],
"zones": [
```

```
      {
        "name": "nova",
        "available": true,
        "hosts": {},
        "id": "6bcc505f-152e-44ed-8661-8ebdc658285e",
        "hbVersion": 1,
        "shared": false,
        "metadata": {}
      }
    ],
    "name": "InP1",
    "authUrl": "http://controller:35357/v3",
    "location": {
      "name": "RegionOne",
      "latitude": "52.525876",
      "longitude": "13.314400",
      "id": "6c8a61ac-6cc0-4f19-a637-8b42035d2b59",
      "hbVersion": 0,
      "shared": false,
      "metadata": {}
    },
    "type": "openstack",
    "active": true,
    "id": "91348be1-4aa5-4e33-8520-2b6834869dc8",
    "hbVersion": 11,
    "projectId": "04aec32d-fb4d-4734-a97d-d85bfa84dbd4",
    "shared": false,
    "metadata": {}
  }
]
```


D.4 Test Upload of Network Service Descriptor to Fog MANO

To upload a VNFD of the industrial IoT application, a POST request to the endpoint host/api/v1/vnf-descriptors/ was made.

```
$ curl 'http://localhost:8080/api/v1/ns-descriptors' -i -X POST -H
    'project-id:
8a387f1e-9a42-43c7-bab0-3915719c6fca' -H 'Authorization: Bearer
    ac85d1ab-03c8-4633-92d9-a8edb509eb9c' -H 'Content-Type: application/json'
    -d '{
  "name": "IoTGateway",
  "vendor": "Tasimba Chirindo",
  "version": "1.0.0",
  "vld": [
    {
      "name": "dummy"
    }
  ],
  "vnfd": [
    {
      "name": "iot-gateway-1",
      "vendor": "Tasimba Chirindo",
      "version": "1.0.0",
      "lifecycle_event": [
        {
          "event": "INSTANTIATE",
          "lifecycle_events": [
            "install.sh"
          ]
        }
      ],
      "vdu": [
        {
          "vm_image": [
            "ubuntu-gateway"
          ],
          "scale_in_out": 1,
          "vnfc": [
            {
              "connection_point": [
                {
                  "floatingIp": "random",
                  "virtual_link_reference": "dummy",
```

```
                "interfaceId": 0
              }
            ]
          }
        ],
        "vimInstanceName": []
      }
    ],
    "virtual_link": [
      {
        "name": "dummy"
      }
    ],
    "deployment_flavour": [
      {
        "flavour_key": "m1.slice"
      }
    ],
    "type": "client",
    "endpoint": "generic",
    "vnfPackageLocation":
        "https://nyatsimbamutota1@bitbucket.org/nyatsimbamutota1/pushing-iot-analytics-to-t
  }
 ]
}'
```

The request returned a 201 response code with the following payload:

```
HTTP/1.1 201 Created
```

D.5 Test Retrieve all Uploaded Network Service Descriptors

```
curl 'http://localhost:8080/api/v1/ns-descriptors' -i -H 'project-id:
   04aec32d-fb4d-4734-a97d-d85bfa84dbd4' -H 'Authorization: Bearer
   ac85d1ab-03c8-4633-92d9-a8edb509eb9c'
```

D.6 Test Instantiate Network Service

```
curl
    'http://localhost:8080/api/v1/ns-records/c60ec37a-654a-4605-93fb-5a1932db6ecf'
    -i
-X POST -H 'project-id: 04aec32d-fb4d-4734-a97d-d85bfa84dbd4' -H
    'Authorization: Bearer
ac85d1ab-03c8-4633-92d9-a8edb509eb9c' -H 'Content-Type: application/json' -d
    '{
  "keys": [
    "mykeypair",
  ],
  "configurations": {
  },
  "vduVimInstances": {
    "IoTGateway": [
      "vim-instance"
    ],
    "serverVdu1": [
      "vim-instance"
    ]
  }
}
```

D.7 Test Terminate Network Service

```
$ curl
    'http://localhost:8080/api/v1/ns-records/66046d77-aade-4f14-ad39-f2976532e5f2'
    -i
-X DELETE -H 'project-id: 04aec32d-fb4d-4734-a97d-d85bfa84dbd4' -H
    'Authorization: Bearer ac85d1ab-03c8-4633-92d9-a8edb509eb9c'
```

The request returned a 204 success response code with the following payload:

```
HTTP/1.1 204 No Content
```

D.8 Test Retrieve Network Service Records

```
$ curl 'http://localhost:8080/api/v1/ns-records/' -i -H 'project-id:
    04aec32d-fb4d-4734-a97d-d85bfa84dbd4' -H 'Authorization: Bearer
    ac85d1ab-03c8-4633-92d9-a8edb509eb9c'
```

The request returned a 200 success response code with the following payload:

```
[
  {
  }
]
```

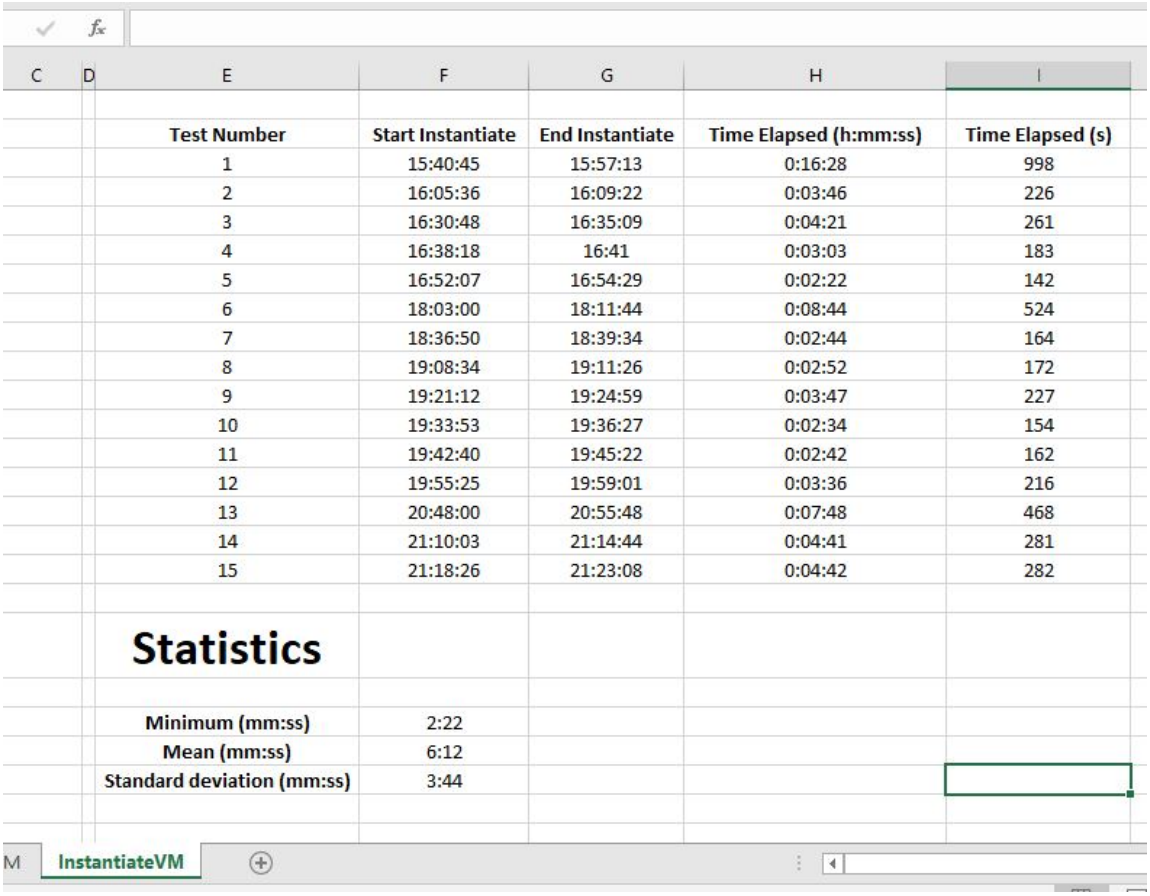
D.9 Data for VM Deployment Time Measurements

C	D	E	F
Test Number	**Start Deploy**	**End Deploy**	**Time Elapsed (s)**
1	15:40:04	15:40:44	40
2	16:05:04	16:05:35	31
3	16:30:10	16:30:47	37
4	16:38:02	16:38:17	15
5	16:51:41	16:52:06	25
6	18:02:22	18:02:59	37
7	18:36:32	18:36:49	17
8	19:08:04	19:08:33	29
9	19:20:45	19:21:11	26
10	19:33:25	19:33:52	27
11	19:42:18	19:42:39	21
12	19:55:11	19:55:24	13
13	20:47:32	20:47:59	27
14	21:09:36	21:10:02	26
15	21:18:01	21:18:25	24
Statistics			
Minimum (s)	13		
Mean (s)	26.33333333		
Standard deviation (s)	7.898161329		

tiateVM | DeployVM | ⊕

Figure D.1: Test data for VM deployment time measurements.

D.10 Data for NSR Instantiation Time Measurements



Test Number	Start Instantiate	End Instantiate	Time Elapsed (h:mm:ss)	Time Elapsed (s)
1	15:40:45	15:57:13	0:16:28	998
2	16:05:36	16:09:22	0:03:46	226
3	16:30:48	16:35:09	0:04:21	261
4	16:38:18	16:41	0:03:03	183
5	16:52:07	16:54:29	0:02:22	142
6	18:03:00	18:11:44	0:08:44	524
7	18:36:50	18:39:34	0:02:44	164
8	19:08:34	19:11:26	0:02:52	172
9	19:21:12	19:24:59	0:03:47	227
10	19:33:53	19:36:27	0:02:34	154
11	19:42:40	19:45:22	0:02:42	162
12	19:55:25	19:59:01	0:03:36	216
13	20:48:00	20:55:48	0:07:48	468
14	21:10:03	21:14:44	0:04:41	281
15	21:18:26	21:23:08	0:04:42	282

Statistics

Minimum (mm:ss)	2:22
Mean (mm:ss)	6:12
Standard deviation (mm:ss)	3:44

Figure D.2: Test data for NSR instantiation time measurements.

Appendix E

Ethics Forms

APPLICATION FORM

Please Note:
Any person planning to undertake research in the Faculty of Engineering and the Built Environment (EBE) at the University of Cape Town is required to complete this form **before** collecting or analysing data. The objective of submitting this application *prior* to embarking on research is to ensure that the highest ethical standards in research, conducted under the auspices of the EBE Faculty, are met. Please ensure that you have read, and understood the **EBE Ethics in Research Handbook** (available from the UCT EBE, Research Ethics website) prior to completing this application form: http://www.ebe.uct.ac.za/ebe/research/ethics1

APPLICANT'S DETAILS		
Name of principal researcher, student or external applicant		Tasimba Chirindo
Department		Electrical Engineering
Preferred email address of applicant:		Chrtas004@myuct.ac.za
If Student	Your Degree: e.g., MSc, PhD, etc.	MSc (Eng.) Electrical Engineering
	Credit Value of Research: e.g., 60/120/180/360 etc.	180
	Name of Supervisor (if supervised):	Joyce Mwangama
If this is a researchcontract, indicate the source of funding/sponsorship		Click here to enter text.
Project Title		An Open Vendor Agnostic Fog Computing Framework for Mission-Critical and Data-Dense Applications

I hereby undertake to carry out my research in such a way that:
- there is no apparent legal objection to the nature or the method of research; and
- the research will not compromise staff or students or the other responsibilities of the University;
- the stated objective will be achieved, and the findings will have a high degree of validity;
- limitations and alternative interpretations will be considered;
- the findings could be subject to peer review and publicly available; and
- I will comply with the conventions of copyright and avoid any practice that would constitute plagiarism.

SIGNED BY	Full name	Signature	Date
Principal Researcher/ Student/External applicant	Tasimba Chirindo		**14 Sep 2018**

APPLICATION APPROVED BY	Full name	Signature	Date
Supervisor (where applicable)	Joyce Mwangama		**14 Sep 2018**
HOD (or delegated nominee) Final authority for all applicants who have answered NO to all questions in Section1; and for all Undergraduateresearch (Including Honours).	Click here to enter text.		Click here to enter a date.
Chair : Faculty EIR Committee For applicants other than undergraduate students who have answered YES to any of the above questions.	R Behrens		17 Sep 2018