COMBINING CLOUD, FOG, ROOF AND DEW COMPUTING WITH SDN TO IMPROVE QoS OF IoT ECOSYSTEM

ISHTIAQ AHAMMAD

MASTER OF SCIENCE IN INFORMATION AND COMMUNICATION ENGINEERING

NOAKHALI SCIENCE AND TECHNOLOGY UNIVERSITY

COMBINING CLOUD, FOG, ROOF AND DEW COMPUTING WITH SDN TO IMPROVE QoS OF IoT ECOSYSTEM

ISHTIAQ AHAMMAD

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Master of Science in Information and Communication Engineering

Department of Information and Communication Engineering

NOAKHALI SCIENCE AND TECHNOLOGY UNIVERSITY

January 2021

SUPERVISOR'S DECLARATION

We hereby declare that we have checked this thesis and in our opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Master of Science in Information and Communication Engineering.

_____

Name of Supervisor: DR. MD. ASHIKUR RAHMAN KHAN

Position: PROFESSOR & CHAIRMAN

Date:

_____

Name of Co-Supervisor: ZAYED-US-SALEHIN

Position: ASSISTANT PROFESSOR

Date:

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is my own except for quotations, references and summaries which have been duly acknowledged. This thesis has not been accepted for any degree and is not concurrently submitted for award of other degree.

Ishtiaq Ahammad

Roll: ASH1811MSc114M

Date:

Dedicated to my Parents

ACKNOWLEDGEMENTS

I am grateful and would like to express my sincere gratitude to my supervisor Professor Dr. Md. Ashikur Rahman Khan for his germinal ideas, invaluable guidance, continuous encouragement and constant support in making this research possible. He has always impressed me with his outstanding professional conduct, his strong conviction for science, and his belief that a M.Sc. program is only a start of a life-long learning experience. I appreciate his consistent support from the first day I applied to graduate program to these concluding moments. I am truly grateful for his progressive vision about my training in science, his tolerance of my native mistakes, and his commitment to my future carrier. I also would like to express very special thanks to my co-supervisor Zayed-Us-Salehin for his suggestions and co-operations throughout the study. I also sincerely thanks for the time spent proofreading and correcting my many mistakes.

My sincere thanks go to all my lab mates, members of the staff of the Information and Communication Engineering Department, NSTU, who helped me in many ways and made my stay at NSTU pleasant and unforgettable.

I acknowledge my sincere indebtedness and gratitude to my parents for their love, dream and sacrifice throughout my life. Special thanks should be given to my committee members. I would like to acknowledge their comments and suggestions which were crucial for the successful completion of this study.

ABSTRACT

Recently, there's been a growing curiosity in the Internet of Things (IoT), due to the ubiquitous existence of the internet. Such curiosity leads the concept of IoT ecosystem. But contemporary fragmented ecosystem of regulations, technologies, and systems slows IoT deployments. Because they typically concentrate on a particular use case, or are confined to a specific standard. Therefore, an IoT ecosystem which encompasses all the elements (i.e. remote controls, dashboards, communications, access points, analytics, data management, security etc.) to enable companies, governments, and users to seamlessly connect their IoT devices to the existing ecosystem framework to receive services is a necessity. But some concerns in this ubiquitous IoT ecosystem infrastructure may end up making its development a really difficult task. Thus in this paper multi-tiered computational infrastructure is considered which would be feasible to provide services from the nearest possible location of end-user devices and thus fix most of the difficulty issues. However, bringing computational infrastructures to user surroundings doesn't really automatically address all technological challenges. It also generates complications of its own if it is not properly managed. Therefore, this paper takes into account Software-Defined Networking (SDN) as a solution. SDN has shown its utility in lessening the complications of management in today's networks. Now to understand the journey behind SDN-supported multi-tier computational infrastructure; elaborate study on cloud, fog, roof, and dew computing and their interaction with SDN is performed in this paper. Then two novel methods are presented to improve QoS in an IoT ecosystem. In “Method I”, a novel architecture is proposed to combine independently researched areas of SDN and Fog Computing to improve QoS. In support of the architecture, there is also a proposed algorithm based on Virtual Partition for optimal access point and optimal place of operation selection. The “Method II” presents SD-DRFC framework architecture for today's IoT ecosystem which combines Cloud, Fog, Roof and Dew computing domains which are managed by SDN. The novelty of this SD-DRFC framework lies in its ability to deliver services from the nearest possible location, manage data routing pathways individually for various types of data, efficiently balance huge data traffic on the network, offer limited services without internet connection, and distribute network and computing resources appropriately. Next simulation setup is performed for both of these proposed methods. For simulation, a use case is defined for each method which is identical to the proposed methods architecture. These simulation setups are performed on iFogSim simulator. The result shows a significant improvement of several QoS parameters in the execution of proposed methods compared to the cloud-only execution. In specific, the simulation result of “Method I” offers much better results for energy consumption, network usage and latency compared with previous similar approaches. The result shows a 275.9% reduction in latency and a 212.21% reduction in network usage. For both “Method I” and “Method II”; the simulation results justify major improvement in some QoS parameter in favor of the proposed methods. Since these parameters are the most significant ones in an IoT ecosystem; hence their improvement indicates overall QoS improvement of IoT ecosystem.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

S	Software Defined Network
S_1, S_2,…,S_N	Small Network Domain
F_1, F_2,.., F_N	Fog Nodes (FNs) with specific region coverage
p	Present multi-network functionality in partition of controlled
q	Radio-access technologies supported
r	Types of services things are requesting
h	Handover actions among heterogeneous network of access
z	Potential destination in near future
D	Produced data by things
f_1, f_2, …., f_n	Smaller sub-regions of F
F_M	Merged Fog Node
T_S	Time sensitive data
T_L	Less time sensitive data
T_I	Time insensitive data
C_1	Topology of Network
C_2	Distance of Inter-nodes
C_3	Bandwidth between Nodes
C_4	Mean Rate of Request
C_5	Utmost Controller Retention
C_6	Least Space among Two Controllers
B_1	Optimum Progressive SDN Controller Placement
B_2	Distributed Policy Builds on Non-Zero-Sum Game

LIST OF ABBREVIATIONS

IoT	Internet-of-Things
CC	Cloud Computing
FC	Fog Computing
RC	Roof Computing
DC	Dew Computing
QoS	Quality-of-Service
SDN	Software-Defined Networking
SLA	Service Level Agreement
IDE	Integrated Development Environment
JDK	Java Development Kit
BYOD	Bring Your Own Device
VM	Virtual Machine
FN	Fog Node
AFN	Aggregate Fog Node
AP	Access Point
CPS	Cyber Physical Systems
VDC	Virtual Dew Cluster
WPAN	Wireless Personal Area Network
DS	Dew Server
DAS	Dew Analytics Server
AID	Artificial Intelligence of Dew
DUI	Dew site User Identity
WUI	Website User Identity
MCS	Mobile Cloud Synchronization
SAE	Security Analytics Engine
EPC	Evolved Packet Core

CHAPTER 1

INTRODUCTION

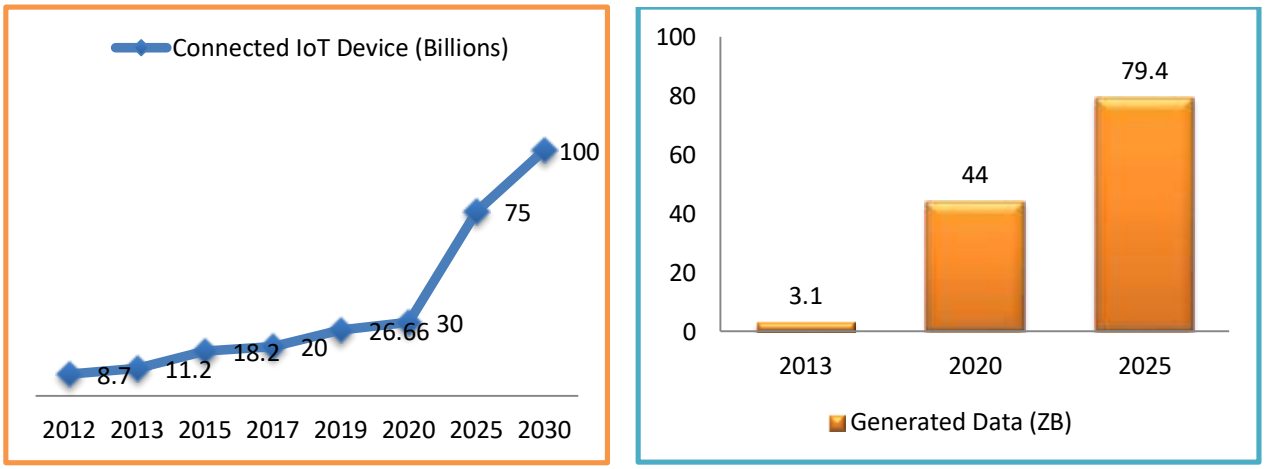
1.1 INTRODUCTION

In this chapter, the introductory study of this thesis is performed. As this thesis is focuses on combining cloud, fog, roof and dew computing with SDN to improve IoT ecosystem's QoS; hence background study on these sectors are performed first. Then the problem statement is presented and based on this, the research objectives are defined. At the end of this chapter, the outline of this thesis is presented.

1.2 BACKGROUND

The Internet of Things (IoT) is composed of devices which access to internet and exchange data among themselves. The goal of IoT is to bring any object online. This can range from something as small as a medicine pill to something as large as a vessel. With the emergence of IoT, our communication capabilities will not be limited to mobile platforms only. Instead, it will enlarge to all of the things we exist side by side. The IoT will allow its connected objects to communicate along other media and with the data produced from each object. The data won't stay raw as it is today. Rather than the data will be personalized to the users according to their necessity and even overlap with diverse data [1]. The IoT devices will overflow the world in near future. Far more devices are migrating to the IoT, and it is now evolving into the Internet of Everything (IoE) [2]. As of 2012, the numbers of IoT devices were had more than tripled than the previous year [3]. There had been approximately 20 billion interconnected IoT devices in 2017, and this amount will rise to approximately 30 billion by the end of 2020, and has more

than double by 2025 [4]. Those device overflows are predicted to generate 79.4 ZB of data in 2025 [5]. This will trigger organizations to re-examine their strategies on data collection, maintenance, and use. The McKinsey Global Institute reports a financial impact of as much as $11.1 trillion per annum in 2025 for IoT systems [6]. Based on the study and statistics from [2-5], the number of IoT devices and the data generated by these devices is presented in Figure 1.1. As massive amount of devices are connected to IoT and associated massive data is generated, therefore it takes advance planning and preparation to cope with this rapid growth of IoT device and associated data.



(a) (b)

Figure 1.1: (a) Connected IoT devices worldwide and (b) Data generated by them

Recent and upcoming communication technologies (e.g. Li-Fi, 5G) and recent innovations in artificial intelligence (e.g. machine learning, deep learning) will power the future IoT [7]. In recent IoT implementation process, developers and integrators will develop and deploy the whole end-to-end software, hardware, and communications system to deliver a smart application. But when tens of billions of devices are connected in coming years, sensing and actuation systems would already be in position. In this scenario, the use of current legacy sensing and actuation system requires new programs to standardize application deployment, additional features, and reduced costs. This scenario will create a complicated ecosystem of elements in software, hardware, and communication which can be termed as IoT ecosystem.

An IoT ecosystem is a composite of different IoT layers that start from the user layer to the connectivity layer. An ecosystem basically arises around a core, which narrates some assets usually used by elements of the ecosystem. The core concentrates on [8]: interlinked devices and access points (gateways), interconnection among devices to the internet, communication standards and protocols, services placed on top of such communication with the support of familiar software platforms and standards, and so many more. However managing the IoT ecosystem is a challenging task because of many reasons. The major issues related to IoT ecosystem is presented in Table 1.1.

Table 1.1: Issues Related to IoT Ecosystem

Manual Maintenance	Heterogeneity	Harmonization	Power Managment
Reliability	Privacy & Security	Data Management	Device Management
Compatibility	Handling Unstructured Data	Data Capturing Capabilities	Regulatory and Legal Issues
Determinism of the network	Lack of a Common Architecture and Standardization	Scalability	Identification and Authentication of Technologies
Integration of IoT Products with IoT Platforms	Limitations of the Available Sensors	Bandwidth	Intelligent Analytics

Computing paradigms are techniques that can greatly minimize user interference in the governance of IoT ecosystems and strengthen IoT systems QoS. First such computing paradigm to enhance the QoS of the IoT system was Cloud Computing (CC). Today, most of the IoT ecosystems depend on centralized client-server framework, communicating to CC through the internet [9]. The IoT devices encounter some issues (presented in Table 1.1). CC including a vast network with infinite storage and processing capacity, offers a flexible, sophisticated environment that allows for progressive integration of data from different sources. Thereby CC successfully mitigated most of those issues with the IoT ecosystem. But we are living in a world of mass connectivity. Even the most sophisticated cloud computing infrastructure continues to struggle with this huge amount of data. Therefore, there are grossly overblown problems

regarding the current IoT with CC. In addition, because of the intensely distributed and dynamic nature of IoT devices, such a centralized cloud deployment is not appropriate. Such a centralized model is prone to blockages, downtime and synchronized attacks which could impact the overall network activities. The upcoming IoT systems therefore have to be distributed and decentralized.

To compensate CC's difficulties, Cisco initially introduced a dependable and reliable solution through Fog Computing (FC) to bring the cloud resources and services near to users. Similar to CC, FC provides users with data, computation, storing, and application services, but comparatively much nearer to users. This enables the utilization of available infrastructure and services in the edge networks. Unlike cloud, fog gives quicker reaction and higher quality services. Therefore, fog can be marked as the ideal option for enabling the IoT to deliver feasible and stable services (real-time services) to plenty of IoT users in an IoT ecosystem [10]. Fog is distinguishable from cloud by its closeness to end-users, its concentrated geographical distribution and mobility support [11]. Fog not just enhances QoS for a large number of IoT applications but can also narrowing greatly the usage of backbone network bandwidth. Consequently, the shortened service costs will favor the users [12]. It can be seen that a major portion of CC challenges is solved by the FC advantageous features. This means the CC limitations and FC advantages are compatible. Table 1.2 presents this relationship.

Table1.2: Compatible Relationship among CC Limitations and FC Advantages

Cloud Limitations	**FC Advantages**
Centralized Nature	Distributed and Decentralized
Lack of Heterogeneity	Supports Heterogeneity
High Latency	Low Latency
Lack of Position Awareness	Provides Position Awareness
Lack of Mobility Support	Provides Mobility Support
No “On-Spot” Processing	Performs “On-Spot” Processing
Lack of Security and Privacy	Enhanced Security and Privacy
No Real-Time Interaction	Delivers Real-Time Interaction

Fog networking principally uses regional resources instead of using outer resources. This leads to a reduction in latency problems and performance [13]. FC's awareness aspect transforms IoT devices from inactive to savvy active devices which can respond without depending on a centralized entity's decision [14]. While, in the market, FC holds a tough competition, there are still several obstacles that need to be addressed to fit FC in an IoT ecosystem. Two emerging technologies can be considered to solve and mitigate FC challenges. These are Roof Computing (RC) and Software Defined Networking (SDN). Table 1.3 explains which FC challenges these two technologies (RC & SDN) can solve, and how.

Table 1.3: Fog Challenges and Suggested Technologies to Mitigate those Challenges

Fog Challenge	**Technology that could Helps to Mitigate FC Challenges**	**Related Work that Supports the Comments of Beside Column**
Container Management	SDN (Through Resource Allocation & Service Orchestration Feature)	[15-18]
Strict Support	SDN	[19-20]
Package Portability	SDN (Through Service orchestration feature)	[21]
Rapid and Dense Migration Support	SDN (Through Routing & Traffic Management Feature)	[22-24]
Heterogeneity	Roof (Through Harmonization Feature)	
Load Management	SDN (Through Load Balancer Feature)	[25-26]
Data Analytics	Roof (By Early Preprocessing & Context Building)	
System Benchmarking	SDN (Through Service orchestration feature)	[27-28]
Network Management	SDN (Through Mobility, Routing & Service orchestration feature)	[29-32]
Security	Roof (Through Contextual-Awareness Feature)	

Energy Consumption	Roof (Since RC releases less Data to FC, Hence Less Processing in FC. So Less Power Consumption)	
Context Awareness	Roof (Through Building Real-Time Context)	
User Mobility	SDN (Through Mobility & Handover Feature)	[33-35]

RC is originated to reduce the CC and FC limitations. RC's objective is to reduce the usage of cloud or fog assets by processing the repeatedly accessed data at the brink of the user's existing network (e.g., router, access point, server, or related computing platform as suitable). This increases security and performance of the entire network. Key elements of security and privacy (authorization, authentication, encryption, network security etc.) of RC can be accomplished by context awareness. Unlink FC, RC offers more effective approach at the brink of the network in terms of efficiency in data processing and analysis. Although RC is beneficial for IoT ecosystem to handle Big Data but it struggles from some troubles. These are: programmability, identifying, standardization, abstraction of data, management of services, battery capacity and life and consumer costs.

Researchers have presented a new computational approach called dew computing (DC) in this context to cope with billions of devices running at the edge of the consumer with limited affection for internetwork. DC introduces huge changes to the framework of internet computing, since it can offer real-time as well as other services without internet access. In the classic CC/FC/RC computing domain the consumer is incapable to obtain their data if the internet access to the data centers is lost. DC is trying to solve this, i.e., dependency on internet. But DC will encounter a series of technical issues, including energy management issues, processor efficiency, and file storage issues. Therefore DC alone can’t deliver much service to an IoT ecosystem. Furthermore, the DC paradigm will involve additional architectural and programming methods to minimize complexity and enhance the effectiveness and functionality of optimized distributed computing.

Managing QoS is one of the critical networking issues that have not yet come up with an acceptable solution. There are many IoT devices, and they have growing functions in our everyday lives. Thus IoT ecosystems QoS expectations can differ from one application to another. Simply QoS means network connection performance. QoS assists to control the device functionality and resources needed to deliver IoT services [37]. QoS parameters can help consumers figure out the ideal IoT solution for their use, and also discuss service quality optimization. QoS essentially has a lot of parameters in it. The key parameters relevant to smart devices and physical sensors are: (i) Quality (ii) Energy consumption (iii) Latency (iv) Monetary Cost (v) Network Usage (vi) Bandwidth (vii) Reliability (viii) Mobility Support (ix) System Lifetime (x) Delay and Delay Variation and (xi) Data volume.

The boundaries of traditional networking in terms of QoS have mitigated through SDN. SDN is an evolving concept of computing and networking which is a network virtualization approach of deployment. SDN splits control plane and data plane to allow for robust network traffic management. The efficiency of a system's data plane relies on instructions coming from the centralized controller's control plane. Separation of planes enables control and monitoring of features and functions at the lower part remotely. Compared with traditional networks, the SDN has abundant benefits. These can be [38]: Breakdown of data plane and control plane, Programmability, Easier network management and addition of new features, Ubiquitous perspective of SDN controller, Can quickly add and remove forwarding devices, More granular security, Lower operating cost, Greater speed and agility, Hardware savings and reduced capital expenditure, Cloud Abstraction , and Guaranteed content delivery.

1.3 PROBLEM STATEMENT

The early IoT ecosystem was integrated with CC paradigm to acquire necessary QoS. But because of its limitations, another computing paradigm named FC emerges. FC resolved nearly all the limitations that cloud experienced. But when fog and cloud integrated together to improve QoS of IoT ecosystem then the traditional network comes as a major blockage issue. Therefore a modern networking strategy comes into existence, namely SDN to overcome the issue. The service orchestration cites a key challenge assigned by the notion of fog and cloud integrated architecture. The orchestration involves the automated set-up, counterpart and passage of service

instances with an extensive range of capabilities on fog nodes in a broad volume. SDN can be used to achieve this, instead of traditional networks. By collaborating SDN with FC and CC integrated architecture, the IoT QoS issues can be solved and can be enhanced where appropriate.

The study from “Literature Review” related to “Method I” indicates that FC and SDN have conducted some work in the area of QoS improvement. But these experiments were performed exclusively on the basis of either FC or SDN. Some present techniques for boosting QoS utilizing FC only, and some other present approaches utilizing SDN. By combining SDN and FC, these methods do not explain the overall QoS improvement scheme. Therefore, an Method Is necessary to show how FC and CC can be integrated and how SDN and fog and cloud integrated architecture can be properly merged to recover the deficiencies of each other and how they're being incorporated to maximize the IoT system's QoS.

But as the number of IoT devices keeps increasing significantly, the combined fog and cloud system has fails to meet the services needed for modern IoT systems. A new computing model, namely RC, thus arises to solve these drawbacks. With the support of SDN, RC can solve numerous problems facing FC. That means when cloud, fog and roof computing paradigms are integrated and this integrated architecture is managed by SDN; then it can provide much better QoS to the IoT ecosystem. The cloud, fog, and roof, however, all need internet connectivity to perform the operation they need. DC attempts to solve this issue. This means DC can be included in the integrated cloud, fog and roof architecture. That makes four computing paradigms (CC, FC, RC and DC) are integrated together to improve IoT ecosystem’s QoS. It can thus be stated that the new computing paradigm eliminates the shortcomings of previous paradigms. They are compatible paradigms that need to be used together instead of being replacements for one another. Once introducing such ecosystem (i.e. combining four computing paradigms), several technological problems of the IoT system could be solved. But as a major technical challenge, there will always be some issues and these will be very complex. Service Orchestration, Heterogeneity, Smooth Service Delivery, Service-Centric Structure, Inter-operability and Soft State may be the triggers why this complication occurs. Even the small list of above indicates that there exists a multiple disciplines technical problem that is impossible to handle by utilizing the traditional networking system. To make appropriate use of combined cloud, fog,

roof and dew computing architecture, a strategy needs to be stepped up that covers all the inner complexity of users especially software developers and service providers. To this end, SDN, along with its network programming capabilities, is emerging as an acceptable solution to orchestrate the system, application services and connected gadgets by hiding the challenges of this interconnected ecosystem from end users. Once all the complexities and requirements that have been listed above are kept in mind, it can be seen that SDN is the right choice for eliminating the limitations of the proposed integrated multi-tier computing paradigms. SDN can efficiently improve network management, identify network flows, extend network capacity and promote network virtualization. The SDN concept focuses the network intellect on the centralized software-based controller. So it eases the edge devices remarkably from carrying out complicated networking activities like service detection and orchestration.

When bring all of these innovations and developments combined, this can be seen that a multi-tiered architecture that incorporates multiple computing paradigms (i.e. dew, roof, fog and cloud computing) controlled by SDN has considerable potential to reduce the barriers and constraints that IoT ecosystem experience. The key concept behind these technologies is to move the power from a centralized body to the edge of the network to gain maximum efficiency for each layer. It has the potential to fulfill the QoS requirements and maximize consumer satisfaction. Table 1.4 demonstrates growing obstacles in IoT ecosystem (presented in Table 1.1) and by which entities of the proposed methods can solve those obstacles.

Table 1.4: IoT Ecosystem Challenges and Solution Provided by Proposed Methods

IoT Ecosystem Challenge	**Probable Solution (Provided by Proposed Method)**
Data Management	✓ Roof (Context Building & Extract Meaningful Data) ✓ SDN (Keeping Track of Data) ✓ Fog (Local Data Storage) ✓ Cloud (Permanent Data Storage)
Power Management	✓ Dew & Roof (As perform data preprocessing or processing at edge of the network, hence less data is sent to upper layer. Thereby using less power. Because processing requires less power compared to transmit for

	same amount of data) ✓ SDN (Keeping track of devices battery life. Hence appropriate decision is made based of battery power)
Manual Maintenance	✓ SDN (Dynamic & Programmability feature removes manual maintenance and configuration)
Heterogeneity	✓ Roof (Perform things/dew devices heterogeneity) ✓ SDN (Perform network devices heterogeneity)
Harmonization	✓ Roof (Make common platform to provide services)
Device Management	✓ SDN (Through Remote updating & Diagnostics)
Reliability	✓ Dew (Provide offline service)
Privacy & Security	✓ Roof (Through Contextual-awareness)

1.4 RESEARCH OBJECTIVES

Here the novelty of this thesis work is elaborately explained. The study shows that appropriate approaches are required to improve the QoS of IoT ecosystem. Hence we have presented two novel methods which effectively increase the QoS. Firstly, the study shows that a method is necessary to show how SDN, fog and cloud can be integrated recover the deficiencies of each other and how they're being incorporated to maximize the IoT system's QoS. Therefore we have provided “Method I” which presents an architecture and algorithm to improve the IoT ecosystem’s QoS. The architecture combines cloud, fog and SDN. The algorithm is presented to show the actual procedure of the proposed architecture to improve the QoS. The presented architecture in “Method I” is evaluated via iFogSim simulator. The result shows promising result in favor of our proposed approach. Form the survey on “Literature Review”, this can be seen that there is no such architecture which combines SDN with fog and cloud computing and then simulating this with iFogSim. This shows the novelty of our proposed “Method I”.

Secondly it has been shown that a multi-tiered structure that combines SDN-controlled dew, roof, fog and cloud has tremendous opportunity to decrease the barriers and challenges faced by today’s modern IoT ecosystem. Therefore in “Method II” we have presented “Software-Defined Dew, Roof, Fog and Cloud (SD-DRFC)” framework for IoT ecosystem to boost the system's

QoS. From “Literature Review” it can be seen that there is no framework or related research that incorporates dew, roof, fog and cloud architecture where this combined architecture is regulated by SDN. In addition there is no simulation performance on that kind of frameworks/architectures. But in this work we have presented the framework combining dew, roof, fog and cloud computing which is assisted by SDN. Then the performance and efficiency of the framework is evaluated via iFogSim simulator. Results show improved QoS in favor of our “Method II”. This further added the novelty to this work. To summarize, this thesis work’s actual **objectives** are as follows:

I. An illustrative survey of background and related work is performed on cloud, fog, roof and dew computing and their interaction with SDN. This survey will enable the understanding of the journey of multi-tiered architecture that incorporates multiple computing paradigms which are controlled by SDN.
II. Two novel methods are presented based on the concept of multi-tiered architecture to improve the QoS of IoT ecosystem.
III. Method I present an architecture that combines Cloud and Fog which are managed by SDN. The architecture is described and illustrates how the upper layer is linked to the underlying layer and how the SDN controller is capable of controlling the overall system. An algorithm (On the basis of SDN network partition) is suggested to assist the architecture for optimizing access point selection and putting data in the optimal layer for processing.
IV. For simulation, a use case on the grounds of the presented architecture of “Method I” is described. Next, through the iFogSim Simulator, the use case is deployed and tested. The findings indicate a substantial improvement in QoS in favor of the proposed policy.
V. Method II proposes a novel SD-DRFC framework architecture to advance QoS of IoT ecosystem. This framework's novelty stands in its ability to deliver services from the nearest possible location, manage data routing pathways individually for diverse data types, efficiently balance enormous data traffic on network, offer limited services without internet connection, and appropriately distribute network and computing resources.
VI. A use case dependent on the proposed framework architecture of “Method II” is provided which is then evaluated using the iFogSim simulator. When compared the findings of the

simulation; the proposed framework execution performs much better than traditional system.

1.5 SCOPE OF THE STUDY

The required tools to carry out this work are:

- Eclipse Integrated Development Environment (IDE)
- Java Development Kit (JDK)
- iFogSim Simulator
- Inkscape Vector Graphics Editor

1.6 ORGANIZATION OF THE THESIS

This thesis consists of 6 chapters and these chapters are organized as follows:

Chapter 1 – Introduction: Performs the background study of IoT Ecosystem, CC, FC, RC, DC, QoS, and SDN. In addition presents the problem statement, objectives and scope of this thesis.

Chapter 2 - Literature Review: An illustrative review of related work is performed on IoT ecosystem, CC, FC, RC, DC and SDN. Initially the review is performed concerning Method I and then review concerning Method II is carried out.

Chapter 3 – Methodology: Presents two approaches based on the concept of multi-tiered architecture to improve QoS of IoT ecosystem. Method I presents an architecture incorporating CC, FC and SDN and an algorithm to assist the architecture. Method II presents a novel SD-DRFC framework architecture for denser IoT ecosystems.

Chapter 4 – Simulation Setup: Firstly presents the simulation setup on Method I. Next the simulation setup on Method II is presented. Both of this simulation setup is carried out on iFogSim simulator.

Chapter 5 - Results and Discussions: The simulation result of Method I is presents here and then the result is discussed to evaluate the efficiency of the presented approach. The same procedure is followed for Method II.

Chapter 6 – Conclusions: This chapter describes conclusion and future research scope.

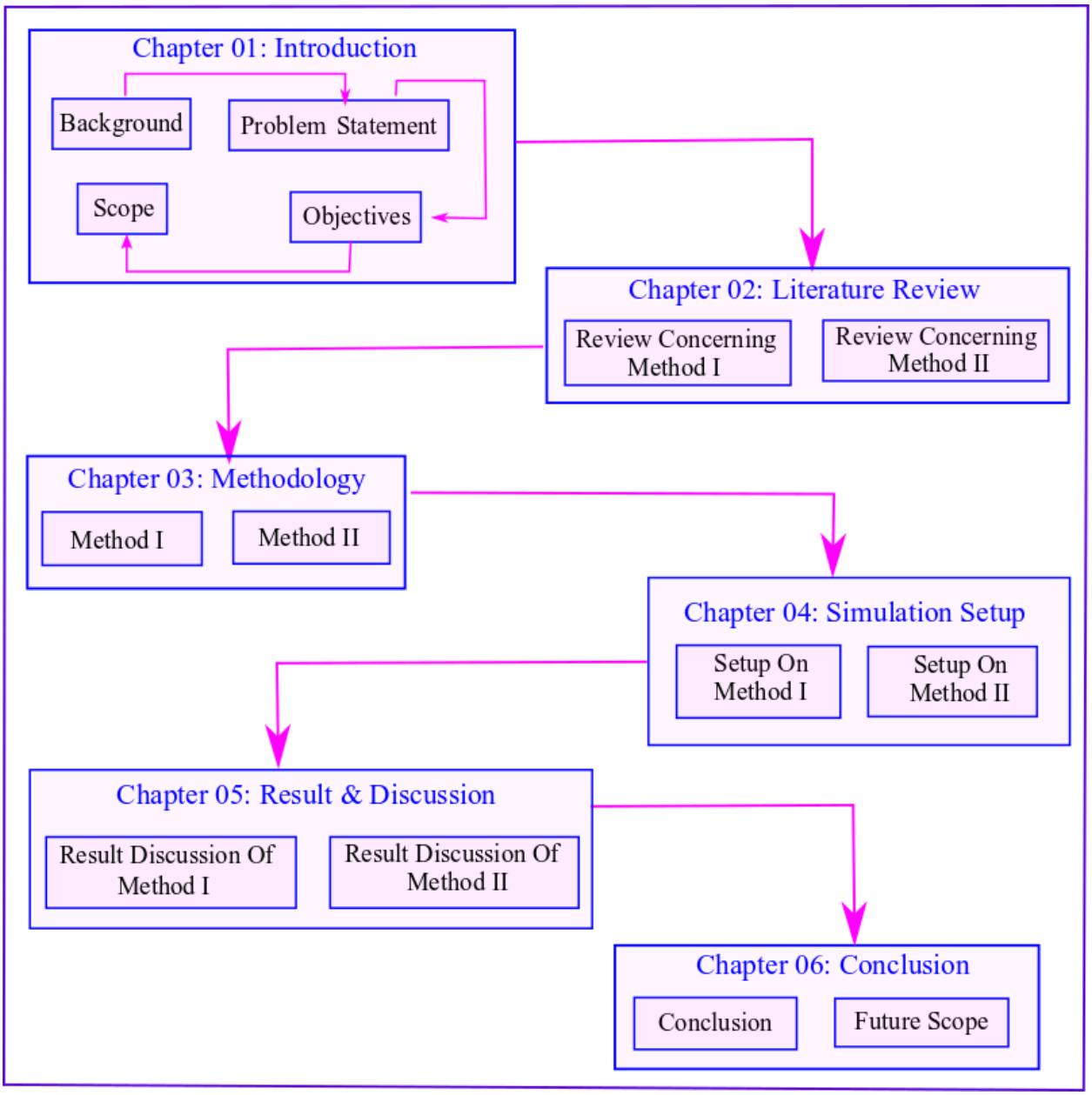


Figure 1.2: Organization of the Thesis

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

In this thesis, we have presented two methods to improve IoT ecosystem's QoS. Therefore in this chapter we first perform the literature review concerning the first method (Method I). Then we perform the literature review which concerns our second method (Method II).

2.2 REVIEW CONCERNING COMBINED FOG AND SDN (METHOD I)

The basic study on IoT, cloud and fog and their interrelationship is performed in [13, 39-40]. In [40], in the sense of big data on fog networking, the authors identified the fog issues and problems. By illustrating the advantages and possible barriers, the work [13] introduces the state-of-the-art of FC and its incorporation with the IoT. The focus of this analysis will also be on the fog architecture and current IoT technologies that will be enhanced by utilizing the fog model. Finally, there is discussion of open problems and possible research directions about FC and the IoT. One of the major issues with fog services is QoS. The work [39] thus offers a cloud computing survey, addresses the QoS metrics, and the possible FC research directions. Kumari A. et al. [41] provides data analytics mechanism for the fog. The authors discuss the unique characteristics of the Fog Data Analytics (FDA) and its difficulties. As indicated by the authors survey results, powerful and effective data storage & data processing algorithms are essential for correct analytics as we step along with FC. An IoT data placement idea called iFogStor was proposed by Naas M. et al. [42]. The objective of iFogStor is to take the benefits of fog node's heterogeneity and location to reduce the overall latency of fog data storage and extraction. They provided a method to test their approach based on iFogSim. The test indicates, iFogStor clearly shows promising results.

The interrelations among FC and the cloud must be understood and the impact of FC on the QoS of the IoT service must be evaluated. Redowan Mahmud et al., [43] bring in a latency-aware application module management framework for the fog environment that fits the complex latency of the service delivery and the quantity of data signals to be processed in different applications per unit of time. The strategy aims at ensuring QoS implementations in meeting service delivery deadlines and maximizing the use of resource in fog environment. A general framework for IoT-fog-cloud architectures and a fog-capable delay-minimizing technique is implemented to reduce service delays for IoT implementations is presented by Yousefpour, A. et al. [44]. They then develop an analytical model and show how the designed methodology helps to reduce the delay in the IoT system in order to validate their policy. In next work, the authors introduce FOGPLAN, a QoS-aware Dynamic Fog Service Provisioning (QDFSP) system, in their work [45]. QDFSP is concerned with the dynamic implementation of application services on fog nodes, in order to satisfy application criteria for low latency and QoS while reducing costs. Debabrata Sarddar et al. [46] proposed QoS-aware resource management technique for efficient resource management which envisages accomplishment time, network use and energy consumption as parameters of QoS. The efficiency of the mentioned technique was tested and the experimental consequences indicate that as for QoS parameters, the mentioned technique performs better. It has seen that SDN does not include in the above mentioned frameworks [41-46]. Considering SDN instead of conventional network would be much more productive for these frameworks. The work [47] presents an analysis of the convergence of cloud & FC for perspective on IoT platforms. The authors presented the idea of FC, which acts as a proactive for data resources that are time sensitive. Furthermore the paper contains a brief explanation about SDN and its function in FC.

SDN steps in to operate and orchestrate IoT systems and its interconnection with cloud and FC domain. SDN also mitigates the shortcoming that rises when IoT is collaborated with both of the cloud and FC domains. A number of works on collaborating SDN with IoT have already been performed. In [48], discussion on some methods for different domains for the merged SDN-based IoT and recent developments in research are described. A comprehensive survey of numerous SDN-based technologies from different network aspects (e.g. edge, entrance, center, and data center networking) that are advantageous for fulfilling IoT requirements is performed in [49]. In [50], a critical view of IoT and SDN techniques, current research strategies and future

contributing parameters is provided. QoS management in SDN networks based on Multi Path Routing (MPR) called MPRSDN is approached in [51]. The proposed approach abdicates resource preservation in assistance of their proficient utilization and thereby implements a best effort approach. Wang, J., & Li, D. [52] present an Adaptive Computing Optimization technique in SDN-Based Industrial IoT with FC. The authors briefly focus on the problems (regional computing, fog or cloud computing) inside a combined computing system. A system based on Computing Mode Selection (CMS) and task priority based redaction sequences (ASTP) is proposed to ensure QoS.

Optimal SDN controller selection is one of critical task in SDN deployed networking systems. As a single SDN controller can't provide optimum service to a wide area; therefore multi-SDN controller can be placed to solve the issue. A compression study for multi-controller analysis in SDN is presented in [53]. Initially, the multi-controller overall view, including the roots of the multi-controller as well as its obstacles, is introduced. Then, according to the method of deploying the multi-controller, the work categorizes multi-controller study into four categories (scalability, consistency, reliability, load balancing). Finally, the authors suggest some important research concerns to resolve in the ahead and complete the multi-controller study. The strategy propounded in [54] considers an optimum progressive SDN controller placement that decreases latency of controllers-switches, latency of inter-controllers and weight of controllers. A distributed policy builds on non-zero-sum game [55] is propounded to deploy the multi-controller to maximum effect. Every controller has an apparatus optimization with that policy, which enumerates a payoff duty and performs comparison to its personal payoff grade to accumulate costs and prosper QoS by optimizing controller locations. The paper [56] suggested an optimal choice of controllers for SDNs to increase QoS. Here, a two-step Method Is introduced for the selection of optimal SDN controllers. Zhao, Z., & Wu, B. [57] proposed Scalable SDN architecture with distributed placement of controllers for WAN. The authors suggest distributed positioning of controllers in WAN to create a scalable SDN architecture, taking into account demand flow, traffic latency and load balancing. An algorithmic task allocation approach and a Genetic Algorithm Based Virtual Machine Placement as answers for the task allocation and virtual machine implementation problem models are presented in [58]. The tests are performed and the outcomes exhibit that the solutions propounded enhance QoS in the cloud/FC environment with regards to the allocation cost.

While computing domain is consider, SDN was initially merged with cloud. Elaborate review that combines SDN with cloud is introduced in [59]. This study focuses on maximizing the resources of data centers, traffic engineering, network virtualization and privacy. The study also offers various simulation and evidence assessment methods designed for SDN-enabled clouds. In [60], an innovative SDN definition that can be used in future developments of cloud networking is implemented. In [61], an advanced SDN-based cloud infrastructure called Software Defined Cloud (SDCloud) integrating several software-defined cloud services to manage complexities associated with cloud platforms is introduced. The authors set up an SDN-based cloud computing environment in [62] through the OpenFLow open source switch and controller packages. The experimental findings indicate that SDN-based environments can deliver cloud services assured by QoS.

As the cloud-SDN faces challenges in supporting the latest IoT systems, fog must therefore be merged with the cloud-SDN combination. Salman, O. et al. [63] conducts an IoT analysis from SDN & fog point of view. They critically analyze the approaches focused on SDN and FC to solve the key challenges of the IoT, illustrate its benefits and reveal its limitations. In order to achieve goals (standardization, management of QoS and flexible adaptability), Gupta, H. et al., [64] discussed limitations and proposed a service-oriented middleware to leverage cloud and FC integration alongside SDN. The proposed model, known as Software Defined Fog (SDFog), brings entities together as tools and allows end-to-end QoS requirements for implementations to orchestrate such services. The IoT-Fog based SDN activated system structure is shown in [65]. A model for IoT system architecture is given in this report which utilizes an edge computing level of fog nodes. But this study has the lack of clear insight into the layout of the system. However the research [66] titled “Software-Defined Fog Network Architecture for IoT” overcomes this lacking. This work offers thorough insight into the structure of the system and the features of its key components. The advantages of the proposed design as well as its possible facilities are also addressed. Such possible service measures essentially by matrices called QoS.

Most of the fog structure of today usually depends on the wireless network. In [67], a study that uses SDN for wireless fog network control is identified. The authors present an SDN permitted wireless fog architecture that incorporates both OpenFlow and decentralized wireless protocols. By enabling customizable fog routers, the solutions offer reduced latency and efficient load

balancing to release the network load. But the objective of this work is not wireless network QoS. A study focusing on wireless network QoS is done in [68]. This study proposes a system for software-defined QoS providing advanced wireless sensor networks (WSNs) for fog. The word mobility needs to be addressed when talking about wireless networks. In [69], the authors implement robust SDN-based fog architecture by splitting mobility management and data routing. Under their architecture, they build efficient signaling mechanisms to provide smooth and reliable mobility support to mobile users, and deliver an optimized route optimization algorithm by considering the efficiency gain in mobile fog data and network overhead transmission. The suggested technique also dramatically improves the handover efficiency and delivers higher data exchange capacity in the mobile fog environment.

Now the novelty of this thesis's first approach compared to the above related previous works is discussed below and the summary is presented in Table 2.1. Literature reported in [47], [53-56], [61-62], [64-66] & [68] provides policy to improve QoS. But these policies don't provide any specific simulation use case. In addition neither any simulation is performed on them nor is any descriptive review on simulation is provided. To mitigate these limitations, in this thesis's work; a specific use case is provided which is also simulated using iFogSim. SDN neither included in proposed policies nor description is provided in [42-46]. Without SDN, today's cloud and FC can't interoperate efficiently. That's why to solve such issue this thesis has included SDN in its work. The works from [53-56] & [68] provides knowledge on SDN controller placement. But these policies don't perform simulation and doesn't provide any simulated use case. Limited work is performed on the term mobility. The work [69] includes the detail study on mobility. But this work doesn't include the study on SDN controller placement. Therefore a single work is most needed which perform the detail study on both SDN controller placement and mobility and also provides a policy to improve QoS combining them. This thesis's first approach performed that study. The study [42-47], [52-56] & [64-69] considers the term QoS and its parameter. But among these; [64-66] & [68] is only discussed about the term QoS. No specific QoS parameter is designed or evaluated. The work [42-44] considers only one QoS parameter. The work [47], [52], [67] considers two and the work [46] considers three QoS parameters. There are a lot more parameters which was either ignored or unanswered by these works. By considering limited number parameter, the overall QoS for an IoT system can't be justified. For this purpose 4 QoS parameters are considered and evaluated in this thesis's work. The summary of the above

mentioned previous works and the novelty of this paper compared to those previous works are shown in Table 2.1.

Table 2.1: Summary of Literature Concerning Method I

Work	**Sect-ors Incl-uded**	**Back-ground study**	**Propose policy for QoS**	**Policy inclu-des SDN**	**Optimal Con-troller Place-ment**	**In depth mob-ility study**	**Perf-orm Simu-lation**	**Simu-lation use case**	**Consi-dered QoS para-meters**
[13, 39-40]	IoT, Cloud & Fog	✓	×	×	×	×	×	×	×
[42-44]	IoT, Cloud & Fog	✓	✓	×	×	×	✓	×	1 (Latency)
[45]	IoT, Cloud & Fog	✓	✓	×	×	×	✓	×	2 (Latency & Service Cost)
[46]	IoT, Cloud & Fog	✓	✓	×	×	×	✓	×	3 (Time, Network usage & Energy Consu-mption)
[47]	IoT, Cloud &	✓	✓	✓	×	×	×	×	2 (Bandwid th usage

	Fog								& Energy Consu-mption)
[48-50]	IoT & SDN	✓	×	×	×	×	×	×	×
[52]	IoT, SDN, Cloud & Fog	✓	✓	✓	×	×	✓	×	2 (Latency & Relia-bility)
[53-56]	SDN	✓	✓	✓	✓	×	×	×	"QoS" term Dis-cussed
[59-60]	SDN & Cloud	✓	×	×	×	×	×	×	×
[61-62]	IoT, SDN & Cloud	✓	✓	✓	×	×	×	×	×
[63]	IoT, SDN, Cloud & Fog	✓	×	×	×	×	×	×	×
[64-66]	IoT, SDN, Cloud & Fog	✓	✓	✓	×	×	×	×	"QoS" term Dis-cussed
[67]	IoT,	✓	✓	✓	✓	×	✓	×	2

	SDN, Cloud & Fog								(Latency & Load Balan-cing)
[68]	SDN	✓	✓	✓	✓	×	×	×	"QoS" term Dis-cussed
[69]	SDN, Fog & Cloud	✓	✓	✓	×	✓	✓	×	2 (Latency & System Cost)
This Thes-is's Met-hod I	IoT, SDN, Cloud & Fog	✓	✓	✓	✓	✓	✓	✓	4 (Net-work usage, Cost, Latency & Power Consu-mption)

2.3 REVIEW CONCERNING SD-DRFC FRAMEWORK (METHOD II)

This thesis's second method provides a framework for IoT ecosystem combining cloud, fog, roof and dew computing where this multi-computing paradigm architecture is managed by SDN. The proposed method includes the sectors of IoT, cloud, fog, roof, dew and SDN. Therefore survey on these four computing paradigm and their relationship with IoT ecosystem is performed first. Then the study on SDN and its interrelationship with above mentioned computing paradigms is performed.

The basic concept of an IoT ecosystem is delivered by Oleksiy Mazhelis et al. [70]. According to them, knowing the aspect and components of IoT ecosystems is of greatest priority. A survey on components of the IoT ecosystem is conducted on [71]. The authors provide the following core

supporters: devices, gateways, operating systems, middleware, and communication. Many of the most significant developments are illustrated by a unique taxonomy for IoT technologies, as well as few applications which have the possibility to make a significant improvement in living beings are assessed in [72]. An image of the key technical elements required to allow interconnection between things for the realization of IoT theory and application is given in [73]. Survey associated with IoT ecosystem specifications is conducted in [74] in order to provide highly incorporated and context-aware rational IoT services. If we switch to particular purposes then research on communication elements for IoT-based developments in smart homes is available in [75], IoT-enabled smart-socket specifications in [76]. A thorough review on state-of-the-art solutions to facilitate interoperability among multiple IoT platforms as well as the critical problems in this subject is presented in [77]. From the perspective of security and privacy, the IoT ecosystem survey is conducted in [78-81]. Authors from [78] present an exploration of the current IoT security research within 2016-2018, its developments and open questions. Authors in [79] describe the open problems in the privacy protecting approaches when conducted in IoT systems. Review of various IoT security environment and outlined the lack of a security tier in all of these models are discussed in [80]. Survey has been conducted in [81] to examine subsisted protocols and frameworks for securing communication in the IoT along with research challenges.

Table 2.2: Notable works of CC and Aspects Covered by these Works

Notable Work	**Incorporates IoT**	**Fundamental Study**	**Architecture**	**Challen-ges**	**Future Research Direction**	**Benefits/ Advan-tages**
[82]	✓	✓	×	✓	✓	×
[83]	✓	×	✓	✓	✓	×
[84]	✓	✓	×	×	×	✓
[85]	✓	✓	✓	✓	✓	✓
[86]	✓	×	×	✓	✓	✓
[87]	✓	✓	✓	✓	✓	✓
[88]	✓	✓	✓	✓	✓	×

A study on the incorporation of cloud and IoT ecosystem is undertaken in [82]. Starting with an overview of the fundamentals of both IoT and cloud, authors explore their complementarities, explaining what is influencing their convergence at present. This also offers an up-to-date image of cloud-IoT implementations in literature, concentrating on specific challenging issues. The analysis of open issues and potential solutions for Future Internet, which incorporates IoT and cloud, is also performed in [83]. The current trends, developments and open issues are discussed in [84]. The open issues and challenges regarding cloud integration with IoT could also be found in [85]. The challenges of bridging the Cloud and IoT ecosystem are discussed in depth in [86]. Some notable works which integrates cloud and IoT is shown in Table 2.2. The aspects covered by these works are highlighted in the Table 2.2.

Studying open issues and problems in [82-86]; it is understandable that cloud wouldn't be enough for today's IoT ecosystem. Therefore fog computing incorporates with cloud to built ideal architecture for IoT ecosystem. FC and its role in the IoT are discussed in [89]. As per authors, due to certain features fog is the ideal platform for a range of essential IoT applications & services. The study [90] presents an analysis of the convergence of cloud & fog computing for perspective on IoT platforms. The initial phase for IoT ecosystem data management from cloud to fog is set in [91]. The work from [41] provides data analytics mechanism for the fog. The authors discuss the unique characteristics of the Fog Data Analytics (FDA) and its difficulties. As indicated by the authors survey results, powerful and effective data storage & data processing algorithms are essential for correct analytics as we step along with FC. A detailed review of the IoT, cloud and fog partnership is reported at [13, 92-94]. These reviews cover open issues, architecture, benefits, implementation challenges, applications and future research directions. The work [95] announces a framework and strategy designed to minimize IoT delays. Within this paper the authors present a general framework for applications in the IoT-fog-cloud. They also suggest a latency-minimizing strategy for fog-capable gadgets aimed at reducing the delay in operation for IoT applications. The detail FC architecture for IoT applications & services, their evaluation, and future research directions about FC architecture is presented in [96-99]. The study [100] briefly analyzes a new phenomenon called Fog of everything (FoE), which is essentially a fusion of two different technical paradigms named fog computing & IoE. Because of their compatible features, however, the authors predict that their

incorporation will promote a variety of computing and network-intensive ubiquitous applications within the Future Internet field. An IoT data placement idea called iFogStor was proposed by Naas M. et al. [42]. The objective of iFogStor is to take the benefits of fog node's heterogeneity and location to reduce the overall latency of fog data storage and extraction. They provided a method to test their approach based on iFogSim. The test indicates, iFogStor clearly shows promising results. There are a number of challenges relating FC which includes general challenges [93-94], data aggregation challenge [101] and security challenges [93, 102-103]. Some notable works in the context of FC and IoT is shown in Table 2.3. The aspects covered by these works are also highlighted in the Table 2.3.

Table 2.3: Notable works of FC and Aspects Covered by these Works

Notable Work	**Incorporates IoT and Cloud**	**Fundamental Study**	**Architecture**	**Challen-ges**	**Future Research Direction**	**Benefits/ Advan-tages**
[13]	Only IoT	✓	✓	✓	✓	✓
[89]	Only IoT	✓	×	×	×	✓
[91]	Both	✓	×	✓	×	✓
[92]	Both	✓	✓	✓	×	✓
[93]	Both	✓	✓	✓	✓	×
[96]	Only IoT	✓	✓	×	✓	✓
[97]	Only IoT	✓	✓	×	×	✓
[98]	Both	✓	✓	×	✓	×

RC has emerged to mitigate the challenges presented by FC. The basic concept of roof, its roots, connection with IoT and how roof bridges the lacking gap among cloud, fog and IoT is explained briefly in [104]. The basic features and roof objectives are given here as well. Anatol Badach in [105] sets out the fundamental distinction among cloud, fog, and roof computing. He also offers multiple layer architecture by integrating those computing paradigms. The IEEE 1931.1 Working Group is recently employing on the definition of an Internet standard for the technical and functional interoperability of federated roof IoT systems operating and functioning in a reliable,

semi-autonomous, and decentralized way. Details of the project are presented in [106]. The usage of blockchain in IoT, the architectural template and the usage of blockchain in the IEEE 1931.1 standard is explored in [107].The authors tried to bring about benefits, criticalities, and subsequent evaluation that are essential in terms of blockchain for a productive implementation within RC. Syam Madanapalli presents the impact of roofing on IoT and its big data in [108].Only few efforts have been reported in the roof domain, and they are already shown in [104-108]. From these; we see that roofing suffers from programmability difficulties, absence of internetwork-free service, standardization, point-of-call service, abstraction of data, shortage of intense scalability service, etc.

To remove roof and its upper computing layers (fog, cloud) challenges and also remove the necessity of always internet connection; DC steps in. WANG et al.'s work [109] focuses on the evolution of dew computing, particularly its roots, research status, evolution status, and its effects on Internet computing paradigms' transition. The article in [110] takes the perceptual method of organizing the perpendicular hierarchical links among the scalable distributed paradigms of computing: FC, CC and DC. In the obtainable distributed computing hierarchy, the DC is demonstrated and realized as an innovative structural layer. One other work [111] as well addresses dew as a latest edge computing aspect for flexible distributed networks. DC is strongly connected to CC and it's the supplementary part of CC [112]. In [93] the authors claim that DC is a part of CC interdependently. Cloud-dew architecture thus is an integral part of DC study. The architectural works for the cloud-dew are discussed in [113-115]. The architecture in [114] exploring the potential in unstable networks of distributed systems. Some particular dew architecture includes: [116]: focusing on decentralized adaptive latency-aware cloud-edge-dew architecture for unstable network and [117]: focusing on cyber-physical systems (CPS) and IoT DC architecture. The detailed view on DC's role and significance in service and management technology presents in [118]. DC integrates the two significant CC service models notably SaaS and SaaP to deliver effective services, as CC can’t meet recent demand [119].Now if we switch to specific dew-based system services; we see there are only few activities in that sector. Mathias Longo et al [120] make a model for hourly forecasting of power availability that works towards combining mobile devices into DC. Another work [121] provides insights into the role and significance of smart watches in DC. This work demonstrates the research, evaluation, interpretation and discussion of DC environment smart watches. However challenges are needed

to be discussed to provide future research directions in the DC domain. The study [112-113] and [122-123] covers detail research challenges of dew computing. Some notable works in the context of DC with other computing domains and IoT is shown in Table 2.4. The aspects covered by these works are also highlighted in the Table 2.4.

Table 2.4: Notable works of DC and Aspects Covered by these Works

Notable Work	**Incorporates IoT?**	**Incorporating Computing Domain**	**Fundamental Study**	**Architecture**	**Challenges**	**Future Research Direction**	**Benefits/ Advantages**
[109]	×	CC	✓	✓	×	✓	✓
[110]	✓	CC and FC	✓	×	×	×	✓
[111]	✓	FC and Edge Computing	✓	✓	×	×	✓
[112]	×	CC	✓	✓	✓	✓	✓
[113]	✓	CC and FC	✓	✓	✓	×	✓
[114]	×	CC	✓	✓	✓	×	✓
[117]	✓	CC and Edge Computing	✓	✓	×	×	✓
[118]	✓	CC and FC	✓	×	✓	✓	✓
[122]	✓	CC, FC and Edge Computing	✓	✓	✓	✓	✓
[123]	✓	CC, FC and Edge Computing	✓	✓	✓	✓	✓

The IoT ecosystem and its association with computing paradigms are controlled and coordinated by SDN. A variety of research studies have been performed on the integration of SDN with IoT. The work [49] carryies out a thorough study of different SDN-based innovations that are useful for satisfying IoT requirements, from different networking dimensions: edge, entry, core, and data center networking. A detailed view of IoT and SDN methods, current research activities and future major contributors is performed in [50]. The IoT ecosystem structure based on SDN can be seen in the work of [124]. However, IoT deployment with SDN will raise several difficulties. In [49], these research problems and future research perspectives can be found. SDN had first been integrated with the cloud to manage an IoT ecosystem, while having the computing paradigm into consideration. The work of [59] introduces a detailed examination of the research of the combined SDN with CC. The work [61] performs a revolutionary software-based cloud management structure termed Software-Defined Cloud (SDCloud) which integrates different software-defined cloud services to handle problems associated with CC systems. As the SDN cloud faces difficulties in supporting the new IoT ecosystem, FC must therefore be integrated with the cloud-SDN pair. The study [65] shows the IoT-Fog dependent SDN enabled device structure. In this article, a model for the IoT device architecture is provided that uses a Fog node of edge computing stage. But there is a lack of specific analysis into the system's architecture in this article. Nevertheless the study [66] resolved the lack. This study provides a detailed analysis into the system's function and the characteristics of its main components. The benefits of the new design as well as its future facilities are also discussed. These potential service steps are basically based on matrices known as QoS. But the overall QoS specifics of the system are not discussed in that study. The authors in [64] implement a QoS Aware Service Orchestration architecture called SDFog. This research explores the difficult problems influencing the cloud and suggests a service oriented middleware that along with SDN incorporation uses cloud and FC. A study that utilizes SDN to monitor wireless fog networks is presented in [67]. The authors provide an SDN permitted wireless fog architecture that incorporates both OpenFlow and distributed wireless protocols. By enabling customizable fog routers, the module offers less latency and efficient load balancing to release the network load. But the subject of this study is not wireless network QoS. The work [68] conducted a study concentrating on wireless network QoS. This study suggests a system that offers advanced FC WSNs for software-defined QoS. Some analysis was also done by addressing unique application that combines SDN and Fog. The work [125] emphasizes on

dynamic computing optimization of Industrial IoT by combining SDN and Fog, and [126-127] emphasizes on the adhoc network of vehicles by combining SDN and Fog.

Now the novelty of our thesis's second approach compared to the above related previous works can be seen. As per our study and knowledge concern, there is no research of the type that couples roof computing with SDN. That means there is no framework/architecture that integrates the paradigms of cloud, fog and roof computing with SDN. The same goes for DC as well. This implies that there is no framework or related research that incorporates dew, roof, fog and cloud architecture where this combined architecture is regulated by SDN. In addition, there is no kind of study that conducts simulation on the architecture of combined SDN and roof computing. This implies that there is no simulation performance on the frameworks/architectures that incorporates the cloud, fog and roof computing paradigms and supported by SDN. The same statement goes for DC as well. This again implies that there is no simulation performance on the frameworks/architectures that incorporates the cloud, fog, roof and dew computing paradigms and is managed by SDN.

2.4 SUMMARY

As this thesis presents two approaches to improve IoT ecosystem's QoS, therefore literature review is performed considering those two approaches. Initially the literature review concerning first Method Is presented and summary is provided in tabular form to understand the novelty of the proposed "Method I". Then the study of literature concerning second Method Is presented. These studies are analyzed and make comparisons with our presented "Method II" to show the novelty of this thesis.

CHAPTER 3

METHODOLOGY

3.1 INTRODUCTION

The major contributions and actual novelty of this thesis lies in this chapter. Here two novel methods have proposed for IoT ecosystem to improve its QoS. These methods have proposed based on the concept of multi-tiered computing paradigm architecture which is incorporated with SDN. Initially, the first method (Method I) is presented which combines Cloud and Fog computing and they are incorporated with SDN. Then a second method (Method II) is proposed which presents a novel SD-DRFC framework combining Cloud, Fog, Roof and Dew computing paradigm which is also incorporated with SDN. Therefore this chapter mainly divides into two sections. One section demonstrates the Method I and another section demonstrate the Method II.

3.2 METHOD I: COMBINED FOG AND SDN

The "Method I" combines Cloud and Fog computing and they are incorporated with SDN. The objective of this "Method I" is to improve the QoS of IoT ecosystem. For this purpose, a novel architecture and algorithm is presented in "Method I".

3.2.1 The Proposed Architecture of Method I

IoT devices produce or perceive data from environment. Fog computing collect those data and process them locally that are time-sensitive and require real-time processing. Data that are less time-sensitive or time-insensitive are processed respectively by aggregate fog node and cloud. SDN here used as a things-fog-cloud network. In essence SDN is the controller for its underlying

overall process. The control plane of the SDN controllers controls the data plane created by IoT devices. Figure 3.1 defines an architecture model that takes advantage of SDN and Fog computing paradigm to boost IoT system's QoS. The architecture has four layers based on functionality. The architecture explains how each layer is connected to another layer, and which part depends on what for better processing the data and to provide better QoS. The Layers and their functionalities are:

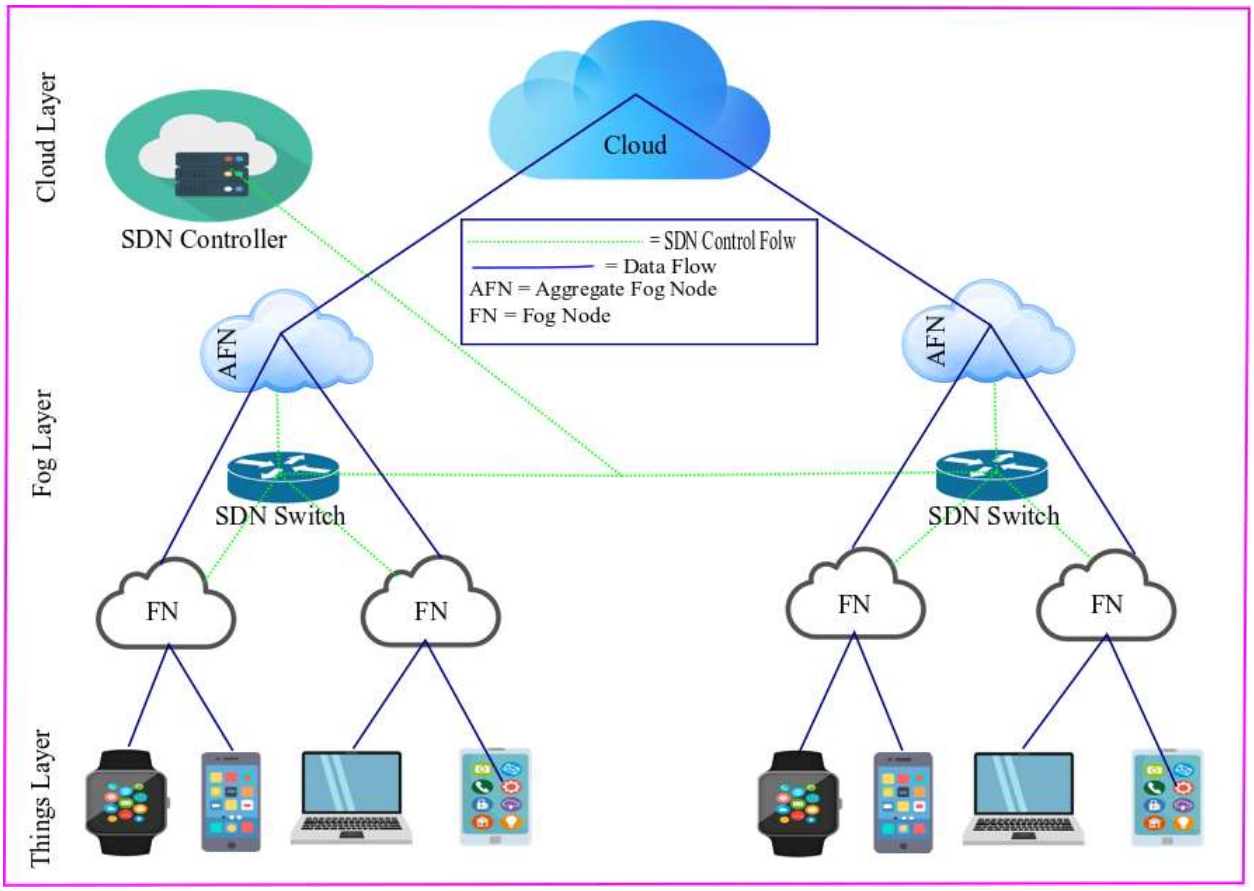


Figure 3.1: Proposed Architecture of Method I

Things Layer: The Edge architecture layer that usually forms with IoT devices can be called "Things Layer". In this lowest layer, the end devices (e.g. gateways, sensors, and controllers etc.) are laid [128]. These are lightweight, memory-restricted, often battery-conducted electronic gadgets with on-board sensors and actuators [129]. Basically these devices are called internet-connected devices that mostly have low demands on bandwidth. The main capabilities of things layer devices are: Sensing and storing data, Performing mild computing, Able to join to networks, Able to send recorded data to next layer, and Able to sends the separated data that

demands services used up locally by the fog layer. Users can communicate with fog paradigms using IoT sensors through IoT applications. Installing the intelligent and creative software on end devices improves its functionality.

Fog Layer: By fact this is a multi-layer structure. The layer's edge is referred as the Fog Nodes (FNs). A FN could be any instrument with computing, analysis, storage and connectivity to network. For example: Industrial Controllers, Routers, Switches, Embedded Sensors, and Surveillance Cameras etc. A FN consists of three key components: Controller, Software, and Communication (Depicted in Fig. 3.2). The Computational Element offers tools for implementing Software Modules. Modules are imposed to Micro Computing Instances (MCIs) inside the Computational Component, where resources, such as CPU, memory, bandwidth, etc., are whacked according to the necessity. Communication Component performs functionalities in networking such as routing, packet forwarding, etc. The connection to the network could be called access points (APs). Each AP covers a specific region and manages all data within that region. A fog node's controller portion controls and handles computational and communication components operations. In addition, many data structures are maintained by the controller component. The Module Sleeping Block (MSB) includes non-executing Application Modules (AMs) among them. A FN monitors the AMs introduced in it employing Placement List (PL) and stocks route-related information in its Routing List (RL) of other application modules. After executing a module, the host of subsequent module is either connected with PL or RL of the preceding FN. In fact, FN conserves metadata of the settled modules in a Data Container. Additionally, all data stored in Data Container temporarily. Here data container performs processing and storing. It analyzed the data on the basis of SDN controller-defined rules. SDN controllers do not monitor server of the fog directly. Alternatively, it uses the AP controller that is part of the Aggregate Fog Node (AFN). AFN is simply an FN but with a larger size and several FN operations are combined. A cluster of AP is attached to it within the AP controller. It thus covers and monitors a wider region and manages all data from the APs and servers that are located under its covered area. It is the highest layer of Fog and it governs all the networks that underlie it. The FNs nearest to edge of the Network ingest IoT system data. The fog IoT application module then guides the different data types to the optimal place for analysis. On the FN nearest to things producing data, the utmost time-sensory data is analyzed. They can react to Sub-seconds within Milliseconds. Data is forwarded to AFN for review and execution, which

could take seconds or minutes to wait for action. Within Second to Minutes, they can respond. Time-insensitive data for historical analysis, analyzes of big data, and long-term storage is sent to the cloud. Their response varies within Minutes, Days, and Weeks.

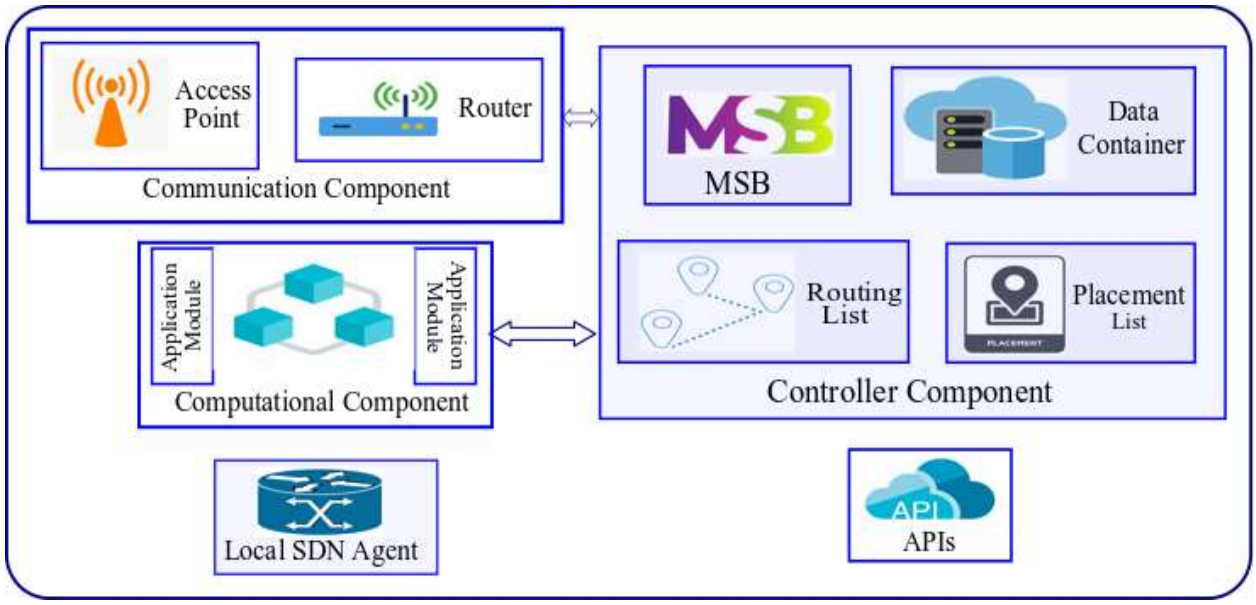


Figure 3.2: Components of FN

In addition, FNs contain an array of APIs and a local SDN agent with three main components. APIs reveal for delivery and development of software, managing of resource and power. Such APIs provide full entry on a physical machine to hypervisors, diverse OS and service containers. These also allow distant tracking and control of physical arrangements as interfaces with the CPU, memory and network. Local SDN agent allows the fog node to do the SDN controllers prescribed operation.

SDN Layer: It's the most significant layer of architecture. SDN consisting primarily of three layers (Layer of Application, Control and Infrastructure). Infrastructure layer consists of different networking component which builds the underlying network for transmitting network traffic. It might be a combination of the network data center switches and routers. Control layer is the ground of the control plane where the logical purpose of the SDN controllers is to control the network infrastructure. Two key operational aspects implemented by SDN are: SDN controller and SDN switch. There should be a management exchange protocol between them. The traffic management functions were optioned out from SDN switch and puts into SDN

controller. The SDN controller has been the centralized control unit that administers a series of SDN switches on an SDN protocol. OpenFlow is such a standardized protocol by ONF allowing connectivity among SDN controller with SDN switch. The SDN switches which run through the OpenFlow protocol can be called the OpenFlow-enabled SDN switch. Every SDN controller covers a massive area under it and can be named as a partition.

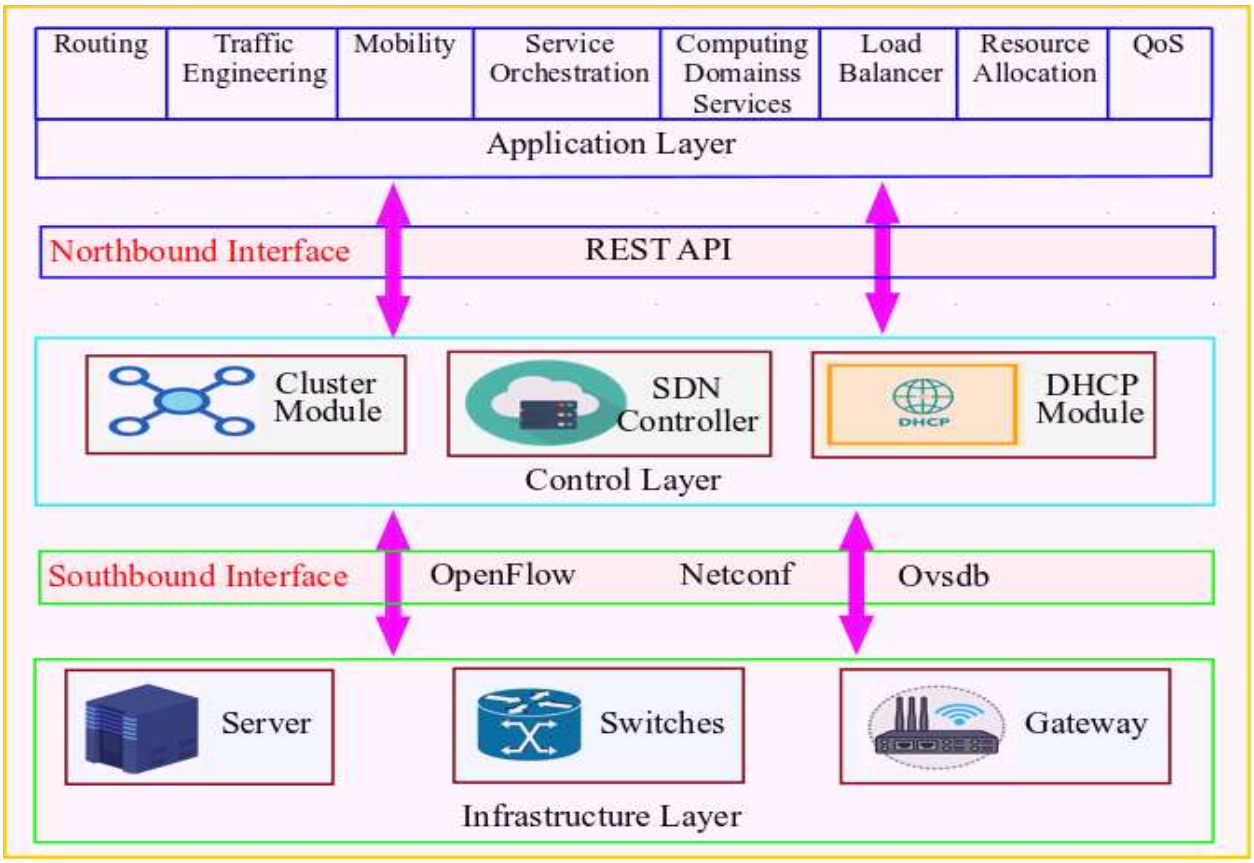


Figure 3.3: Components of SDN

Application layer is the uncovered zone for creating as many creative applications as feasible through targeting whole network topology knowledge, network status, network statistics etc. Many forms of applications can be created, such as network automation, network configuration and management, network governance, troubleshooting the network, security and network regulations [130]. Northbound interface is intended to interact with the superior layer and normally would be understood by REST APIs of SDN controller. Southbound interface is intended to communicate with lower network elements infrastructure layer most usually would be implemented via southbound protocols such as: Openflow, Netconf, Ovsdb, etc. The SDN

switch of one partition can be connected to the switch of another partition to allow for data exchange between controllers. This involves the scheduling of traffic flows located within various partitions among IoT devices. The SDN controllers use common application interfaces to integrate and mediate between various controller domains [131]. Within our architecture SDN's functions are: SDN manager controls switch/router flow control, Fog orchestration, Running analytics algorithms and orchestration of new rules across the network, Network state management, Injection of subduing logic into network elements allowed by SDNs, Enables smart networking by allowing IT to control the flow between physical devices on one side and applications on the other, Listing what devices are inside the network, and Optimal election of IoT access points.

Cloud Layer: This is the Architecture's upmost layer. Cloud computing is the availability of various on-demand services of computer system resources via the internet. A cloud computing architecture can be classified in four layers: physical, infrastructure, platform and application [132]. The physical layer is responsible for dealing with the cloud's physical assets including routers, servers, switches, cooling systems and energy. The infrastructure layer is created by partitioning constitutional resources using virtualization technologies such as KVM and VMware to create a tank of storage capacity and computation resources. The platform layer settles on top of the layer of infrastructure, and this layer consists of operating systems and functionality frameworks. The application layer contains the actual cloud rules, for example, business software, and multimedia and web services. Cloud computing performs broadly: Universal Computing and Global Storage. The gathered data from IoT devices are transmitted to cloud which needs to be processed during a longer time period and which also needs to perform complex analysis.

3.2.2 The Presented Algorithm and its Design Model of Method I

The latency in linking SDN switches to SDN controllers will be minimized if the network itself is partitioned across multiple small domains and then places the least number of controllers in the correct node of each domain. Regrettably, it also results in extra latency among controller ranges in multiple SDN domains. In addition there is need to choose the optimal access point for the data generated by the things devices and analyze that data in the optimal location to provide the best QoS. It is therefore necessary to propose an algorithm.

3.2.2.1 Design Model

The FNs and AFNs are intended to keep SDN controller constantly aware of capacities and location of the things devices to be served. Controller can thus create a full connectivity graph and optimization algorithms running on a regular basis to make better conduct of network resources and to boost QoS. The IoT devices should be IP handled for this algorithm. The steps involved are:

Step 1: The SDN is split into several small network domains. There is only one central controller that can be considered the master controller, and small controller domains can be considered slave controllers. SDN switches are being utilized as boundary nodes to partition the main distributed routing domain into multiple sub domains [133]. Hence SDN controller is the master controller and SDN switches are the slave controller.

Step 2: One can label any small network domain as AFN under the slave controller. Every things device in the controller of the specific partition communicates through diverse variations of AP, coupled to native SDN switches (i.e. slave controller) to impose on different types of data flow from data servers.

Step 3: Optimal controller placement as well as its coverage area is based on: network topology, internodes size, node-to-node bandwidth, typical request rate, maximum persistence of the controller and minimal space between the two controllers.

Step 4: For effective OCP, utilize either optimum progressive SDN controller placement [54] or distributed policy builds on non-zero-sum game [55]

Step 5: Next a single AFN domain is clustered into several FN domain. These clustering are performed by the regulations of SDN controller. The regulations are executed by the SDN switches.

Step 6: SDN controller regulations that will be determined by application service providers (ASPs) monitor and control both the master and slave controller.

Step 7: When new things devices reached to the fog layer, optimal AP will be chosen based on: current multi-network capabilities in controlled partition, wireless communication processing

implemented and promising types of service devices. If it is not possible to locate the optimum AP within a fixed time frame, handover is recommended.

Step 8: For WLANs, a user-based AP choosing method [134] that incorporates factual data on AP can take place. But given that the range of Wi-Fi spectrum is minimal. The significant increase in need for wireless traffic is currently caused intra-WLANs over-congestion. For this reason a creative AP allocation algorithm [135] for dense Wi-Fi nets (e.g. airports, bus stations etc.) that is based on a centralized system based on SDWN-based framework can adopted

Step 9: As things devices are mobile, the SDN controller forecasts the eventual location of the things device in the near future. UbiFlow [136] is best for mobile devices, and is the main SDN-dependent IoT platform for flexible flow management and multi-network mobility management. Based on UbiFlow strategy SDN controller can forecasts the eventual location of the things device in the near future.

Step 10: When things devices reach a new area, the handover is carried out since each FN adopts the Mobile Fog Programming Model [137] in the same ecosystem. Appropriate handover can be achieved by cooperation between controllers in the regulation of SDN-based mobility. UbiFlow provides a layered connectivity based system approach for both device handover and mobility regulation

Step 11: The FN that is optimal to process the thing devices data is selected based on optimal access point selection strategies (which are mentioned above) and this optimal FN will ingest the things devices data.

Step 12: If ingested data from things devices under a FN region overloaded it's (i.e. FNs) processing capability then central controller will split the FN of that region to several smaller sub-regions. Then when these smaller sub-regions has less data load, these regions will merge together to create a single coverage area as before.

Step 13: The spliced FN or unified FN will perform the regulations of SDN controller via SDN switches.

Step 14: Different types of semi-structured and unstructured data will be collected by things devices. FN brings these data together, filter it out and pick the core values. Based on these core

values, FN will perform the weighting on these data. The optimal layer to process that data will be selected on that weight. That means the service offered to the data (real or non-real) will be depending on the weight of data. SDN controller will know the weights of data through SDN switches. It then choose the appropriate layer (i.e. FN/AFN/cloud) based on the data weight.

Step 15: Through the initial data processing and putting weight on data in FN; the SDN controller decides which data is time-sensitive and require service in real time and which data is time-insensitive and did not require service in real time.

Step 16: Finally ingested data will be processed in the optimal place. For time-sensitive data; it will be processed in the FN layer. For less time-sensitive data; it will be processed in the AFN layer. For time-insensitive data; it will be processed in the cloud layer.

3.2.2.2 Notations

Table 3.1 represents the Notations utilized in the presented algorithm.

Table 3.1: Notations

Symbol	Definition
S	Software Defined Network
S_1, S_2,…,S_N	Small network domain
SDNC	SDN controller
SDNS	SDN switch
F_A	AFN (aggregate fog node)
OCP	Optimal Controller Placement
C_1	Network topology
C_2	Internodes size
C_3	Node-to-node bandwidth
C_4	Typical request rate
C_5	Maximum persistence of the controller
C_6	Minimal space between the two controllers
B_1	Optimum Progressive SDN Controller Placement [54]
B_2	Distributed Policy Builds on Non-Zero-Sum Game [55]

$F_1, F_2,.., F_N$	FNs clustered from AFN
ASP	Application Service Providers
OAP	Optimal Access Point
p	Current multi-network capabilities in controlled partition
q	Wireless communication processing implemented
r	Promising types of service devices
h	Handover actions among heterogeneous network of access
E_1	User-based AP choosing method [134]
E_2	Creative AP allocation algorithm [135]
z	Eventual location of the things device in the near future
D_S	Semi-structured data
D_U	Unstructured data
D	Raw data generated from things devices (both D_S & D_U)
$f_1, f_2, …., f_n$	Smaller sub-regions of FN
F_U	Unified FN
D_W	Weighted data
D_{TS}	Time-sensitivity of the data
D_{OPP}	Optimal processing place for data
T_S	Time sensitive data
S_R	Real-time service
T_L	Less time sensitive data
S_{NR}	Near real-time service
T_I	Time insensitive data
S_{NONR}	Non real-time service

3.2.2.3 Presented Algorithm of Virtually Partitioned SDN

Algorithm (I) shows the algorithm's pseudo code for virtually partition the SDN. The main objective of this algorithm is to provide improved QoS by partitioning the corresponding FN and AFN through the SDN controller. It blends the benefits of SDN and fog simultaneously.

Algorithm (I): Pseudo-code for the Algorithm of Virtually Partitioned SDN

Partitioned SDN into

$S \rightarrow S_1, S_2,\ldots,S_N$;

Label each $S_1, S_2,\ldots,S_N$ as AFN;

Select OCP based On

$OCP \rightarrow C_1 \cap C_2 \cap C_3 \cap C_4 \cap C_5 \cap C_6$;

For effective OCP

Apply $\rightarrow B_1$ or B_2 ;

Clustered AFN into

$F_A \rightarrow F_1, F_2,.., F_N$;

Monitor & Control SDNC & SDNS based on

Regulations of ASP;

IF things devices enter fog layer then

Select OAP based on $OAP \rightarrow p \cap q \cap r$;

ELSE IF OAP can't found within a fixed time frame

then controller suggest $\rightarrow h$;

ENDIF

For WLANs OAP

Apply E_1 ;

For Dense Wi-Fi's OAP

Apply E_2 ;

When things is mobile then

Controller predicts $\rightarrow z$;

IF things devices reach a new area then

Controller performs $\rightarrow h$;

ENDIF

FN optimal to things ingest D;

IF D overloaded FN region then

Splice $F = f_1, f_2,\ldots, f_n$;

ELSE IF D is under loaded in FN region then

Unify $f_1, f_2,..., f_N = F_U$;

ENDIF

Spliced or unified FN performs SDNC regulations via SDNS;

FN converts

D_S& $D_U \rightarrow D_W$;

Based on D_W

SDNC determines D_{TS} & choose D_{OPP} ;

IF D = T_S then

Execute in FN & provide S_R ;

ELSE IF D = T_L then

Execute in AFN & provide S_{NR} ;

ELSE IF D = T_I then

Execute in cloud & provide S_{NONR} ;

ENDIF

3.3 METHOD II: SD-DRFC FRAMEWORK

Anyone in today's world typically uses multiple smart devices. Smart phone has become a must-keep item for users amongst these. In addition, other smart devices (e.g. smart watch, fitness tracker, etc.) are carried by the user when moves. The smart phone can thus be used as a dew device in such cases, and can provide the dew computing service. Because compared to smart watch or other similar low powered smart devices, smart phone has much higher storage and processing power capacity. The concept does not only apply to low powered smart devices where smart phones can be used as dew devices. The sensors that can fall within the smart phone range and have the ability to communicate with it (i.e. the smart phone) can also used as a dew device for them (the sensors). Basically we may claim that here smart phone is the dew device that performs the dew computing and other devices that use this smart phone as the dew device are the things devices. So in this context, smart watch, fitness tracker, smart car, sensors etc. are all things devices. In addition the laptop, PC or other similar devices may be used as a dew device when the user is still and stays in one fixed location. The things devices will be the same as description given above. Because modern laptops/PCs offer considerably much more

processing, storage, and battery power than things devices. This is another perspective which mainly focuses on users in fixed location.

Thus, in our proposed framework, a Virtual Dew Cluster (VDC) will be created by the smart phone or related computing power devices (dew devices) along with its connected things devices. The detailed explanation of VDC is given in the section 3.3.1. Since the dew devices stay very close to the things; therefore the latency will be negligible. So this interplay of dew-things can satisfy delay sensitive services that needs very fast response. For their (dew-things) activity, there will be something like an instant response to users. This indicates that if the user action (transmitted data/command) activates some predefined DC system rules; then the DC system will execute the predefined rule as an immediate response. It will guarantee the service in extreme real-time. We know that in an architectural hierarchy the lower the level, the more the number of nodes is. But in terms of processing capability and storage the resources of these lower level devices are far less. The DC system thus can't deliver the services that demand a great deal of processing power or storage. So a DC system could get the necessary volume of data processing from roof, fog or cloud counterpart with internet connection. Users can also then access and change the data (which is now available on the internet) at any level of the roof, fog, or cloud based on current internet connectivity. The proposed four-tier model enhances customer preference at any moment in time for any tier of service. Dew server has full flexibility in the retrieval and synchronization of its own data from any place of the framework. Thus the DC makes its consumers super versatile experience by optimizing the options of data processing with roof, fog or cloud services. The roof server will be scanned at when asked data is not available in dew. Upon query failure in roof, fog server will be checked and clouds will perform the function upon fog failure. This is for data retrieval by DC from upper layer services. But if we focus on offloading data processing on these upper layer services then the flow of operation will be different. The roof server will be used when requested data processing is beyond the capability of DC system. Upon processing failure, fog server will be used and clouds will represent the purpose after fog failure.

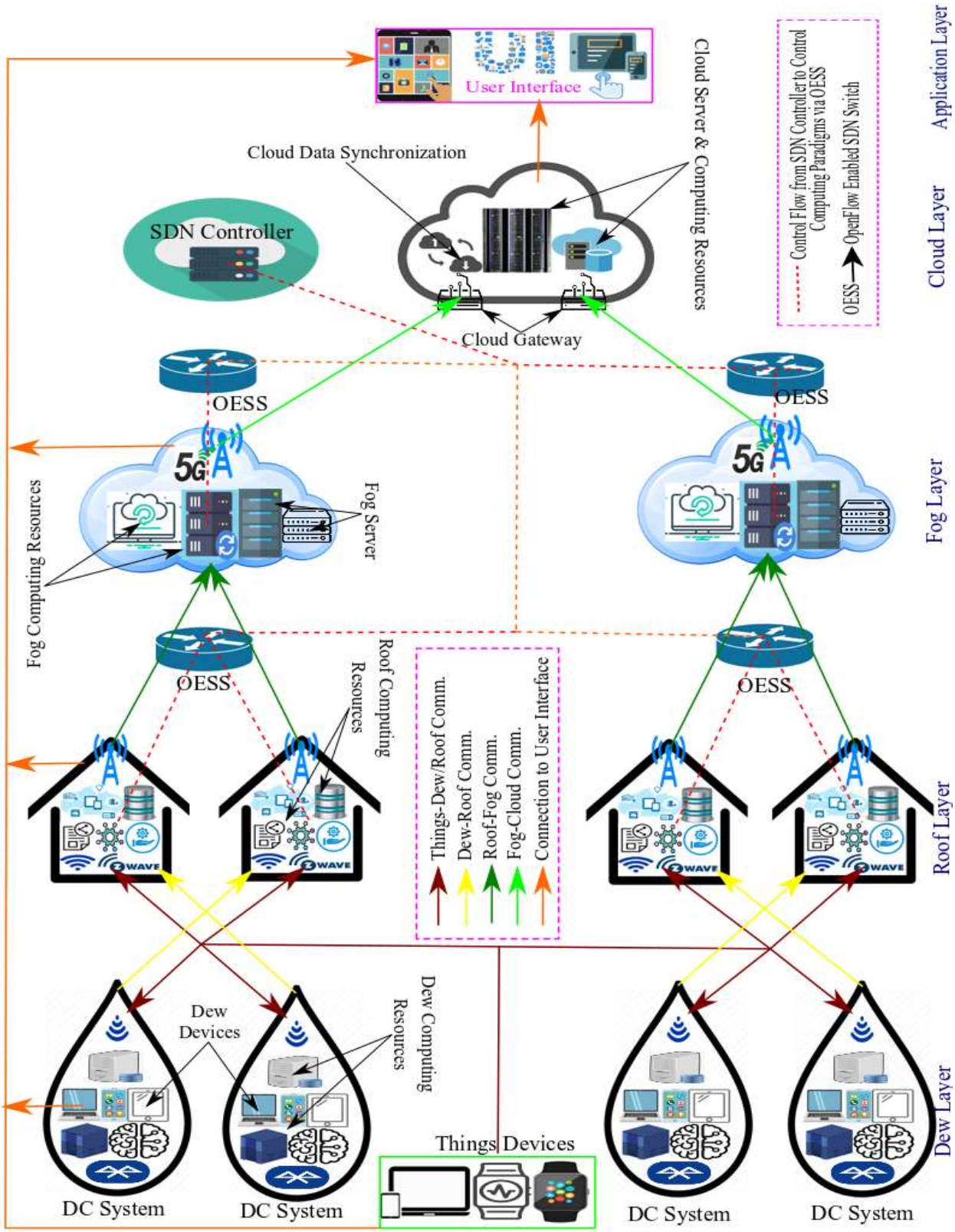


Figure 3.4: Proposed SD-DRFC Framework Architecture of Method II

Now consider a situation where the user is mobile and has no smart phone or related devices that can be used as a dew unit. In addition take into account that the user is at home and the laptop, smart phone, PC and so on are all out of operation and therefore can't be used as a dew unit. But users must use devices (things) that are not capable of operating as a dew unit. In all these cases the things will be connected directly to the RC network. Therefore the nearest RC device must serve as the immediate computing paradigm as the DC is not available in this situation. The RC module will therefore serve as next-hop communication to both dew/things devices in this context. In the situation when dew devices are connected to the RC network, the network would perform the processing or storage operation (i.e. the services) that DC system isn't able to perform. That means RC will support the DC system by performing the remaining processing of dew. In the situation where things are connected directly to the RC network then the RC network will support the things with the initial services. This means that the RC network will provide the preprocessing and other related services for the initial perceived data by things devices. Throughout an IoT ecosystem, the roof is a fixed component of the network which can always be reached. The Roof connects the physical and virtual world by linking the dew devices and things to the Fog/Cloud to create cyber physical systems (CPS). Thereby roof creates highly scalable CPS for the IoT ecosystem. The RC module constructs the context from the data it obtains from the dew/things devices and based on the context it offers security to the corresponding devices. This is called "context-awareness security". Numerous vendors manufacture the dew/things products. The industry for the IoT platform includes multiple vendors (over 600), such as Cisco, IBM, and SAP etc [138]. Therefore, for these products, harmonization is important to deliver services. This is accomplished in the RC module, too.

The FC module exists in the middle of the network. This acquires data from RC devices, processes and stores data in local FC network. It dramatically reduces data flow around the internet and offers high-quality, endpoint-supported localized services with efficiency. Because when the things devices are connected to the DC network, the data initially migrate through dew processing. The devices that can't be connected to the dew network are connected to another processing system named RC. Indeed the data that needs extensive processing, which DC can't meet, shifts to RC. RC conducts preprocessing of data, elimination of redundancy, cleanup and filtering of data, extraction of useful information and all these are carried out locally. Only data that requires additional processing that can't be supplied by minimal computing power from

roofs is sent to FC. The data that requires to be temporarily placed somewhere in the middle of the network, rather than cloud storage; then fog server is used. FC therefore allows for low latency and matches the requirement for near-real-time interactions. Finally, the cloud is this framework's uppermost part which essentially handles long-term storage and analysis of big data. User data that required to be permanently placed then cloud storage is undertaken. The permanently stored data is analyzed using cloud-based methods (i.e. big data analytics). Then the planning is created on the findings of the analysis, or advice is given to enhance the user experience. Cloud typically performs services that aren't sensitive to delay (i.e. that don't demand real-time service).

How these four computing paradigms could provide functionality to an IoT ecosystem is when integrated is illustrated below. The things devices collects different sorts of semi-structured, unstructured as well as some structured data. It is crucial to bring these data together, filter it out and pick the core values. Perform weighting on these data is also important. Because the service offered to it (whether real or non-real) will be calculated depending on the weight of data. In this context the DC and RC model executes these kinds of functions. That means they organize the unstructured data and turn it to information. Then they filter the information to extract necessary core values. Through these two layers (DC or RC); the ecosystem can decide which data would require service in real time and which did not. Appropriate computation is done in the correct layer based on the insight. In this ecosystem, FC and CC provide additional processing and storage that would be required for some data. FC is in the middle of the framework and can have some advance level of processing compared to DC and RC. Temporary storage is also provided by FC. But we also need cloud support for Big Data analysis and permanent storage. Therefore, CC's Big Data Engines obviously involve machine learning, artificial intelligence mechanisms in numerous ways to enable data useful for future planning, and thus we can manage and control everything.

Here we can see the novelty of this framework architecture. In the past, all these data gathering, putting them together, filtering them, extracting the core values, needing preprocessing and processing, and analysis of big data; all have to conduct either CC or later by integrating CC and FC. Thus it was very difficult for those traditional architectures to obtain time sensitive services. But the collection of data, putting them together, filtering them, extracting the core values and

needing preprocessing and simple processing is done through DC and RC in this proposed framework architecture. FC and CC conduct broader processing that can't be performed by DC or RC due to limited resources, big data analytics and temporal and permanent storage. Thereby RC and DC eliminate the pressure from both CC and FC. Since each layer performs its specific task and there is not much pressure on any layer; hence the framework is suitable for large IoT ecosystem.

There is also another layer called application layer at the top of the proposed framework. It provides user interfaces between the end users and the ecosystem to directly reflect the information gathered from the four computing layer. The applications/services are provided to respective users through these user interfaces. The proposed four-tier framework enables users to pick any layer of service at any moment through these user interfaces. Thus there is a direct link from each computing layer to the application layer. Through this direct link, the application layer's user interface offers users direct access (subject to access rights and privileges) to the necessary resources from the cloud/fog/roof layer.

From above study it can be seen that the resulting structure and its internal relations will be very complex as the design is four-tiered architecture. It would be a crucial task to orchestrate the service amongst these tiers. Within such a structure there will be some other big complexities which we have already mentioned in the “Introduction” chapter. So to handle these complex task for this framework; SDN is used. SDN here is performed as the network backbone for the framework. In this framework, SDN controller controls the overall interwork functionality via OpenFlow enabled SDN switches. In RC system and its underlying devices, SDN manage and orchestrate those devices and the network between them. In addition, the service management of this system is also managed by SDN. The same is true for FC system. SDN’s roles and functionalities in this framework in briefly described in section 3.3.4.

3.3.1 Role and Functionalities of Dew Computing

Most of the time, IoT devices collected data and implicit information are maintained and processed in the corresponding cloud, fog, or roof computing systems. It implies that all the stuff in such a scenario need to always align with internet connectivity. But, for certain reasons (e.g. minor aspects of local climate, socio-political problems, expense, accidents, user behavior, etc.)

it might not always be feasible to use the internet. Therefore, it'd be ideal if IoT can be integrated in any collaborative environment with DC. This new trend would provide all proper assistance to the things gadgets without internet access. Certainly, things will interact with the cloud/fog/roof once needed to upgrade the system and collect relevant data when internet is accessible.

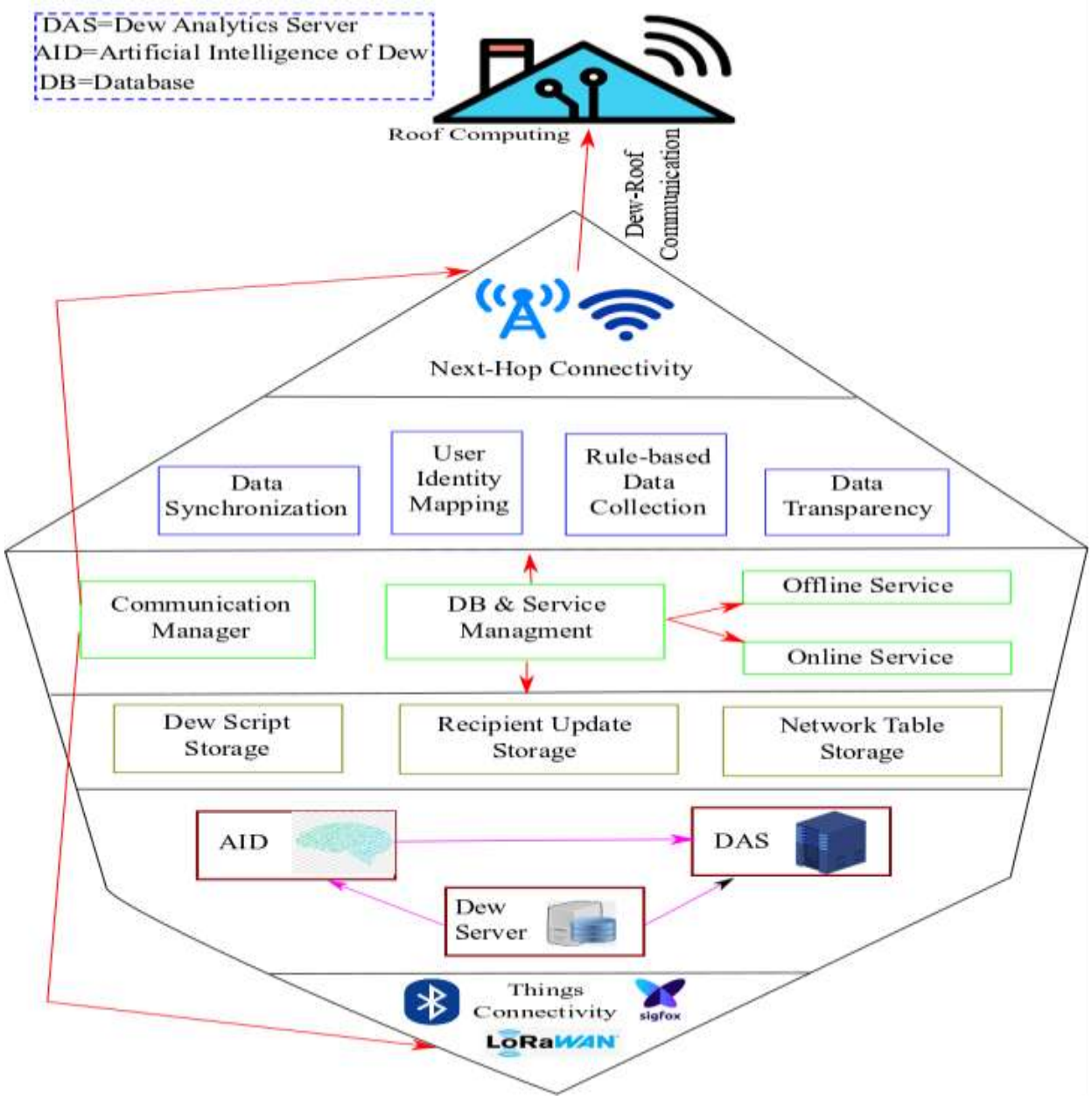


Figure 3.5: DC System Architecture for SD-DRFC Framework

By developing a VDC, one can implement dew services in an insanely dynamic cum distributed manner. VDC is able to provide strictly ad-hoc service delivery and information sharing between

inter-dew devices. The VDC could be a stand-alone computation and interaction unit (e.g. a smart phone, a laptop, a tablet, an intelligent residence processing center etc.). Basically the dew devices and DC system collaboratively constructs the VDC. It requires connectivity with the things devices as well as with high-level servers (roof servers) and allow user interface. Therefore the first point to discuss on VDC is the technologies of access/communication. Communication manager handles access both of the things devices to the dew and dew devices to high-level (RC) networks. It serves the purposes including [139]: Pairing, Connecting, Disconnecting, and Reconnecting. In the setup phase, pairing is performed, and once the things device is fairly close to the dew server, the connection begins. When the things are out of range of the dew server's coverage, it disconnects and attempts to reconnect and setup communication when the paired device gets the signal. The very same regulations apply for the higher-level (RC) communication systems. There are basically two instances of communication manager. One is the low-power radio communication with the things which we called "Things Connectivity." The other is the connection we called "Next-Hop Connectivity" to the higher-level (Roof) servers on the Internet. The things devices can't use large distance dispatch technologies. Bluetooth or other low-powered wireless personal area networks (WPANs) can perform the task pretty much efficiently. Bluetooth5 offers location awareness and fast speed to establish an ad-hoc network among dew devices. For connectivity of things, other WPAN technology such as Zigbee and 6LoWPAN can also be considered. However Wi-Fi can be used for dew-roof connectivity. Low Power Wide Area Network (LPWAN) protocols (e.g. DASH7, Weightless, LoRaWAN, LTE-A, Sigfox etc.) are beneficial in developing a protected peer-to-peer connection among different dew devices in a broad geographical region [140]. Mobile dew users can use Bluetooth Low Energy (BLE), Wi-Fi, 3G, 4G or low-power 5G communication techniques to experience smooth dew-service properties.

In addition to the access technologies, the VDC is formed by a number of other components. The most important component among these is the Dew Server (DS). In this framework, the DS is an intermediary server that can offer services to things devices. Also if more computation or storage is needed, DS can offload services/data to the upper RC/ FC/CC network. So its fundamental task is to absorb, store, process, and deliver data and results to high-level servers from nearby things devices. The DS size varies among device to device which basically depends on the volume and the accuracy of data. The DS operates just like cloud service but on local devices. To

perform its functionalities, DS should have some vital specifications such as: Architecture, objective, and utilization should be specific [122], Should be light-weight and low power in terms of processing and battery, Regularly utilized user data should be stored in its databases, The volume of stored data shall be minimal, and Its services shall be available to users with/without internet access at any time.

The two prominent innovations most helping the DS are: Dew Analytics Server (DAS) and Artificial Intelligence of Dew (AID). DAS is a local version of CC server that either analyzes data gathered from things, or the data that previously stored on DS. The perceived data stream from things can be very noisy. So data preprocessing utilizes digital filters or similar methods to preprocess data (i.e. removal of noise). Once preprocessing is performed (i.e. noise is removed), then the actual processing on data will carried out. Data analysis provided the conclusions for each collected data sample. Typically conclusions are drawn on the basis of a classification algorithm with an aim of assessing the class of received data stream. The AID collects computational outcomes from the DAS. The AID utilizes the data to configure and customize the dew system to the user to improve their experience after collecting data from the DAS regarding usage patterns.

Now short discussion is performed on some sub-component of VDC which have concise but important impact on dew and overall framework. Dew script is a scripting instrument obtainable to dew user for potential improvements to the collected data that deposited on dew device. Scripting can considerably lessen the pressure of network management. Since SDN is not directly monitoring or regulating the DC system in this proposed framework, this dew script is therefore quite efficient and essential to handle or monitor the network falling within the range of the DC system. Once the service or data is delivered to the RC network, SDN will then take on the necessary task of network management.

Advanced synchronization method for data is inevitable for working in an internetwork-free (i.e. both with/without internet) environment. Synchronization is the main factor in DC approach for preserving system sustainability. The Mobile Cloud Synchronization (MCS) concept [141] may be implemented to dew devices for synchronization purposes. The MCS mainly focuses on the syncing among mobile and cloud systems. But this method can also be updated to accomplish synchronization between dew devices and cloud/fog/roof system. MCS is essentially the basic

principle here. But to implement this principle in this framework, appropriate mechanism is required.

Data synchronization mechanism addresses the question "when does an application synchronize data among device and the remote system (e.g., cloud/fog server)?" The solution to this issue needs to take into account the restrictions of several aspects including network connectivity and presence, data freshness criteria and user interface model [142]. There's a few latest APIs for mobile app developers that can use "Data Offline and Sync" technique to make an user app that allows users to execute critical duties offline, and then synchronize all offline alterations with the server whenever the gadget goes back online. Those APIs which is inspired from [143] is shown in Table 3.2. Now based on MCS concept and applying the mentioned mechanisms with the similar APIs that is given in Table 3.2; applications or systems can be built. These would be able to perform synchronization between dew devices and any of upper level servers.

Table 3.2: APIs that Supports Data Offline and Sync Feature

API	Platforms	Features
Sync Express	• Cordova • JavaScript	• Synchronization of fundamentals. • User friendly. • Demands limited modification of existing code. • Operates with any frame of JavaScript. • Dispatch requests for modification to one resource entity at a time when system reconnects. • The server version of the entity is always overwritten.
Synchro-nization	• Android • iOS	• Robust synchronization. • Operates with customized APIs which comply with the synchronization. • Dispatch all modifications in one request once system reconnects. • Provides options as to what to do if the server version of an entity alters while modification is accomplished offline (server wins, client wins, and conflict conserve).

		• Provides options for how long the resources of objects can be stored on the device, when server data can be refreshed, and which resources could be modified offline. • Synchronizes to Storage platform automatically.

Data transparency aspects are: (1) to cover up few details; (2) to build a delusion. DC requires opt-in to two kinds of transparency methods in this context, namely data replication and data distribution. Dew users would have two choices in the data replication method to penetrate the data. One is local dew-data and another is remote cloud/fog/roof data. A dew user should not be conscious about what copy of data he presently accesses while accessing the data. Similarly, in the data distribution method, a dew user should not be capable of recognizing the distribution sequence of data access. The dew DB management system is basically a customized version of the individual DB to provide real-world data storage for things devices and individual dew site surfing experience. There are also many services provided by DC (Scripting, Synchronization, Identity Mapping, Transparency, Synchronization etc.). The service management can also classify the services that can be made available while the dew system is online. Once the system is online it can provide all the dew-supported services. But this is totally different for offline cases. For this reason service management should effectively identify and deliver offline services. This can be done based on the dew device-supported algorithms (most often the machine learning algorithm). VDC's DB and Service Management section manages VDC's overall DB and Service related functionality.

3.3.2 Role and Functionalities of Roof Computing

Figure 3.6 to clearly demonstrate the role and functionalities of the roof. Southbound interface is for the device and network management and communication for the dew systems/things device, so that a device (dew or things) can be placed under the roof in a secure way. The north interface will assist the fog and upper-level service providers to construct an assured scheme for protecting data and privacy. The horizontal interfaces (eastern and western interfaces) will assist edge users to handle things, as well as privacy and security. The security issues engaged are validation, verification, and secure key development for revealing IoT services under roof, while the privacy

issues helps end users to handle appropriate access to data and IoT services. These functionalities of security and privacy are performed using contextual analysis.

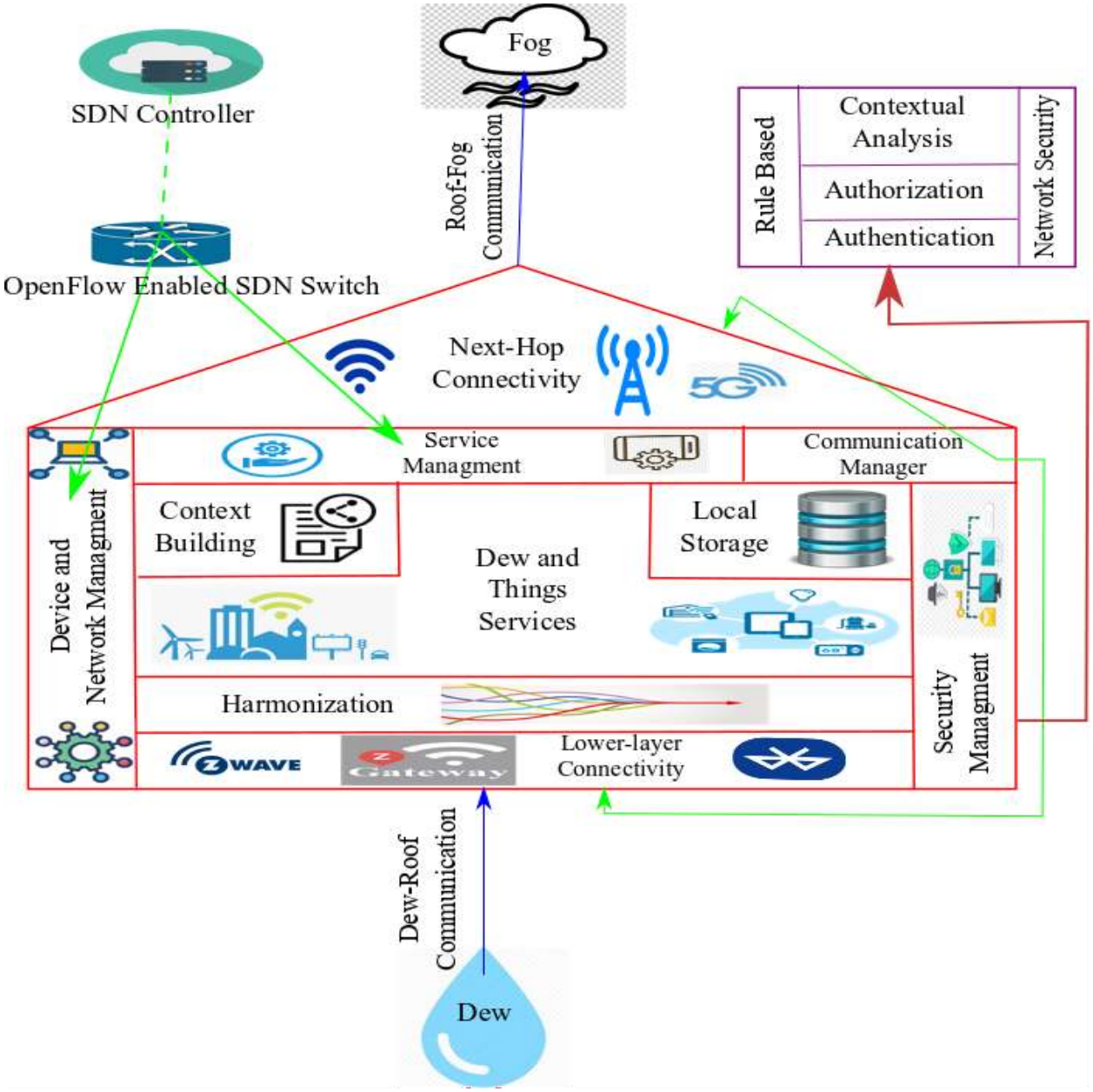


Figure 3.6: RC System Architecture for SD-DRFC Framework

Roof accommodates the broad range of dew/things devices with varying access technologies. Roof is liable for building the local network and offers independent internet connections for remote access. Like dew, RC also uses a communication manager which provides services to two

instances. First one is the low-powered wired or wireless communication with the dew/ things devices. In that framework, this is the "Dew-Roof Connectivity". Technologies used for such communication can be IEEE 802.15.4, IEEE 802.11, Wi-Fi, 3G/4G/5G or other similar radio connection, Bluetooth, various Low Power Wide-Area Networks (LPWANs) and many others. The second connectivity managed by communication manager is the connection to the FC system. This is the “Roof-Fog Connectivity” in this framework. Innovations utilized for such connectivity are low/high-power medium-range communications. Ultra-Wideband (UWB), cellular (2G, 3G, 4G, LTE-MTC, 5G), Sigfox, LoRa for long-range networking are few of the most commonly used standards of communication.

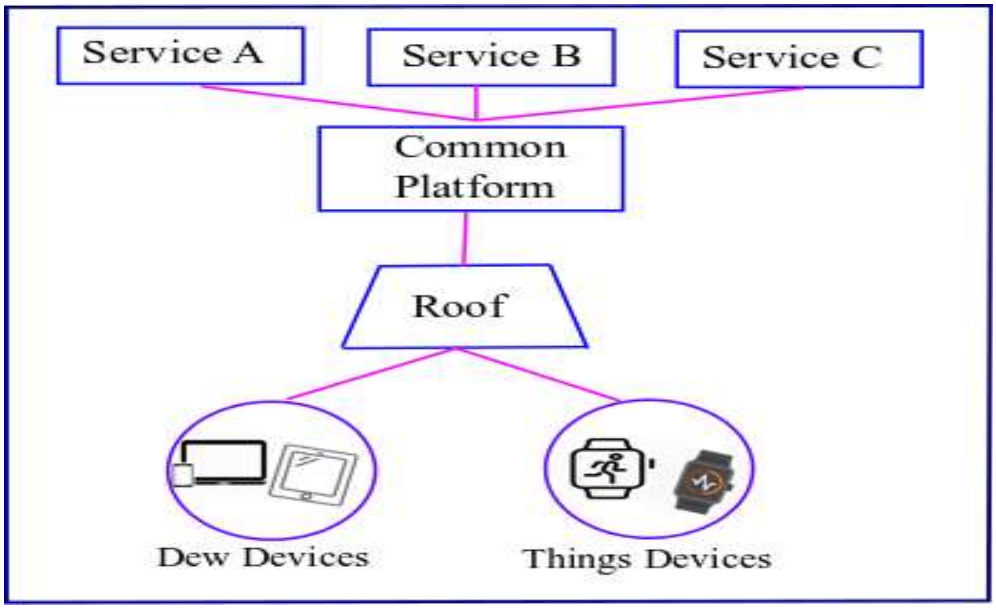


Figure 3.7: Roof Harmonization

One of the most important functionality performed by roof is harmonization. Almost all of the IoT solutions nowadays are vendor guided and partitioned vertically. This implies that the vendor delivers devices, IoT hub/access point, cloud or other computing solutions. Unless the vendor is a major player on the marketplace, it is impossible that one would get a new gadget to connect to their access point in which both of them (the gadget and the access point) are same. In addition, dew and things tools are used in a manner in this proposed framework where both of these tools can be attached to the roof at the same time. There is very little likelihood that these dew and things devices and all of the upper-level services (i.e. Fog/Cloud services) will come from the same vendor. There's a high chance that these gadgets will originated from various

vendors. Therefore harmonization is must needed functionality for this framework and roof provides it. Roof with standard interfaces would harmonize the IoT ecosystem, allowing further standardized IoT access points and upper-level (cloud/fog) services to be designed in such a way that one might pick gadgets from any vendor and then get the services from any other vendor.

IoT applications need two forms of data analytics: Context Building and Big data analytics. Context Building is an analysis of existing situation in real-time. DC provides real-time services within this proposed framework but only in a limited manner. The context building for this framework is also limited in DC system as there is no guarantee that all devices will be connected to DC systems. Roof is the first computing paradigm in which all the underlying dew and things devices are connected and hence the harmonization is performed here. So the overall context building and related actions are takes place in roof in this framework.

The promising factor that made roof an attractive computing paradigm would be its security. Basically, this security is delivered based on the context that constructs in roof. RC is context aware. Context awareness relates to the concept that, depending on their context, devices can both perceive and respond. Devices may have details on the situations under which they can operate, and respond accordingly depending by rules. The procedure of context-awareness generally consists of three phases. These are: Context acquisition, Context representation, and Context reasoning. Let's consider an example, in order to clearly understand the above steps. A context-aware mobile phone can know it is currently in the conference room, and the user already sat down (context acquisition). Such detail is then provided via an effective data model (context representation) in this context-aware framework. The mobile phone could reach the conclusion that the user is presently in a conference and dismiss any insignificant calls (context reasoning). The evolution of mobile users and the growing complexity of attacks are replacing the conventional strategy with context-aware security. A context-aware strategy to the network involves collecting data from end devices, equipment, and network services prior accessibility to network resources could be given. The end device is given access, quarantined or blocked from network based on the form, position and identity of the device and the OS or applications that run on it.

A context-aware model can implements a security analytics engine (SAE) that returns multiple-factor-based risk score [144]. The SAE can be configured to measure who, what, when, where,

and why, as per the requirements of the system, user numbers, risks, procedures, apps, and facilities. The SAE reports scores to control points (like applications for access control, firewalls, and encryption methods) which can permit/refuse access or enable two-factor authentication prior access is granted. When a user tries to access an app (via end device) that utilizes the SAE, a flexible risk policy assesses the risk of giving the access permission. Every risk strategy is composed of many conditions that have assigned scores. These conditions and related scores are assessed separately for every access effort, and a total risk score is then computed by utilizing all the condition scores.

Let's consider an example, in order to better understand the concept. Consider a department's lab-room. Now this lab room will be used by regular students during the laboratory class, research students during non-class time for their research purposes and teachers for either their research purposes or any of the academic purposes. The regular students have to use the lab's device. They don't have permission to use their personal device. But the research students/teachers have the permission to bring their own devices but the device must have to be in the system's BOYD managed list. The lab room's system will blacklisted some websites which might not be necessary for teaching/research purpose. In contrary, there will be some white listed websites by the system which is essential for teaching/research purpose and these sites are frequently accessed. The web sites that aren't listed in either black or white list will have such risk score which will lead the user to a two factor authentication process. Finally the system will only have the permission to use its resources while the user presents in the lab room (on-premises). No remote user access is granted. Considering the parameters along with their conditions, Table 1 is given below with an individual risk score for each condition.

Table 3.3: Risk Scoring Parameter Conditions with Associated Value

Parameter	**Condition**	**Risk Score**
Time	Class Time	0
	Non Class Time but Office Hour (For Research Students/Teachers)	2
	None of Above Conditions	5
Device	Lab's Device	0
	Research Students/Teachers BYOD Managed Devices	2
	Other Devices	5

Website List	White Listed	0
	Black Listed	2
	Listed None of Above Categories	5
Location	On-premises	0
	Remote	5

Based on the conditions and related risk score; the system will evaluate the overall risk score. Depending on this overall risk score the system will permit/refuse access or impose or two-factor authentication prior access is granted. There are three situations. These are:

1. If Overall Risk Score < 6; then access is granted
2. If Overall Risk Score is 6-17; then two-factor authentication is required for access
3. If Overall Risk Score > 17; then access is denied

For example, consider a situation where a research student tends to use the lab room and its internet connection during non class time (Time parameter). The student will use lab's device (Device parameter). Now while using the internet, the student tries to access a website which isn't in the system's white list/blacklist (Website list). Also as the student is in lab room, hence the location will be on-premises (Location parameter). Now based on the above parameter and their condition and using the corresponding risk score from Table 3.3:

The overall risk score = (2+0+5+0) = 7

Hence the student has to go through a two factor authentication process for network access.

The roof gateways not only route but also store data (local storage) and execute data computation. Roof filters unnecessary data. RC will perform the overall data filtering for its underlying IoT ecosystem. After filtering, roof just transfers the beneficial collated data to fog/cloud servers. Thereby roof increases backhaul connectivity performance. The roof could also keep the data locally when the fog /cloud interconnection breaks, and upload the data once the connectivity is restored. In addition, roof enables unified services capable of context-building and defining additional services that provide multiple device-wide decentralized user experience. Again, allowing end users to be conscious of privacy concerns before selecting a service gives the service providers reputation. Throughout this framework, this overall service, device, and

network management between several RC networks is managed by OpenFlow enabled SDN switches.

3.3.3 Role and Functionalities of Fog and Cloud Computing

In SD-DRFC framework, the key concentration is given to DC and RC. The fog and cloud here performs the basic functionality for which they were originally designed. For this framework, FC offers the capabilities for computation, connectivity, management, temporary storage, and facilities at the midpoint of the network. FC layer consists of a wide volume of FNs. These FNs are widely spread among the RC and the cloud. They could be static at a single position, or mobile on a moveable transporter. The RC devices are able to communicate comfortably with FNs to get services. The FNs are capable of calculating, transmitting and temporarily storing the received data. Fog layer can achieve the near-real-time computation and delay-sensitive services. In addition, the FNs are also linked to the cloud server by IP network and are liable for interaction and collaboration with cloud to acquire extra potent computational power and permanent storage.

One significant function performed by FC in this framework is data fusion. Data fusion is the method of combining data from various sources to generate information which is far more balanced, correct, and beneficial than that delivered by any single data source. The use of signal analysis and data fusion methods like features extraction, development plan and principal component analysis (PCA) to evaluate such sensory data will tremendously enhance the positive rate of classification of the device's motion and related contextual status. Thus the contextual awareness (which was initially done by RC layer) will be considerably improved in the FC layer. FC platform provides location-based mobility requirements and allows administrators to manage (in this framework, through SDN) from where clients and devices come and how they obtain the data. That enhances system efficiency and QoS. In this framework, FC needs to perform fewer computing and data transmission duties compared to traditional IoT-Fog-Cloud architecture. As we know, the lower the amount of storing and transmitting data, the lower the power consumption. So in this framework, FC will not produce a lot of temperature and need no extra cooling system which was mandatory for conventional IoT-Fog-Cloud architecture. Additionally, short-range connectivity method and some optimized power management initiatives of sensor nodes obviously reduce energy usage in communication. This would result

in lower power usage, energy savings and cost reductions. Thus FC offers a greener model of computing.

For this framework, the CC layer is composed of several highly efficient servers and storage devices to enable intensive computational processing and a large volume of data being permanently stored. Based on the speed of the network and the server loads, CC could take quite a long time to respond. However, unlike conventional CC architecture, not all processing and storage activities go to cloud under this framework. Data-related computing and storage tasks (initial filtering, preprocessing, further simple processing, local storage, temporary storage etc.) are performed by DC, RC and FC before the data is sent to CC. Therefore, storage and computational pressure on CC should be comparably much lower in this framework as compared to conventional CC. The cloud core modules are handled effectively, depending on the demand load, and can be planned by control mechanisms to maximize the usage of cloud resources. This makes it possible to provide more services to more users with limited CC resources.

3.3.4 Role and Functionalities of SDN

Existing network components (e.g. routers, switches, firewalls, etc.) are narrowly manageable and programmable. This creates a high obstacle for network modifications. SDN is adopted to erase those obstacles. SDN sets up two major operational components in the scene to overcome these network obstacles. These are SDN controller and SDN switch, and a monitoring connectivity protocol among these two. The traffic management operations were removed from the SDN switch and integrated into the SDN controller. The SDN controller is the central management component that administers a number of SDN switches across an SDN protocol. For this framework; consideration is given to OpenFlow protocol. Roof and fog devices in various physical spots are linked via these switches activated by OpenFlow and give access rights to their users. The SDN controller does have the ability to visualize the underlying network in a general manner. The mapping and routing activity of network components can be queried and altered in the SDN controller based on customized regulations. The SDN controller determines the activity to be performed on a data packet, and will add the appropriate flow rule on the relevant switch for future implementation of the same activity on similar packets.

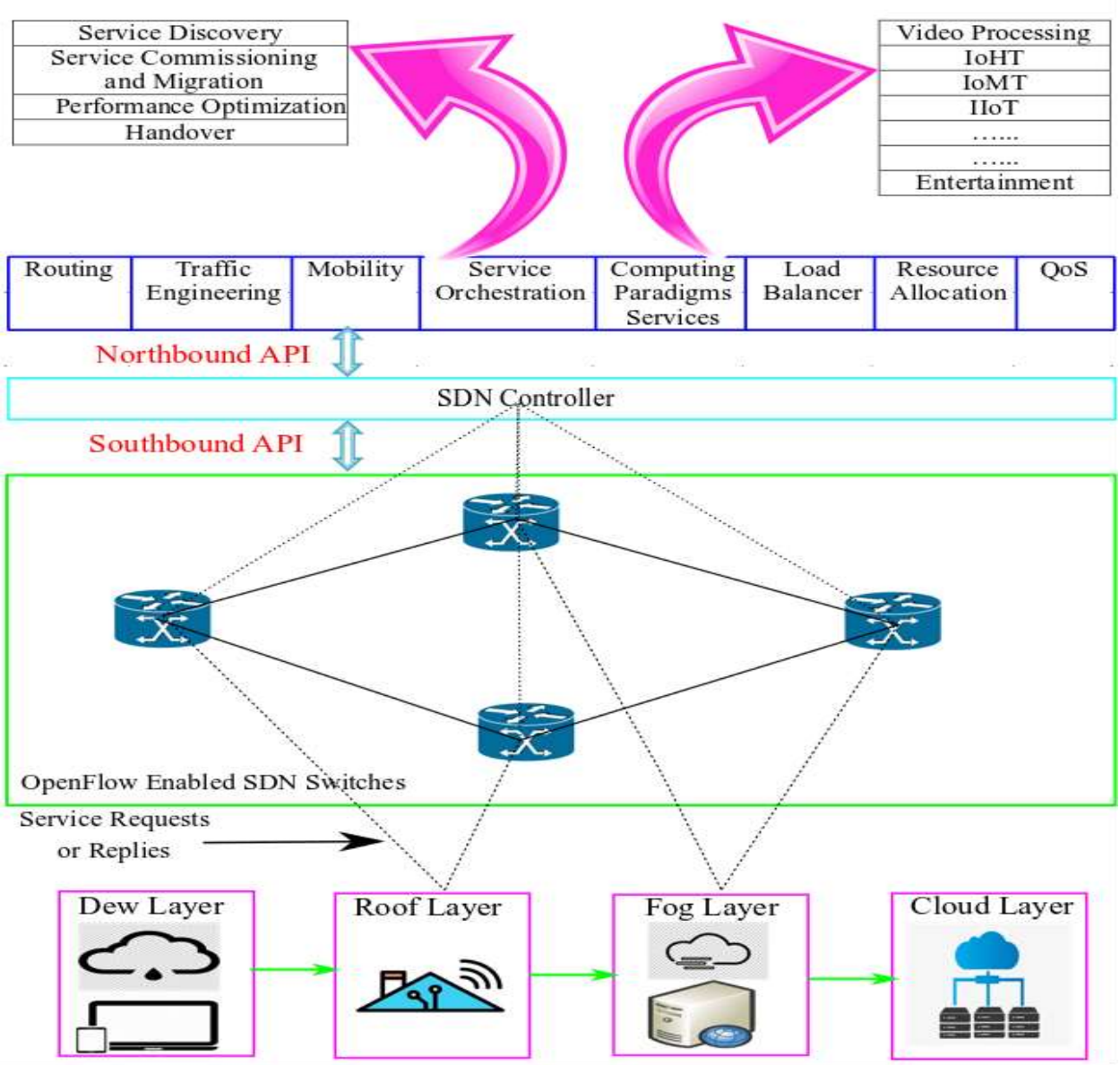


Figure 3.8: SDN Architecture for SD-DRFC Framework

The core SDN controller, which is invisible to the end-user, performs data flow monitoring, service orchestration, as well as other administrative duties of this framework. The controller contains a range of SDN northbound apps to perform the essential activities and convey the basic act for the entire governance and orchestration of the services. All of these applications, if either they reside to an edge service (computing paradigms services) or to the orchestration feature, interacts with the SDN controller via the northbound API and produces instructions in response to an arriving event transmitted from switches to the controller. Now these high-end instructions are assembled and transformed by the controller into low-end OpenFlow messages. These

messages are then need to be submitted to the switches and there the operation is discovered. The originality of this modified SDN structure is defined by upper layer, consisting of modified, virtualized and customized northbound apps that describe the control mechanism's activities. Service orchestration, optimum allocation of resources, mobility management, routing, load balancing, traffic management, and QoS are the major functions and features of these apps. Probable modules in the application of service orchestration are: service discovery, service commissioning & migration, performance optimization, and handover [145]. Throughout RC and FC, SDN can help handle the heterogeneous networks effectively.

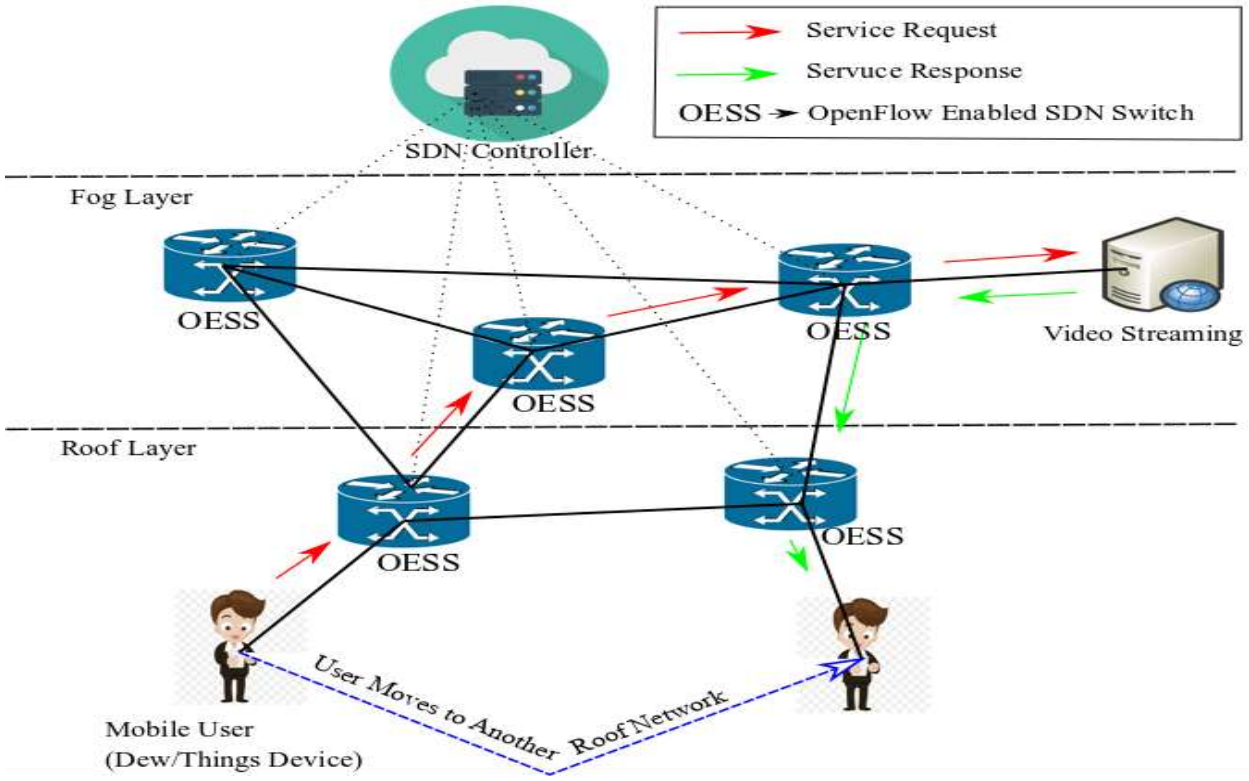


Figure 3.9: Performing Handover using SDN

Because of mobility an end user device may exit a server scope and join another one. The user begins off with service discovery and commissioning processes every time the server is switched on if there is no involvement of handover management. It will not only provoke performance interruptions as encountered by an end-user, but will also lead to deficiencies for computer infrastructure functionalities. Handover procedure utilizes relevant methods to predict the next

coverage state and can use different methods to provide constant services to users like flow redirection or quick live migration of related virtualized resources via SDN. To clearly understand the handover operation by SDN; an example scenario is depicted in Figure 3.9. A mobile user (either things or dew device) resides under a RC network. The user then transmits a request for service to one of the fog layer servers that offers video streaming services. Prior the server completes the request; the user is moved by switching the location to another RC network. The controller could still monitor this migration with its capability to realize the topology and obtain the required details about the recent location of the user, like the IP address that is currently assigned. This enables the finding of the service to be achieved by introducing new routing protocols in the route to the switches. The user is not conscious about the functions happening inside the network throughout this whole process, and therefore the user experience is not disrupted.

As substantial quantities of fresh IoT devices are being added every day; therefore the function of load balancing becomes a critical problem. Within this proposed framework, SDN conducts load balancing. A load balancer must consider both issues (i.e. networking and computing resources) to maximize the system's overall performance. VM migration is a strategy usually used on servers for efficient operation in terms of power costs and allocation of loads [146]. SDN is a perfect arrangement for handling the VM migration procedure, with a central sight of the framework. Due to its ability to reduce idle time throughout migration, protect congestion, enhance throughput and identify the position for greatest performance, SDN is beneficial for these mentioned task too. Afterward a VM migration, SDN controller could track the network, identify a probable congestion and alleviate it by adding new flow principles. VM migration further stands to offer power-wise benefit to this proposed framework. Thus it impacts the overall QoS of the IoT ecosystem.

3.3.5 Layer-Wise Functionality and Computing Operation

From the above study, we have seen that each of the layers of proposed framework has been designed with predefined functionalities relevant to the framework’s data transmission and processing pipeline. To facilitate seamless computation of grasped data, the layer-wise data transmission and computation operation flowchart for this SD-DRFC framework is given in Figure 3.10.

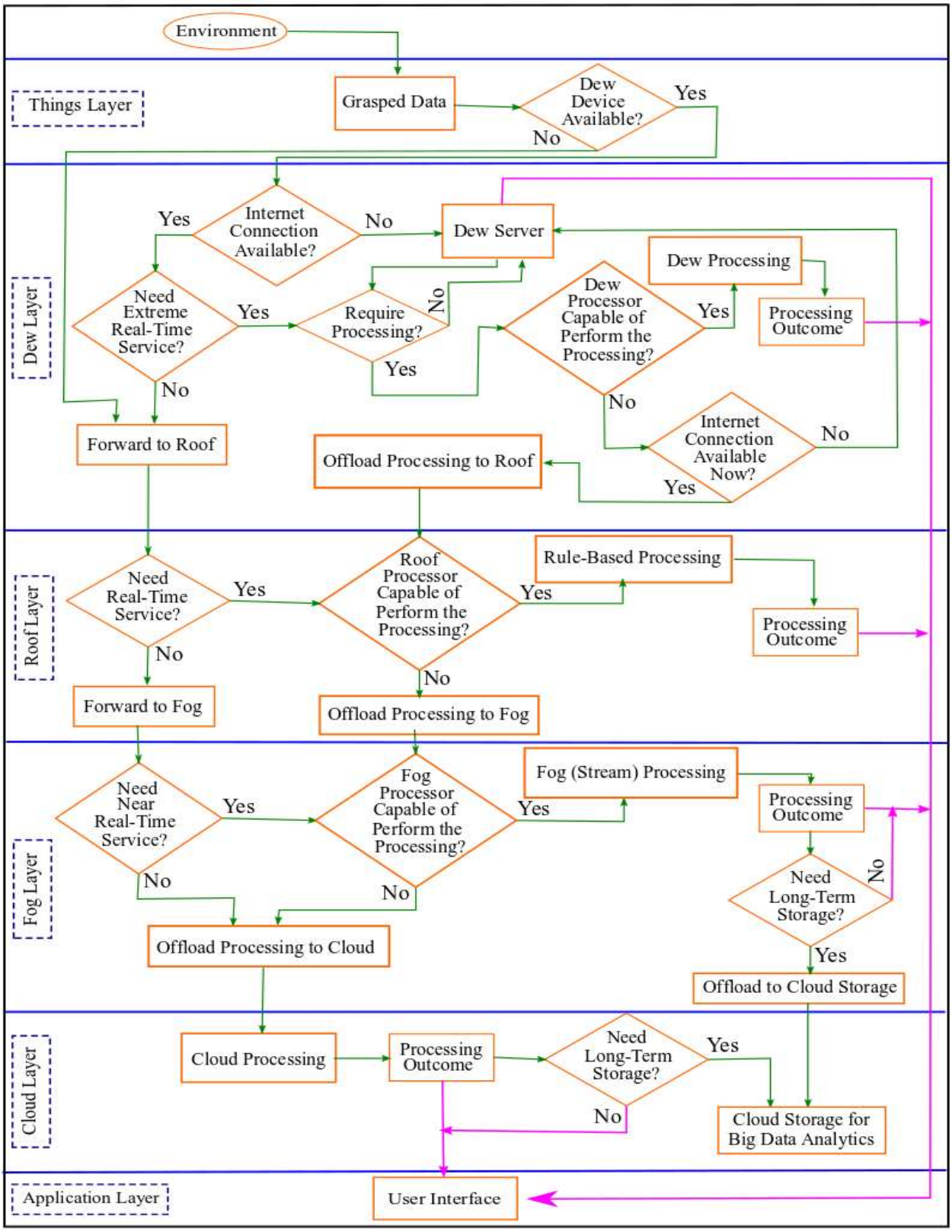


Figure 3.10: Flowchart of the Layer-Wise Data Transmission and Computation Operation

The things layer grasped four possible types of data in terms of delay sensitivity. These are: extreme real-time (service provided by DC), real-time (service provided by RC), near-real-time (service provided by FC), and delay insensitive data (service provided by CC). In order to execute computation at the nearest possible layer, to achieve better QoS, reduced latency, and optimized power consumption; separate transmission paths for each type of data have been used. Based on data traffic and resource and internet connection availability, the computational loads are delegated to an appropriate layer (DC, RC, FC or CC). This resulted in appropriate computation operation in appropriate layer and thereby ensures perfect QoS for an IoT ecosystem.

3.4 SUMMARY

In this chapter, we have presented two novel approaches to improve QoS of IoT ecosystem. For Method I, we have presented an architecture and algorithm which provides better QoS to IoT ecosystem. The Method II presents a novel SD-DRFC framework which provides services from nearest possible layer and thereby enhances IoT ecosystem's QoS. Both of these approaches, their procedures, their roles and their functionalities are elaborately explained and presented with appropriate diagrams in this chapter.

CHAPTER 4

SIMULATION SETUP

4.1 INTRODUCTION

In this chapter, simulation setup is performed based on the two methods which we have presented in “Methodology” chapter. First simulation setup is performed on “Method I” and then simulation setup is performed on “Method II”. The required use case, application model, and device configurations are presented in detail for both cases.

4.2 SIMULATION SETUP ON COMBINED FOG AND SDN (METHOD I)

A use case is defined for simulation, based on propounded architecture and algorithm of “Method I”. The use case is simulated in iFogSim [147] to determine the efficiency of the proposed policy. The propounded architecture contains four layers which are interlinked with one another and the propounded algorithm defines functionality for that architecture. Therefore this simulated use case is also a 4 tier architecture where digital cameras acts as things layer and the rest layers functionality are as same as propounded architecture. Link such cameras to initial gateway where initial gateway acts as FN of propounded architecture. Then all initial gateways are connected to an upper gateway where upper gateway acts as AFN of propounded architecture. SDN defined as SDN classes in iFogSim import rules to the simulated use case. Fog and Cloud devices from iFogSim classes are utilized and moderated for proper execution of propounded use case. iFogSim doesn’t support mobility. Besides being unable to support the mobility, iFogSim still suffers from a few other obstacles. These include: Power-aware resource planning regulations, priority-aware resource planning strategies for multi-vendor ecosystems,

modeling fog failures, dynamic prioritization and SLA-conscious stream placement and resource utilization (combined Edge-network operational efficiency), modeling and comparison of various IoT ecosystem virtualization technologies, scheduling application configurations. Due to these obstacles, the proposed algorithm can't be included in simulated use case fully. Thus some portion of the algorithm is compromised like mobility, location of SDNs etc.

4.2.1 Simulated Use Case

Video surveillance is a big part of smart cities. Large-scale cameras in a community or along the road carry safety programs to a higher level, giving the public a strong sense of security. It is also highly important in "Smart Detection and Tracking of Criminal Activity". That is the simulated use case for "Method I".

There are two benefits to using the new architecture to intelligently identify and monitor criminal activity. The first is that on the basis of QoS criteria, SDN controller decides on routing and resource allocation. So, it'll always search the route that can fulfill video flow bandwidth specifications. The next benefit is that when any event is find out, resource-rich FNs might supply real-time video frame processing and emits end user notification. This dramatically decreases network bandwidth usage, thus increasing device performance (e.g. criminal activity detection). To track a given area the device combines umpteen cameras with various fields of view (FOVs). Coordination between cameras requires synchronized tuning of PTZ parameters, aiming to obtain ideal view of zone. When a person is detected based on the information stored in the law agencies criminal record database (e.g. image, birth mark, etc.), the system alerts the user which could require security authorities to pay attention. The smart camera senses movement in FOV and starts to send a video stream to the application of criminal detection and monitoring. In the video stream sent, the application locates the moving suspect and initiates the monitoring. Tracking moving criminals is achieved by continuously changing the cameras PTZ parameters at that location to get the best view of the criminals being monitored. In addition, the application notifies the system user in case of identification of an event of interest and transmits recorded video streams to user over Internet.

As the criminal travels then the limited coverage of the FN may not be enough for tracking. The criminal may start running or use a vehicle to skip the coverage. AFN steps in that case. The

AFN is put at a higher grid hierarchy, and is therefore much broader in scope. This runs processes which recognize running criminal or vehicles in frame and find out their (For Vehicle) location and license plate. Since the law agencies are interested in the vehicle identified, the vehicle location information is registered in the database and notification should transmit to highest hierarchical step in the application process. If, however, the picture on video isn't pretty clear to allow the license plate number to be detached, the AFN could submit PTZ commands to camera. PTZ module contact does quite small latency need, which can't be supported when video processing is performed on distant cloud server. Figure 4.1 shows the physical topology of the simulated use case in iFogSim simulator.

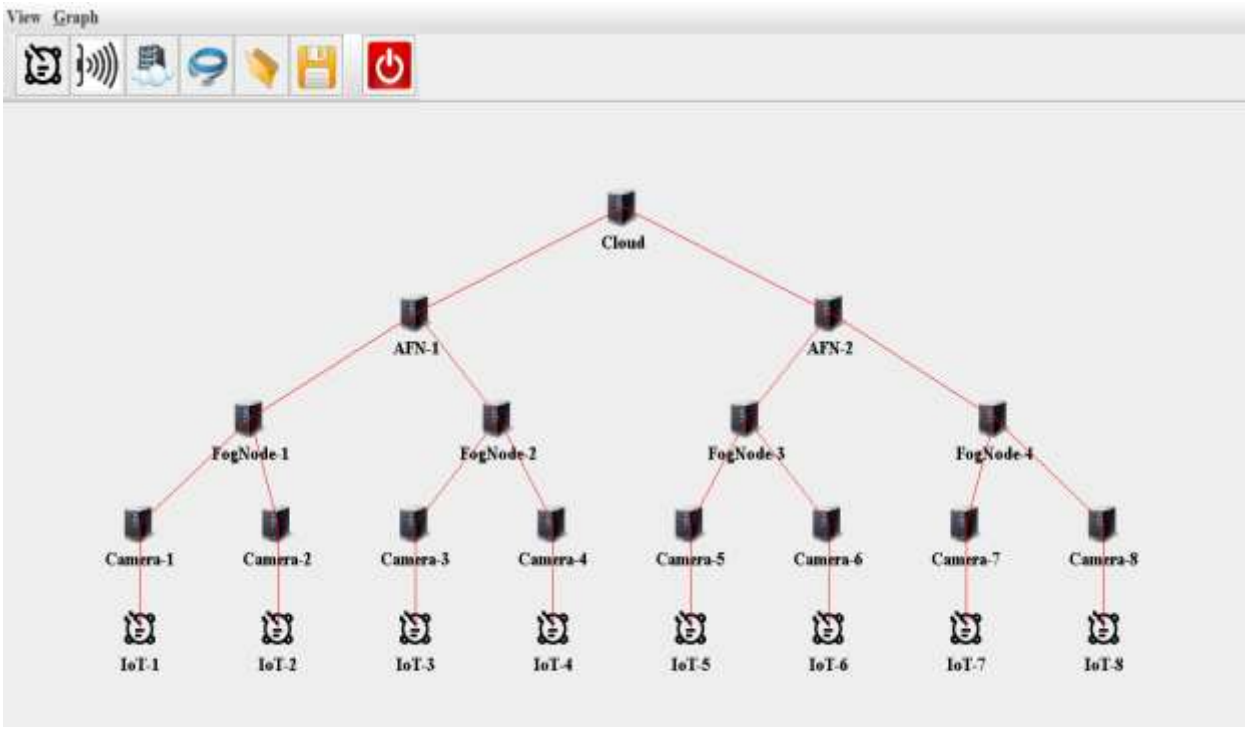


Figure 4.1: Physical Topology of the Simulated Use Case of Method I

4.2.2 Application Model

The "Smart Detection and Tracking of Criminal Activity" application is made of five leading modules which process: Criminal Motion Detector, Criminal Detector, Criminal Tracker, PTZ Control and User Interface (Depicted in Figure 4.2). A variety of CCTV digital cameras feed

live video streams into detection and tracking application and PTZ commands in cameras changes PTZ parameter continuously. The functionality of the modules is the following:

Criminal Motion detection: This module integrated within digital cameras applied in the use case. It scans streams of raw video which the camera captures continuously to trace motion of a person whose information is stored in criminal record of law agencies database. The stream of video is upstream sent to criminal detection module owing to more analysis in case of motion detection in the camera's FOV.

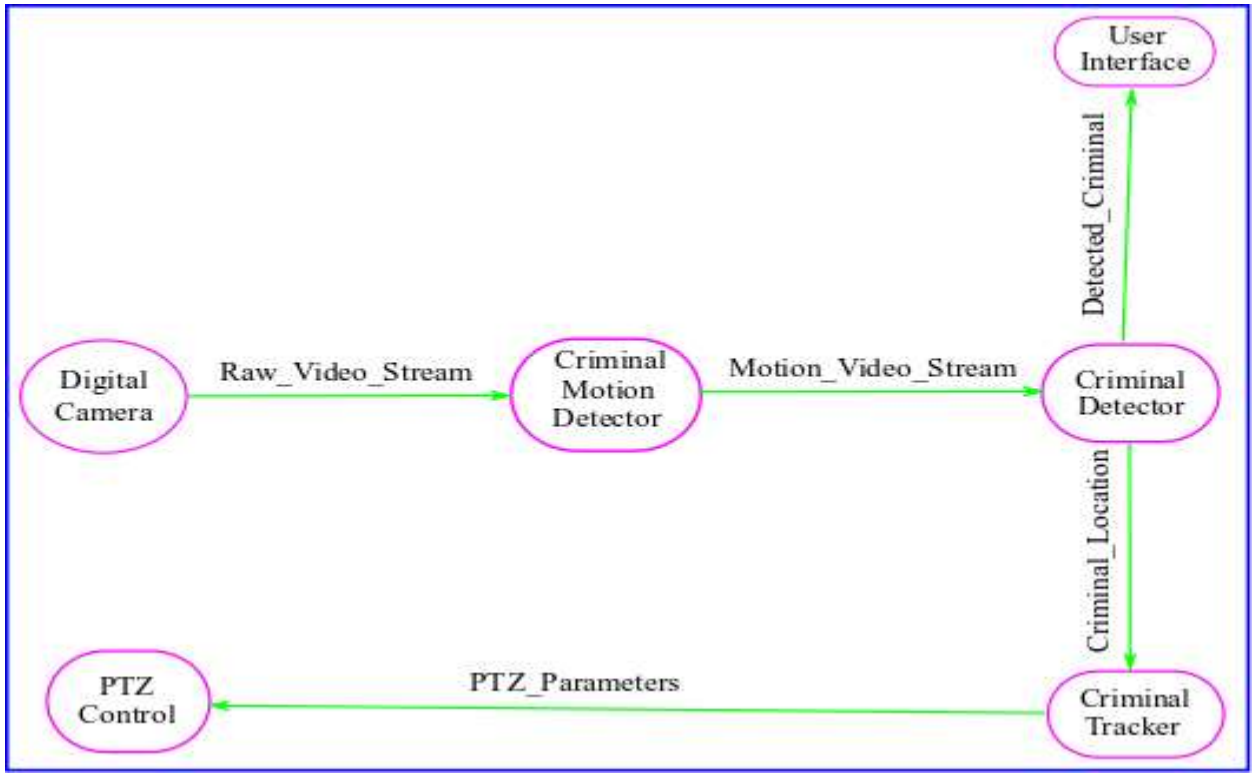


Figure 4.2: Application Model of Smart Detection and Tracking of Criminal Activity

Criminal detection: Criminal detection module takes in stream of video where a person's motion is detected by the smart cameras. The module extracts the moving person from stream of video and does compare with them with information formerly stored. In the event that the identified person has details in criminal record, monitoring for that person is triggered. Additionally it calculates the person's co-ordinates.

Criminal tracker: The criminal tracker module collects the currently monitored person's last estimated coordinates and determines an optimum PTZ system for all those cameras that coating the zone so as to the tracked person can be identified by maximal feasible way. This PTZ knowledge is regularly transmitted to cameras regulated by PTZ.

PTZ control: This module paces on every digital camera and changes the bodily camera to suit optimum PTZ setups that are sent by the object tracking module. This module acts as system's actuator and is integrated into smart cameras themselves.

User interface (UI): The application provides UI by emitting to computer of user a minimal of stream of video which contain every tracked individual. For this use case the criminal detector module includes these filtered video streams.

The CPU length and network length presented in Table 4.1 are practically the properties of tuples transported by edges within device modules.

Table 4.1: Tuple Types with CPU and Network Length

Tuple Type	CPU Length(MIPS)	Network Length (Byte)
Raw_Video_Stream	1400	18000
Motion_Video_Stream	2500	2800
Detected_Criminal	800	1600
Criminal_Location	1500	400
PTZ_Parameters	200	400

Tuples construct the basic unit of connectivity among objects in the Fog and recognize the architectural data stream layer. In iFogSim tuples are being defined as tuple class cases. A tuple is defined by its form, as well as the application modules for the sender and receiver. The CPU length (first tuple property) defines the processing specifications that are specified as Millions of Instructions Per Second (MIPS). The network length (second tuple property) determines the length of data embedded inside the tuple. This is specified in Bytes Unit. Table 4.2 determines the latency of devices used in fog, from source to destination. The configuration of used fog apparatus in case study is shown in Table 4.3.

Table 4.2: Latency of Used Fog Devices

Source	Destination	Latency (ms)
Upper Gateway	Cloud	80
Initial Gateway	Upper Gateway	4
Digital Camera	Initial Gateway	2
Camera (Sensor)	Digital Camera	1
PTZ Control	Digital Camera	1

Table 4.3: Other Configurations of Used Fog Devices

Device	MIPS	RAM (GB)	Uplink Bandwidth (Kbps)	Downlink Bandwidth (Kbps)	Busy Power (W)	Idle Power (W)
Cloud	52000	36	250	12500	10*10	10*10
Upper Gateway	4400	5	10000	10000	80.9	60.33
Initial Gateway	4400	2	7500	7500	80.9	60.33
Digital Camera	1500	1	8000	220	40.5	30.2

The cloud power consumption for busy and idle state is carried out through same method named power nap method. It's basically a built-in method for cloud in iFogSim and used for data consumptions. Since it's a built-in power nap method in iFogSim, therefore it can't be customized. In addition, for idle and busy state, the method is same. So this can be marked as a limitation of iFogSim. Herein the cameras which record live feeds of video function namely sensors yet provide the application with input data.

4.3 SIMULATION SETUP ON SD-DRFC FRAMEWORK (METHOD II)

A use case is defined for simulation, depending on the architecture provided in “Method II” in the “Methodology” chapter. The use case is tested via the simulator iFogSim to evaluate the efficacy of the architecture provided. The architecture presented in this article comprises four layers that are interconnected with each other. Therefore a 4-tier use case, which is similar to the architecture described, is also the use case provided here.

4.3.1 Simulated Use Case

A latency-critical game called “EEG Tractor Beam Game” is the presented use case here. This game is a human-vs.-human game that involves augmented brain-computer interaction. Each player required to wear a wireless EEG headset which is linked to his or her smartphone to play the game. The game runs on a user's smartphone as an Android application. The application analyzes the EEG signals detected by the EEG headset in real-time and measures the user's brain state. The game depicts all players on a ring surrounding a target object on the application's monitor. In proportion to his level of focus, each player will create an appealing impact on the target. A player must attempt to drag the target towards him in order to win the game by practicing focus while trying to deprive other players of their possibilities of seizing the target. Processing in real-time allows the application to be installed very close to the data source (i.e. the real-time services should be provided), preferably on the smartphone itself (i.e. in the dew computing layer). Such an implementation, however, does not allow global coverage, which usually requires the application to be deployed in the cloud. Such a mix of conflicting objectives makes this application a typical use case for combined dew, roof, fog and cloud computing architecture.

Now how this presented use case is identical to the presented architecture of “Method II” is described here. Each EEG headset is connected to a smartphone via Bluetooth communication link. EEG headsets are acts as things layer devices in this use case. These headsets are connected to the smartphone that serves as dew computing layer. Smartphones gain access to the Internet through Wi-Fi gateways i.e. all smartphones are connected to the Wi-Fi gateways. Here Wi-Fi gateways are acts as roof computing layer. All Wi-Fi gateways are then linked to a higher gateway called ISP gateway where the ISP gateway serves as the fog computing layer. Finally all

the ISP gateways are connected to the cloud and perform historical data processing in cloud data center (CDC). In this use case SDN basically performs the allocation tasks. SDN is defined as SDN classes in iFogSim's import regulations. BwProvisionerSimple is an SDN class in this examined use case which imposes a standard best effort allocation policy. This class allocates when bandwidth is sufficient for request; otherwise it fails. One other SDN class which allocates MIPS by Virtual PeId for all Virtual Machines (VMs) is PeProvisionerOverbooking.

4.3.2 Application Model

The application Game "EEG Tractor Beam Game" is composed of 3 key modules for carry out the processing. These are: Consumer, Focus Calculator, and Coordinator (depicted in Figure 4.3). Within iFogSim, the application model modules are formulated utilizing the AppModule class. There have been data dependency among modules, as shown in Figure 4.3, and these dependencies are formulated using iFogSim's AppEdge class. After that, iFogSim uses the AppLoop class to model the control loop for the EEG game. The application is supplied with EEG signals via an EEG sensor and the actuator called DISPLAY shows the consumer the present scene of the game. The responsibilities of the modules addressed above are:

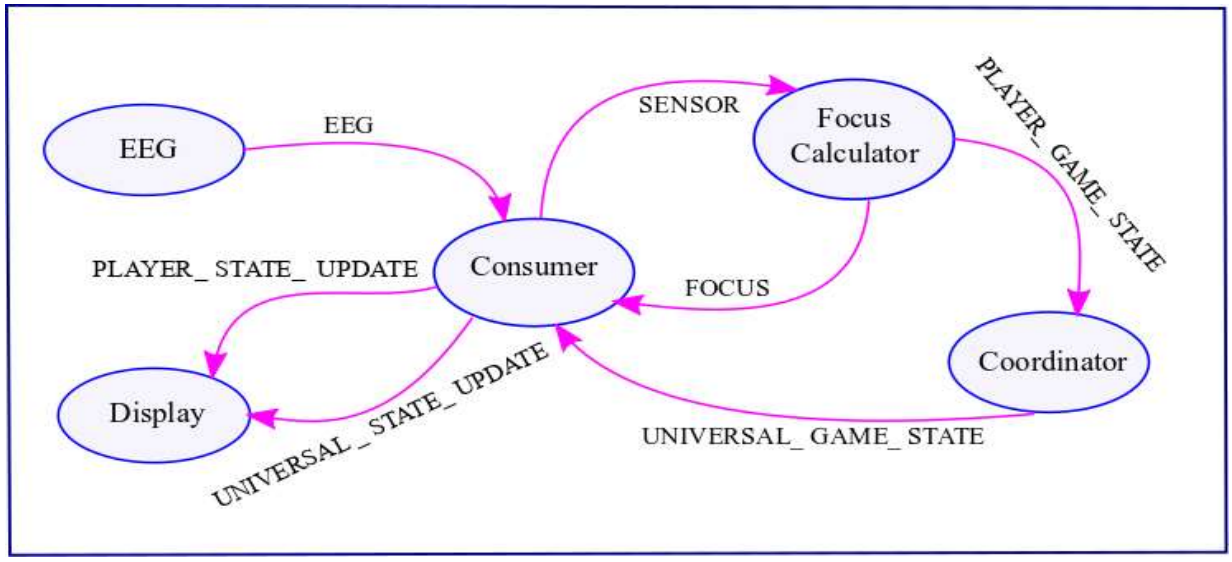


Figure 4.3: Application model of the EEG Tractor Beam Game

Consumer: The consumer module interacts and collects raw EEG signals from the sensor. For any anomaly, it tests the obtained signal values and removes the potentially inaccurate data. If

the quality of the perceived signal is accurate, it delivers the data to the Focus Calculator module to get the user's high degree of concentration from the signal. It shows the data when obtaining the concentration of consumer by submitting the data to the DISPLAY actuator.

Focus Calculator: It is the task of the Focus Calculator module to assess the user's brain-state from the perceived EEG signal properties and measure the concentration level. This module tells the Consumer module of the calculated concentration level, so that it is possible to update the player's game status on the monitor.

Coordinator: The Coordinator functions geographically and incorporates the game among several players who may be participating at locations that are widely separated. The Coordinator delivers the present status of the game constantly to the Consumer module of all related consumers.

Table 4.4: Configuration of Attributes of Tuples for EEG Tractor Beam Game

Tuple type	CPU length (MIPS)	Network length (Byte)
EEG	2000	500
SENSOR	3500	500
PLAYER_ GAME_ STATE	1000	1000
FOCUS	14	500
UNIVERSAL_ GAME_ STATE	1000	1000
UNIVERSAL _ STATE_ UPDATE	1000	500
PLAYER_ STATE_ UPDATE	1000	500

Table 4.5: Specifications of Devices for EEG Tractor Beam Game

Device type	CPU (GHz)	RAM (GB)	POWER (W)
Cloud Data Center (CDC)	3.0	4	107.339(M) 83.433(I)
WiFi Gateway (Fog Device)	3.0	4	107.339(M) 83.433(I)
ISP Gateway (Roof Device)	3.0	4	107.339(M) 83.433(I)
Smartphone (Dew Device)	1.6	1	87.53(M) 82.44(I)

Table 4.4 defines the attributes of tuples (formulated via Tuple class) which are represented by edges among the modules in the application. The specifications of the respective device categories which are utilized in the presented use case are outlined in Table 4.5. The efficiency of the network edge application (i.e. on the dew/roof/fog) relies on the latencies of the connections attaching the system devices. Various sorts of devices have separate latencies among them in the simulation topology, which are described in Table 4.6.

Table 4.6: Description of network links for EEG Tractor Beam Game

Source	Destination	Latency (ms)
EEG headset	Smartphone	6
Smartphone	WiFi gateway	2
WiFi gateway	ISP gateway	4
ISP gateway	Cloud Data Center	100

4.4 SUMMARY

Above the simulation setup for this thesis's two methods are presented. Both of these setups are performed on the iFogSim simulator. Based on these setups, we have evaluated our proposed methods. Performance evaluation on the proposed methods is done by these setups to find out the efficiency of these proposed approaches.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 INTRODUCTION

In this chapter, the results we have obtained from our simulation is presented and discussed. Two methods are proposed in this thesis to improve the QoS of IoT ecosystem. Based on these two methods, two simulation setups are performed in “Simulation Setup” chapter. In this chapter the results of the simulation on “Method I” is presented first. Then the results of the simulation on “Method II” are presented. After analyzing the results, it can be seen that the proposed methods achieve significant improvement in QoS of IoT ecosystem.

5.2 RESULTS AND DISCUSSION OF COMBINING FOG AND SDN (METHOD I)

For the configuration of fog devices, hierarchical structured network topology has been utilized in this simulation and the various metrics reported by iFogSim have also been compiled. The simulation results show how various workloads and positioning strategies of inputs affect the QoS matrices. In this simulation, the amount of zones controlled by the system is varied from 2 to 12. There are also different numbers of cameras in each of the controlled zones to control the covered area. Link those digital cameras to preliminary gateway (PG) which handles activities in same zone. Then all preliminary gateways are connected to a higher gateway (HG) that covers the underlying multiple FN region. The amount of zones monitored and the number of cameras per zone differ across physical topology configurations. The configurations together with their controlled zone and cameras per zone are: Setup 1 (2, 2); Setup 2 (2, 4); Setup 3 (4, 2); Setup 4 (4, 4); Setup 5 (8, 4); Setup 6 (8, 6); Setup 7 (10, 4) and Setup 8 (12, 4).

iFogSim supports two scenarios namely cloud only and edge-ward placement. The cloud-only deployment strategy concentrates on the traditional implementation of cloud-dependent implementations in which all of the components of an application run on cloud servers. In these applications, the percept-process-activate loop is introduced by sending data to cloud that is perceived by sensors. The data is stored and processed in cloud. Then actuators are told when action is needed based on data processing. In comparison, the Edge-ward placement strategy supports the implementation of system modules around the edge of the network. Often devices near the edge of the system might not be computationally efficient enough to accommodate all the application providers. In this kind of scenario, the approach propagates towards cloud and aims to put the leftover operations on cloud server devices. This approach highlights the interrelationship among fog and cloud by positioning modules close to the edge of the network. Therefore cloud-only and edge-ward (termed as "Fog + SDN" in this paper) are two placement strategies that are conducted in this simulation for setting presented application modules over constitutional network. In the first strategy, all application operations are put in cloud data center with the exception of the criminal motion identification module, which links to digital cameras. However, in the second strategy, the criminal identification and criminal tracker modules are impulse to PGs.

As mentioned above, iFogSim supports two scenarios namely cloud only and "Fog + SDN". Fog includes SDN. In iFogSim, fog network is implemented using some SDN strategies. Without those SDN strategies, execute only fog isn't possible. Because these SDN classes basically performs the allocation tasks. Therefore the two scenarios we have considered in this experiment are named: "cloud only" and "Fog + SDN". SDN classes are used to import rules to the setup. BwProvisionerSimple and PeProvisionerOverbooking are two SDN classes that used in the simulation. These two classes are in org.cloudbus.cloudsim.sdn.overbooking package of the iFogSim master package. These two classes are individually assigned and written for their desired task and then imported in the main "Simulator" class. From this "Simulator" class the results (i.e. the QoS parameters) of this simulation are calculated. BwProvisionerSimple is a class that implements a simple best effort allocation policy: if there is bandwidth available to request, it allocates; otherwise, it fails. PeProvisionerOverbooking allocates MIPS for all Virtual Machines (VMs) by Virtual PeId. In addition, some cloudsim classes are extended for creating fog devices. Therefore cloudlets are expanded to perform stream operators that are used to

execute tuples. Now the results of the simulation experiment are presented below. The term QoS contains many parameters. Among these, four most important parameters (Network Usage, Cost, Latency and Power Consumption) are considered in our experiment. The results of these QoS parameters are presented in graphical form and analyzed to evaluate the performance of the presented approach.

5.2.1 Network Usage

Total network usages are shown in Figure 5.1 for both cloud-only and 'Fog + SDN' deployment. The network usage becomes large in cloud-only compared to the 'Fog + SDN' as the controlled zones and cameras per controlled zone rises. Limited network (i.e. the data stream passes reduced amount of tuples) is used by 'Fog + SDN' implementation as it conducts data-intensive activities on small latency connections. Figure 5.1 also shows that for both cloud-only and 'Fog + SDN' deployment, the network usage remains nearly constant from setup 6 and above. The network usage here is taken up in the "Bytes" unit. The reason for this is described in below sub-section (5.2.2).

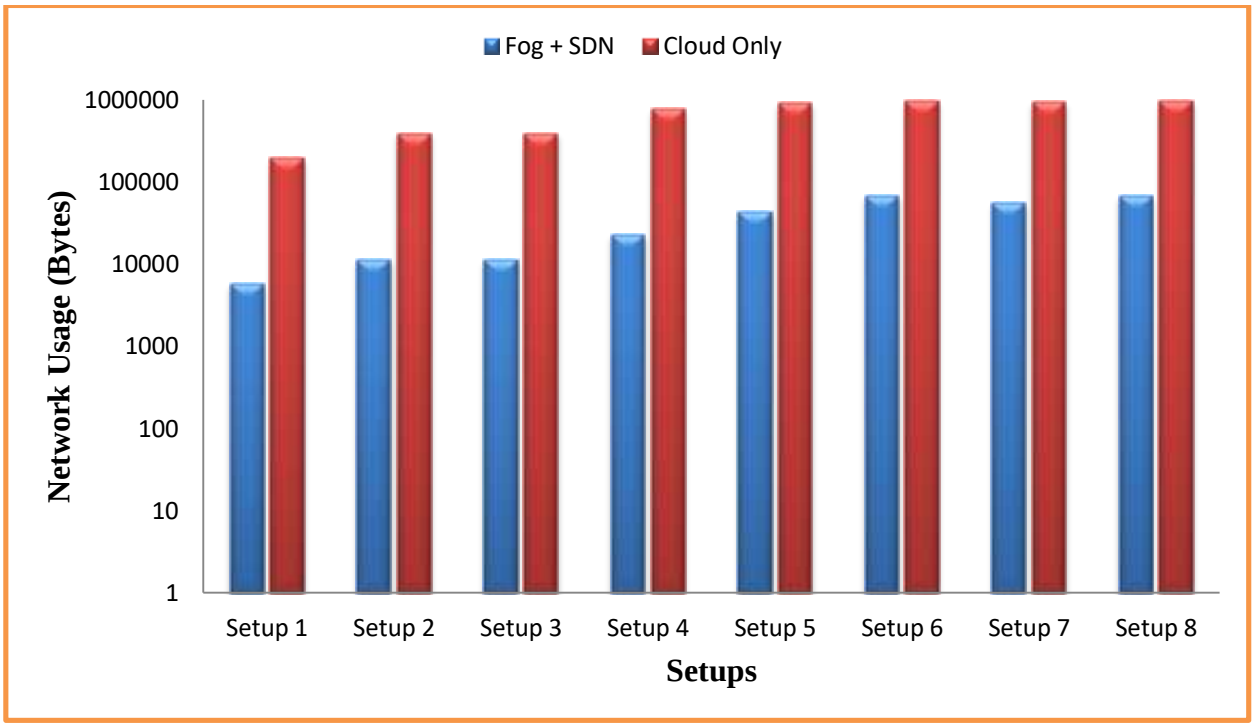


Figure 5.1: Total Network Usage for "Fog + SDN" and "Cloud-only"

5.2.2 Cost

The cost of cloud execution (termed as 'Cost' in this paper) is shown in Figure 5.2 for cloud-only and 'Fog + SDN'. In iFogSim "Cost" is unit less. Since iFogSim basically compares between cloud and fog, so "Cost" is just an indicator about which (fog or cloud) is better based on the numerical value of cost. As seen in the Figure 5.2, once contrasted to 'Fog + SDN', cloud-only costs are far much higher. For example, consider setup 1. Here the value of cost for cloud only placement is 922335.061424276 and for 'Fog + SDN' is 34671.4667. As seen from the value, the cloud-only costs are almost 26 times higher than the 'Fog + SDN'. Again consider setup 2. Here the value of cost for cloud only placement is 1707841.01346953 and for 'Fog + SDN' is 50371.4667. As seen from the value, the cloud-only costs are almost 33 times higher than the 'Fog + SDN'.

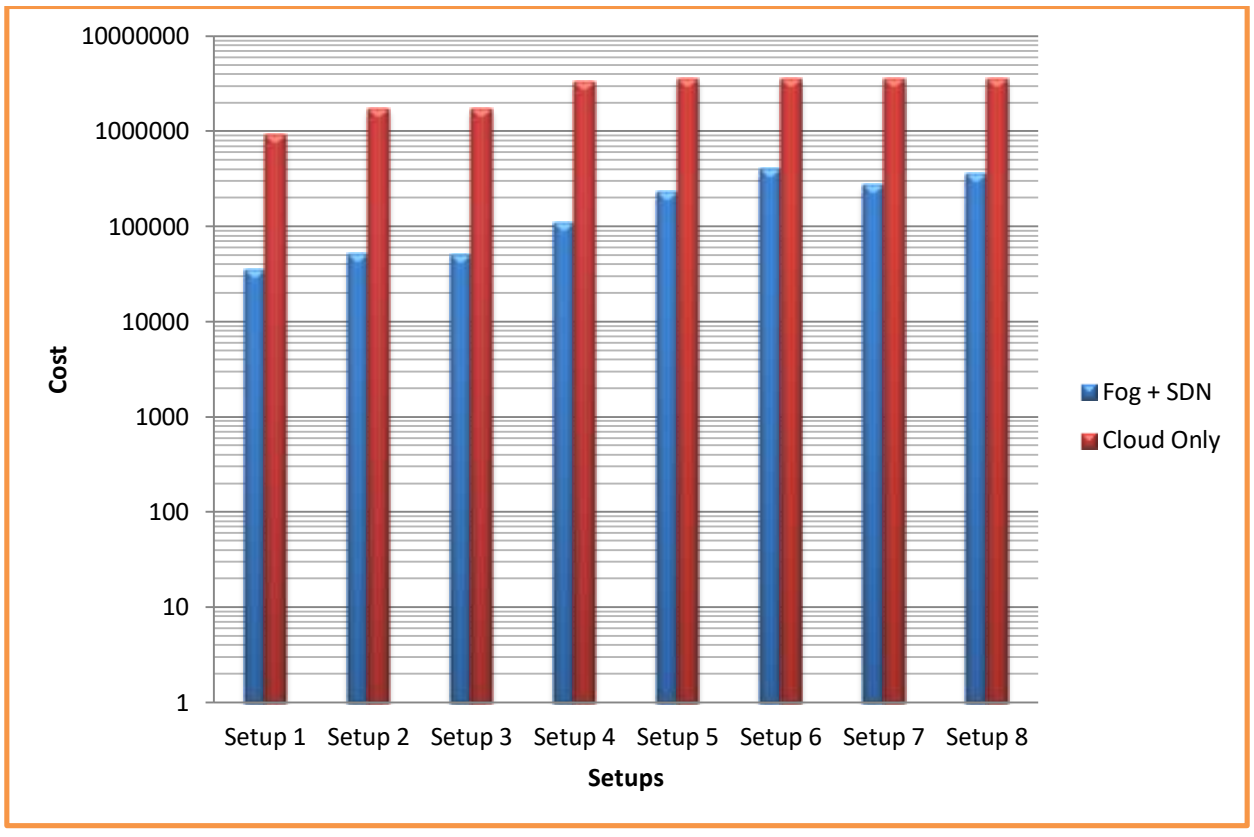


Figure 5.2: Cost of Execution in Cloud for "Fog + SDN" and "Cloud-only"

As well as network usage, from setup 6 and above, the cost is also almost unchanged. This means that when one setups overall configuration (i.e. product of the controlled zone and amount

of cameras per zone) is almost identical to another configuration, then the network usage and costs would also be identical. In addition if one setup overall configuration becomes less compared to the previous one, then both the network usage and cost will reduce. For example, setup 6 has total 48 configurations (8 controlled zone * 6 cameras per zone). Setup 7 has total 40 configurations (10 controlled zone * 4 cameras per zone). Setup 8 again has total 48 configurations (8 controlled zone * 6 cameras per zone). Now we can see that the setup 6 and 8 has the identical number of configuration. Therefore the network usage and cost value for them has also become identical. But setup 7 has less number of configurations compared to setup 6 and 8. Hence the network usage and cost value for this setup is less compared to setup 6 and 8. Therefore in short, if the number of total configuration increases then the network usage and cost is also increases. But if the number of total configuration decreases then the network usage and cost is also decreases. Identical things occurred to other QoS parameters also.

5.2.3 Latency

Figure 5.3 shows the latency for cloud-only and 'Fog + SDN' from video stream to PTZ regulation for this experimented use case. The latency here is taken up in the millisecond (ms) unit. For 'Fog + SDN', for all the amount of controlled zones, the latency stays almost identical and is relatively small when contrasted to cloud-only. That's because the devices are located near to the optimum position for connectivity and execution in 'Fog + SDN', which effectively decreases the latency. In comparison, all the devices had to be positioned in the cloud in the cloud only deployment, which eventually raises the latency.

5.2.4 Power Consumption

For the two cases examined, Figure 5.4 shows power consumption. The power consumption here is taken up in the Joule unit. Essentially in all applications, particularly in energy-powered microelectronics where maximizing battery power is a primary driving force; power consumption is becoming a massive technical problem. The devices from four tires of the proposed simulated use case (but only one device form each tier) is considered for power consumption. These are: cloud data center (CDC), HG, PG, and Camera. HG operated as AFN and PG functioned as FN. Here the power consumption is takes in Joule unit.

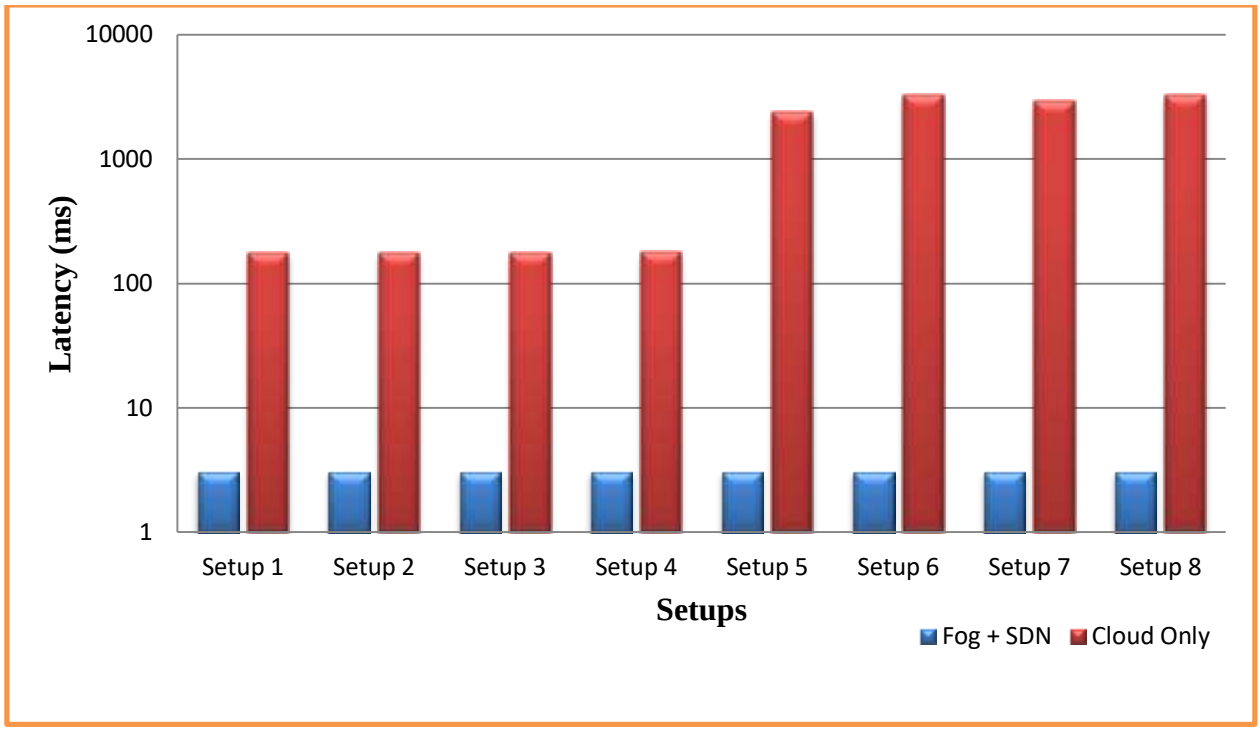


Figure 5.3: Latency for "Fog + SDN" and "Cloud-only"

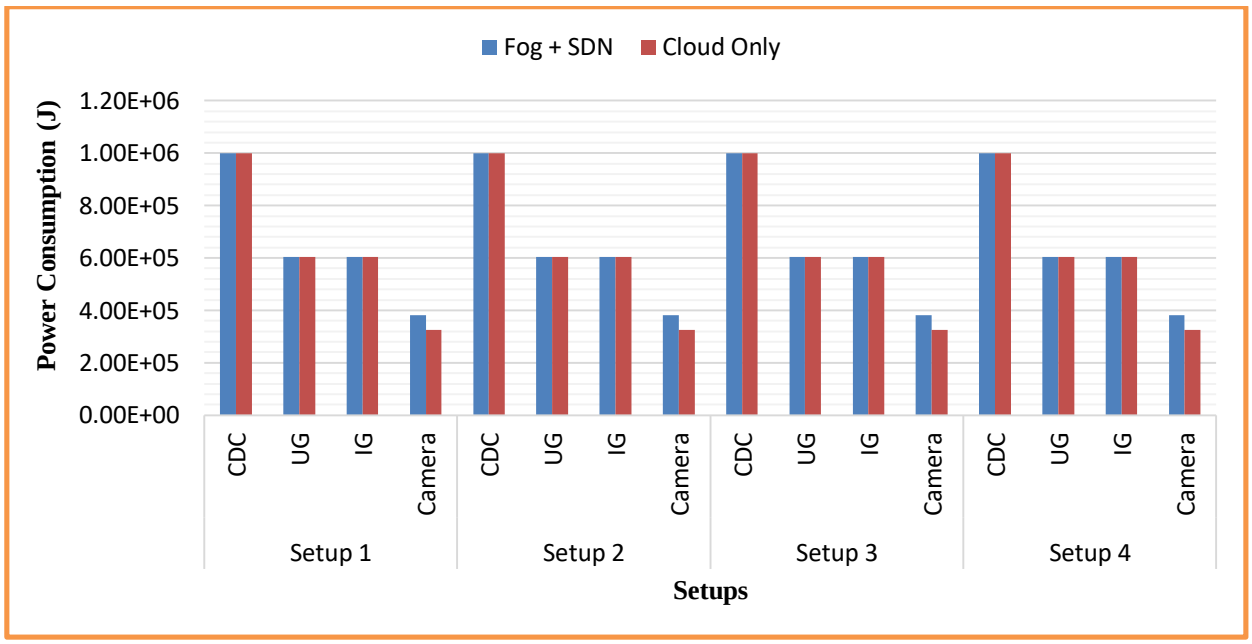


Figure 5.4: Power consumption for "Fog + SDN" and "Cloud-only"

Power usage for CDC, HG and PG is almost the same for cloud-only and 'Fog + SDN', as the figure shows. But the power consumption for the camera (i.e. end device) is higher in 'Fog +

SDN' deployment relative to cloud-only. This is because activities (e.g. mild computing, sending data to upper layer) must be conducted in the end devices in 'Fog + SDN' deployment. This ultimately increases power consumption. Although the previous parameters are evaluated for 8 setups but this parameter is evaluated for 4 setups. Because the remaining setups are also provide almost similar result. In addition, with less number of setup, the presented diagram is way much clearer to understand. When compared to other parameters; here the "Fog + SDN" execution performs lowers than cloud-only. Actually this happens for almost all other existed QoS policies also. But the proposed approach of this paper that used in this simulation is performing better compared to those existed QoS policies and this can only be understood when the comparison is done among the QoS parameter results of this paper's simulation and one other existed QoS policies results.

5.2.5 Comparison of Simulated Use case of Method I and Existed Identical Use Case

Here the comparison is performed among the QoS parameter results of this paper's simulated use case of "Method I" and one other existed QoS policies results. A related example is taken from "Intelligent Surveillance" [153] to compare the results of QoS parameters of our simulation. The intelligent surveillance was performed via distributed camera networks. As the central control methods for evaluating camera-produced data are not suitable mainly due to the extreme volume of data that requires to be transmitting to the central processing system; hence the authors conduct decentralized processing of the video content. The configurations of this presented use case that have the identical number of controlled zones and the same amount of cameras per zone as in the example of [153] (i.e. intelligent surveillance) are selected to make the comparison relevant. That means the results of setup 2, 4 and 5 from the use case of this paper's simulated use case of "Method I" and results of the configuration 2, 3 and 4 from the intelligent surveillance use case are compared here. Setups 2 and 4 with configurations 2 and 3 are contrasted for power consumption, as the following results are almost identical. But all 3 configurations are considered for Network Usage (NU) and Latency.

Figure 5.5 show that this paper's simulated use case of "Method I" reduces serious power consumption in CDC, HG, PG and Camera when compared to the intelligent surveillance use case. Here the power consumption is takes in Joule unit. Figure shows significant less amount of power consumption in CDC in this paper's simulated use case of "Method I" when compared to

previous one, although the two use cases uses the identical setups. The HG, PG and Camera also consume comparatively less energy in this use case than the previous one. This exhibits momentous power consumption improvement is achieved in favor of our presented "Method I" which is one of major aspects of QoS. Thereby power consumption improvement automatically enhances the QoS of the IoT ecosystem.

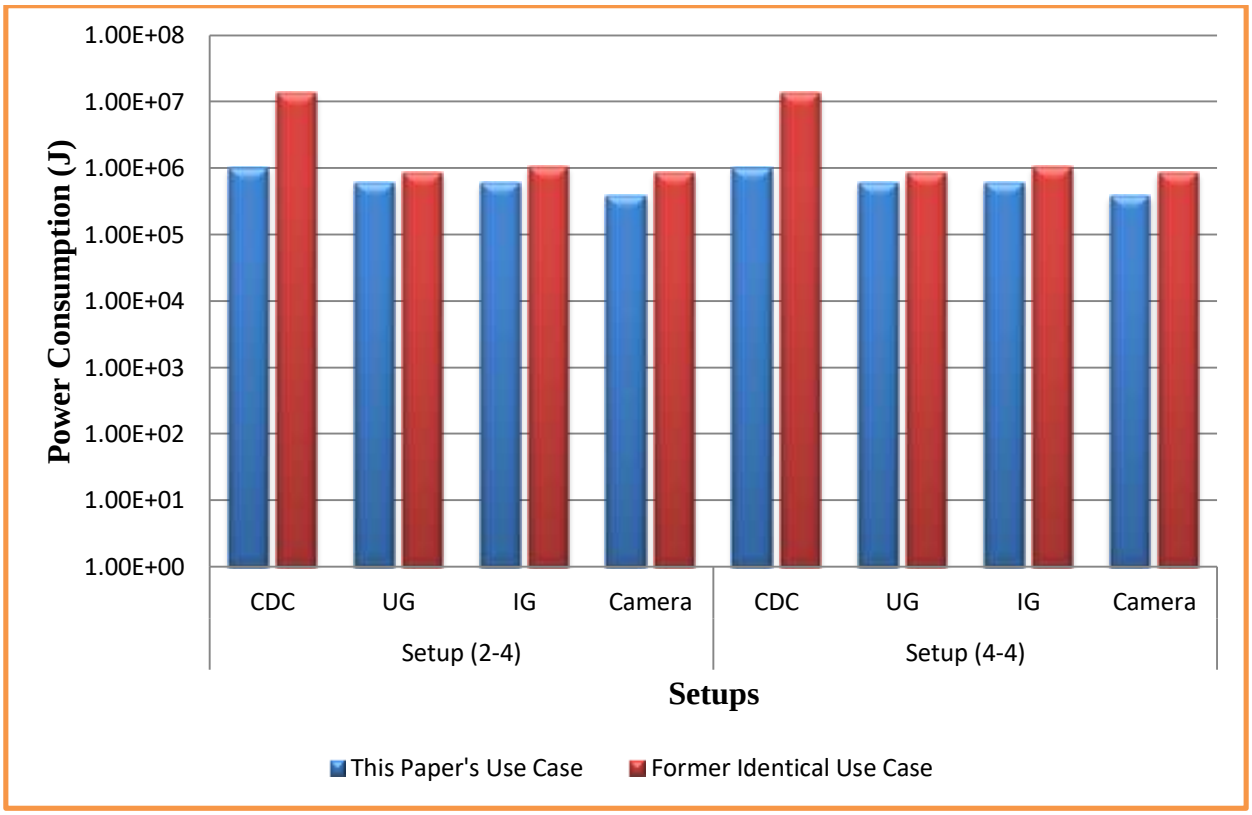


Figure 5.5: Comparison of Power consumption

Figure 5.6 presents the enhanced latency and network usage for this paper's simulated use case of "Method I". The unit of latency is in milliseconds and kilobytes unit are taken by NU. The figure illustrates that both latency and NU for this paper's simulated use case of "Method I" are lowered to about half compared with the previous one. The average latency for our simulated use case, as examined, is 275.9 percent lower and the average NU is 212.21 percent lower than the previous equivalent intelligent surveillance use case. These are among the most important aspects of QoS, and their enhancement thereby suggests the overall significant improvement in QoS of IoT ecosystem.

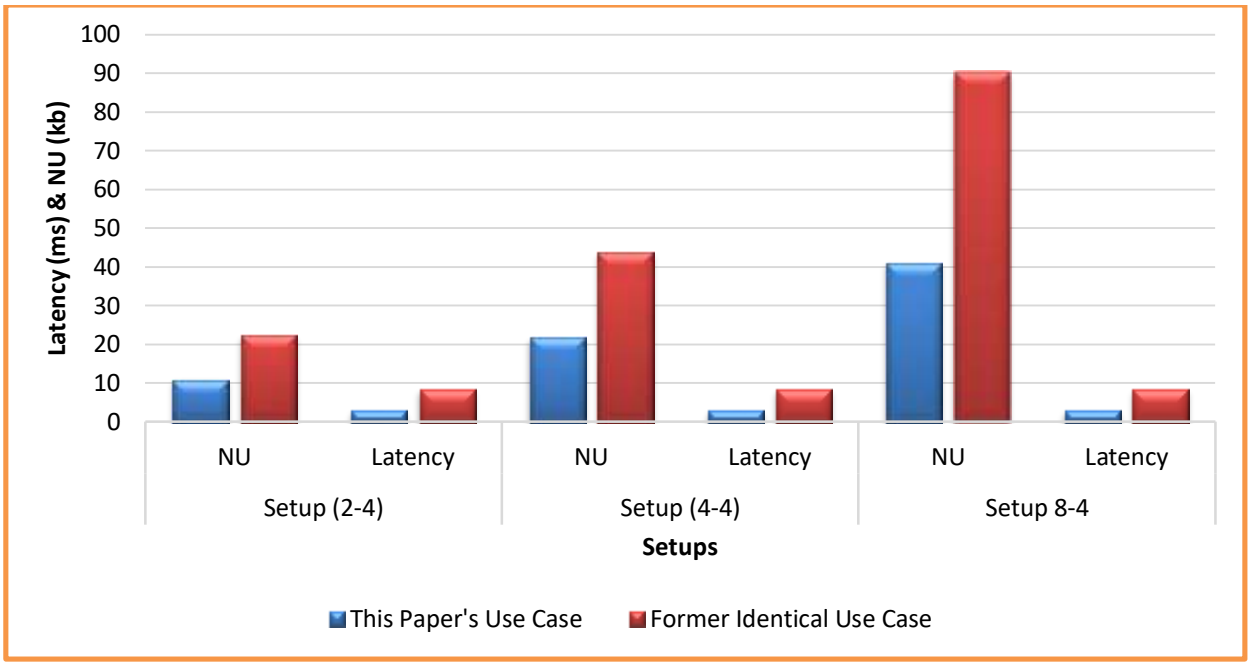


Figure 5.6: Comparison of Latency and NU

5.3 RESULTS AND DISCUSSION OF SD-DRFC FRAMEWORK (METHOD II)

Hierarchical structured network topology has been used for the simulation of presented use case and the different metrics mentioned by iFogSim have been assembled. The simulation findings illustrate how the QoS metrics are influenced by different workloads and placement strategies of inputs. For the sake of checking the efficiency of presented simulated use case of "Method II" for different topology ranges, in this simulation, the amount of Wi-Fi gateways varies from 2 to 8 and the amount of smartphones linked to each gateway varies from 2 to 4. Seven topological configurations have been simulated in this experiment. The configurations together with number of Wi-Fi gateways and the number of smartphones (which are playing the EEG tractor beam game) connected to each gateway are: Setup 1 (2, 2); Setup 2 (2, 4); Setup 3 (4, 2); Setup 4 (4, 4); Setup 5 (6, 2); Setup 6 (6, 4); and Setup 7 (8, 2).

As mentioned earlier, two scenarios are provided by iFogSim, called cloud only and edge-ward placement. The cloud-only implementation approach focuses on the conventional deployment of cloud-based architectures in which cloud servers operate on all modules of an application. The percept-process-activate cycle is implemented in these applications by transmitting data that is interpreted by sensors to the cloud. The data is recorded in the cloud and analyzed when

required. Then actuators are informed when, depending on data processing, action is required. The Edge-ward positioning technique, in contrast, facilitates the deployment of device modules along the network edge. Occasionally, devices close to the edge of the system can not have sufficient processor and storage capacity to satisfy all application providers. The strategy spreads into the upper layer (fog or cloud) in this form of scenario and aims to position the remaining operations on upper servers. By placing modules near to the network edge, this strategy illustrates the interconnections between dew, roof, fog and cloud. Hence in this examined use case, cloud-only and edge-ward (called "Dew+Roof+Fog+Cloud+SDN" in this paper) are two positioning strategies. Four QoS parameters (Latency, Network Use, Cost, and Power Consumption) are evaluated here and then their results are analyzed based on the above mentioned two strategies (i.e. Cloud Only and Dew+Roof+Fog+Cloud+SDN). The results of these QoS parameters are presented in graphical form and analyzed to evaluate the performance of the presented method (Method II).

5.3.1 Latency

The latency of reaction that is responsible for translating the user's brain state into its game state on the smartphone's display is perhaps the most significant control loop in the EEG tractor beam game application. This includes real-time interaction, with fast performance on the classification module, among the smartphone and the machine containing the brain-state classification module. In this loop, delay would seriously harm user experience as it influences objects with which the user communicates directly. Figure 5.7 illustrates the latency of this control loop. The latency here is taken up in the millisecond (ms) unit. The figure demonstrates that the latency of the control loop decreases significantly for the "Dew+Roof+Fog+Cloud+SDN" approach where edge devices are being used for processing. That's because the end user devices are located near to the optimum position for connectivity and processing in "Dew+Roof+Fog+Cloud+SDN", which effectively decreases the latency. Findings suggest that processing in the edge not only decreases response time, but also protects the application from problems of scalability (contributing to a delayed reaction) that occurs due to huge topologies and increased rates of data production.

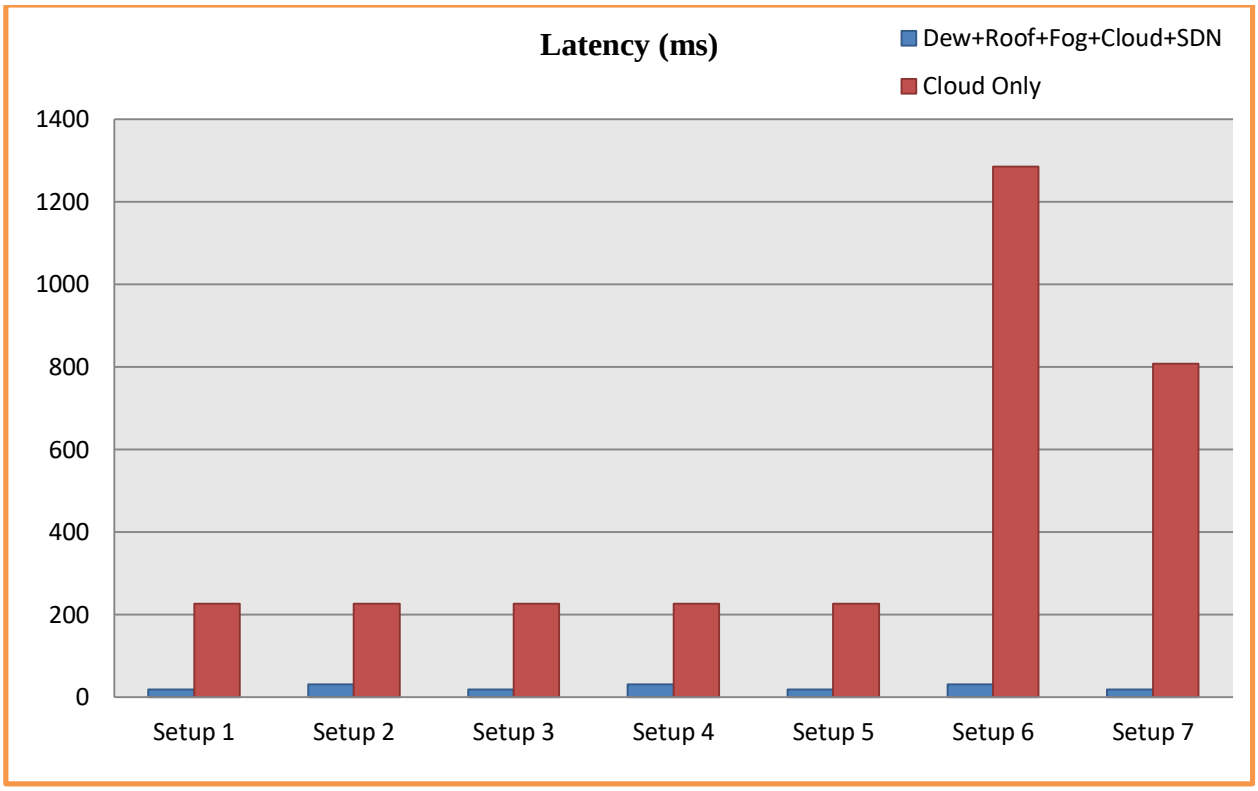


Figure 5.7: Latency for "Dew+Roof+Fog+Cloud+SDN" and “Cloud-only”

5.3.2 Network Usage

Figure 5.8 depicts the network usage of the application for the EEG tractor beam game. Increasing the volume of app-connected devices greatly raises the network load in which only cloud services are used. As seen in Figure 5.8, once edge computing devices are included, the network usage dramatically reduced. The network usage here is taken up in the Bytes unit.

This outcome can also be viewed as a confirmation of edge computing-based device scalability. In the event of cloud-based execution, increased density of network usage can contribute to network congestion and leading to further performance deterioration of the application. If edge computing-based architecture is implemented, such circumstances can be easily handled, and data is pre-processed near to the data source. Notably, a considerable amount of contact between “Consumer” and “Focus Calculator” modules takes place in this program. When these modules are fragmented by long-latency connections with cloud-only placement, this contributes to a high degree of network usage. Nevertheless, the "Dew+Roof+Fog+Cloud+SDN" positions the “Focus Calculator” on the gateways and thus decreases the productive usage of the network.

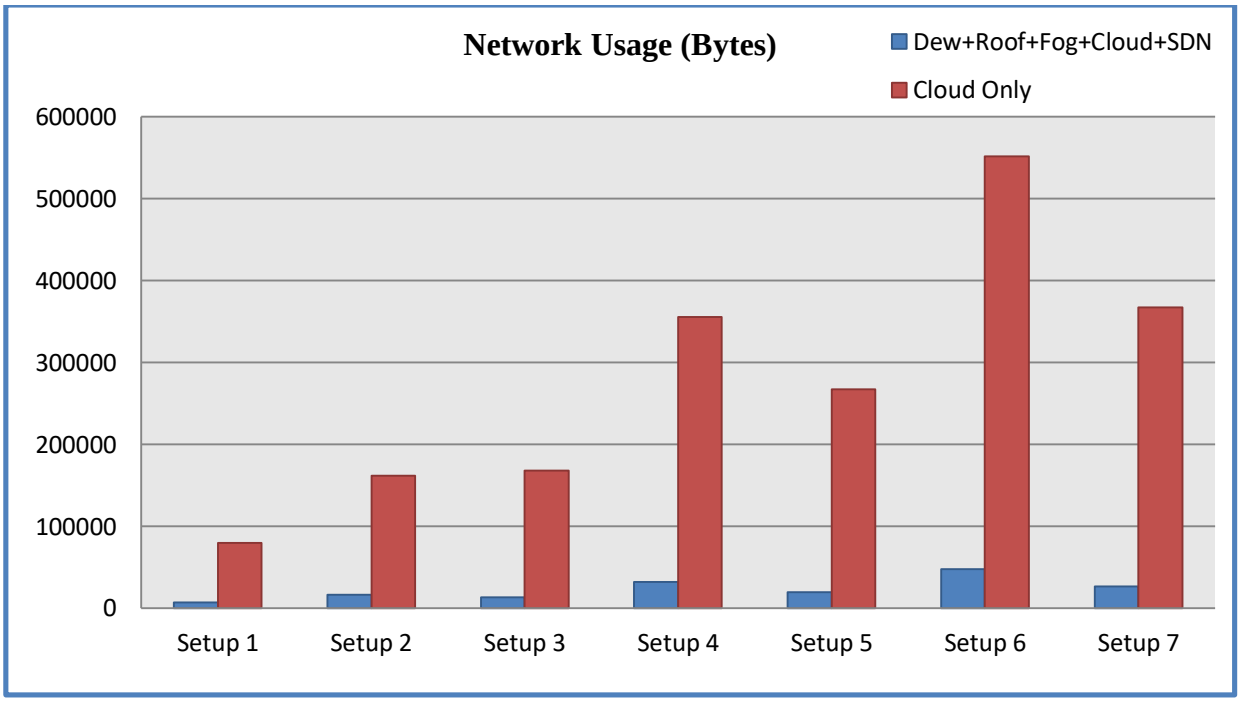


Figure 5.8: Network Usage for "Dew+Roof+Fog+Cloud+SDN" and “Cloud-only”

5.3.3 Cost

Figure 5.9 demonstrates the cloud execution cost (called simply “Cost” in this paper) for cloud-only and "Dew+Roof+Fog+Cloud+SDN" execution. Cost is unit less in iFogSim. iFogSim essentially make comparisons among cloud and edge placement execution. Therefore “cost” is just an indication of which is better place to execute the processing (edge or cloud computing). This indication is done based on the numerical value found from iFogsim. Cloud-only costs are higher, as seen in Figure 5.9, as opposed to "Dew+Roof+Fog+Cloud+SDN". This indicates that the "Dew+Roof+Fog+Cloud+SDN" is better approach to perform the processing of IoT ecosystem data.

5.3.4 Power consumption

For the two cases examined, Figure 5.10 shows the power consumption for the examined simulated use case. The power consumption here is taken up in the Joule unit. As shown in Figure 5.10, using edge computing devices in “Dew+Roof+Fog+Cloud+SDN” strategy reduces power consumption of cloud data centers while slightly increases power consumption of edge devices (specifically Roof devices).

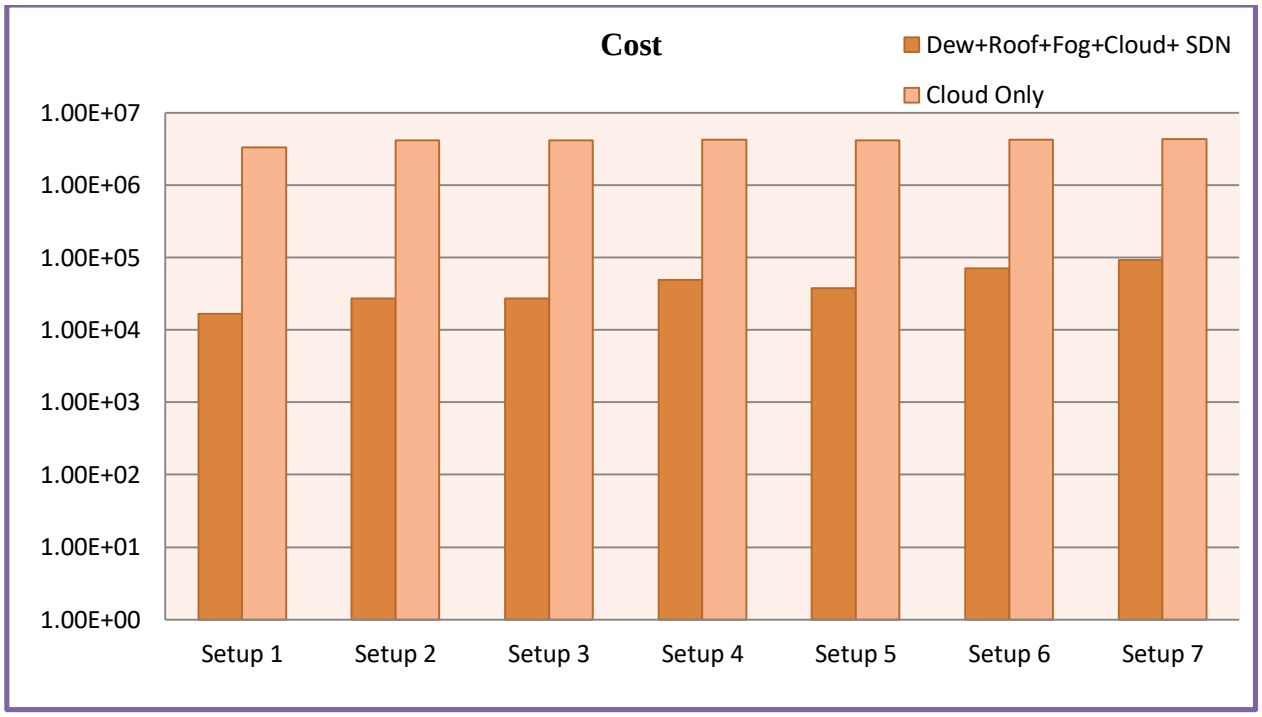


Figure 5.9: Cost for "Dew+Roof+Fog+Cloud+ SDN" and “Cloud-only”

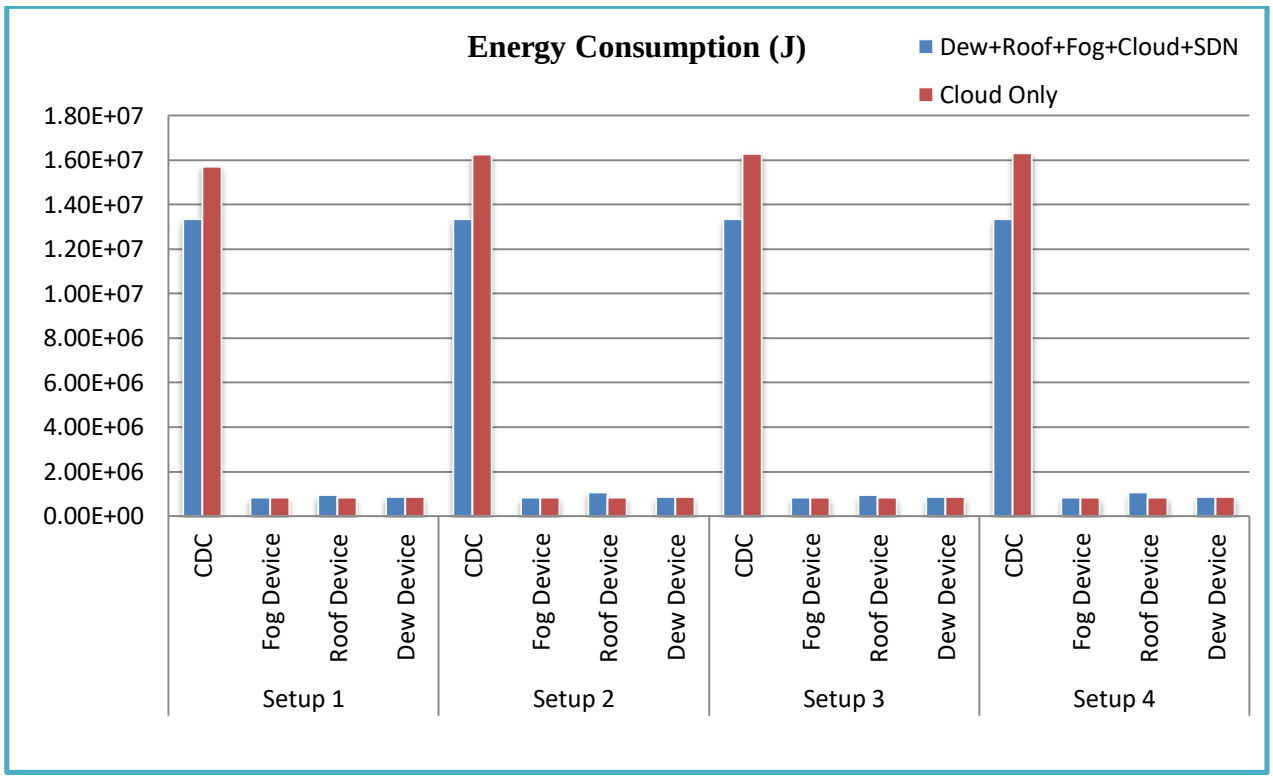


Figure 5.10: Power Consumption for "Dew+Roof+Fog+Cloud+SDN" and “Cloud-only”

In this simulation, the devices from four tires of the proposed simulated use case (but only one device form each tier) is considered for energy consumption. These are: Cloud Data Center (CDC), WiFi Gateway (Fog Device), ISP Gateway (Roof Device), and Smartphone (Dew Device). Power usage for Fog Device and Dew Device is almost the same for cloud-only and "Dew+Roof+Fog+Cloud+SDN" deployment, as the figure shows. But the power consumption for the CDC is pretty much higher in cloud-only deployment when compared to "Dew+Roof+Fog+Cloud+SDN" deployment. Again the power consumption for the Roof Device is slightly higher in "Dew+Roof+Fog+Cloud+SDN" deployment when compared to cloud-only deployment. This is because activities (e.g. data preprocessing, light computing, and delivering data to higher layer) must be conducted in the edge devices in "Dew+Roof+Fog+Cloud+SDN" deployment. This essentially increases the use of power. While the preceding parameters were calculated for 7 configurations, this parameter is calculated for 4 configurations in this simulation. As the rest of the configurations have almost the identical results. Furthermore the following figure is much easier to understand with fewer setup numbers.

When compared the findings of the simulation of above referred four QoS parameters (Latency, Network Use, Cost, and Power Consumption); here the "Dew+Roof+Fog+Cloud+SDN" execution performs much better than cloud-only execution. The findings exhibit pretty much advancement in edge placement execution (i.e. "Dew+Roof+Fog+Cloud+SDN" execution) compared to cloud-only for IoT ecosystem. The above referred parameters are the most significant parameters for IoT system's QoS. Therefore their enhancement indicates a significant advancement in overall QoS of IoT ecosystem. This shows promising results in favor of our proposed method (Method II).

5.4 SUMMARY

To evaluate the performance of the proposed methods, the simulation is performed in this chapter. Using iFogSim simulator, the appropriate setup is made for the simulation and then evaluate to find the performance of the proposed methods. Results show considerable improvement in QoS in favor of presented methods.

CHAPTER 6

CONCLUSION

6.1 INTRODUCTION

This chapter describes the conclusion and future research scope of this thesis work. First the overall summary of this thesis work is presented in the conclusion section. Then the potential future research directions are provided in future research scope section.

6.2 CONCLUSION

Through adding internet access to the objects, the IoT pledges a range of benefits for its users, ranging from quicker and more effective environmental monitoring to more cost-effective manufacturing process monitoring. Since contemporary fragmented ecosystem delays IoT developments; hence an ubiquitous IoT ecosystem will enable developers and integrators to explore through the multitude of innovations, frameworks, solutions and existing infrastructure while expanding smart applications in diverse sectors like agriculture, metro areas, healthcare and business sector. Computing paradigms are such strategies that can be substantially reduced user intervention in IoT ecosystem management. CC was the first such computing paradigm used to administer IoT. But FC comes up because of CCs limitations. RC arrives to perform the assistance needed by FC. But several services which demands internetwork-free environment do not adjust to these existing computing paradigms. Therefore, in order to adapt for this strategy, DC paradigm was adopted that aims to bring cloud, fog, or roof programs and functionalities to the closest periphery of the user without or limited usage of internet. However these paradigms are never to replace each other. They are complementary to each other, and can

be coupled effectively to compensate for each other's weaknesses. Their best use could be achieved when jointly utilizing these multiple computing paradigms. Hence in this paper multiple computing paradigms (cloud, fog, roof, and dew) have incorporated to create a multi-tiered IoT ecosystem infrastructure. But this multi-tiered computational infrastructure involves multidisciplinary technical issues and therefore it would be hard to formulate, organize and build new devices and services from such a strategic approach. Therefore SDN has integrated with that multi-tiered computing infrastructure for mitigating such technical issues. SDN along its network programming abilities, step up as a suitable choice to orchestrate the network, services and devices by covering up the difficulties of this heterogeneous multi-tier computational infrastructure for IoT ecosystem. In this thesis first an elaborate background study is performed among the mentioned multiple computing paradigms. Based on the concept of multi-tiered computing paradigm architecture which is incorporated with SDN; two specific methodss have demonstrated in this paper. The “Method I” combines Cloud and Fog computing and they are incorporated with SDN. A novel architecture and algorithm for this method is presented. The “Method II” presents a novel SD-DRFC framework which combines Cloud, Fog, Roof and Dew computing paradigm which is also incorporated with SDN. This is an ideal framework for today’s IoT ecosystem as it is ubiquitous by nature. Both of these methods are evaluated by the simulator namely iFogSim. For the simulation, IoT use cases which are similar to the proposed approaches architecture are presented and then evaluated. The result shows the proposed approaches offers much better QoS for IoT ecosystem.

6.3 FUTURE RESEARCH SCOPE

Right now there is no simulating environment available that can support mobility while yet providing energy consumption and network usage QoS parameter. Therefore one needs to be compromised when conducting the handover process either of those QoS parameters or the handover operation. A simulation environment incorporating mobility and energy consumption will guide some comprehensive research. The proposed algorithm of “Method I” will also be simulated directly with this kind of environment, as it can't be simulated directly now due to the lack of such environment. Another hearty scope can be to unify radio and network management of resource by developing a multilayer infrastructure that can communicate with SDN and Software Defined Radio (SDR) to increase utilization of spectrum and interplay of channels.

LIST OF REFERENCES

[1] Lee, S. Bae, M., and Kim, H. 2017. Future of IoT networks: A survey. *Applied Sciences*. **7**(10): 1072.

[2]. Miraz, M. H., Ali, M., Excell, P. S., & Picking, R. (2015, September). A review on Internet of Things (IoT), Internet of everything (IoE) and Internet of nano things (IoNT). In *2015 Internet Technologies and Applications (ITA)* (pp. 219-224).

[3]. Data Studio, P., 2020. *The Iot Data Explosion: How Big Is The Iot Data Market?*. [online] Priceonomics. Available at: https://priceonomics.com/the-iot-data-explosion-how-big-is-the-iotdata/

[4]. De Donno, M., Tange, K., & Dragoni, N. (2019). Foundations and evolution of modern computing paradigms: Cloud, iot, edge, and fog. *Ieee Access*, *7*, 150936-150948.

[5]. Framingham, M., 2019. *The Growth In Connected Iot Devices Is Expected To Generate 79.4ZB Of Data In 2025, According To A New IDC Forecast*. [online] IDC: The premier global market intelligence company. Available at: https://www.idc.com/getdoc.jsp?containerId=prUS45213219

[6] Puliafito, C., Vallati, C., Mingozzi, E., Merlino, G., Longo, F., & Puliafito, A. 2019. Container migration in the fog: a performance evaluation. *Sensors*. **19**(7): 1488.

[7] Kamienski, C., Prati, R., Kleinschmidt, J., & Soininen, J. P. July, 2019. Designing an Open IoT Ecosystem. In *Workshop on Cloud Networks, WCN 2019*.

[8] Ecosystem, I., 2020. *Iot Ecosystem | Top 6 Awesome Components Of Iot Ecosystem*. [online] EDUCBA. Available at: <https://www.educba.com/iot-ecosystem/> [Accessed 25 December 2020].

[9] F. Durao, J. Fernando, S. Carvalho, A. Fonseka, and V. C. Garcia. 2014. A systematic review on cloud computing. *J. Supercomput.*. **68**(3): 1321-1346.

[10] Baktyan, A. A., & Zahary, A. T. 2018. A Review on Cloud and Fog Computing Integration for IoT: Platforms Perspective. *EAI Endorsed Transactions on Internet of Things*. **4**(14).

[11] Stojmenovic, I., & Sheng, W. 2014. The fog computing paradigm: Scenarios and security issues. *In Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 1–8.

[12] Luan, T. H., Gao, L., Li, Z., Xiang, Y., Wei, G., & Sun, L. 2015. Fog computing: Focusing on mobile users at the edge. *arXiv preprint arXiv:1502.01815*.

[13] Atlam, H., Walters, R., & Wills, G. 2018. Fog computing and the internet of things: a review, *Big data and cognitive computing*. **2**(2): 10.

[14] Tahir, M., Ashraf, Q. M., & Dabbagh, M. August, 2019. Towards Enabling Autonomic Computing in IoT Ecosystem. In 2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), pp. 646-651.

[15] Zinner, T., Jarschel, M., Blenk, A., Wamser, F., & Kellerer, W. May, 2014. Dynamic application-aware resource management using software-defined networking: Implementation prospects and challenges. In 2014 IEEE Network Operations and Management Symposium (NOMS), pp. 1-6.

[16] Costache, C., Machidon, O., Mladin, A., Sandu, F., & Bocu, R. September, 2014. Software defined networking of linux containers. In 2014 RoEduNet Conference 13th Edition: Networking in Education and Research Joint Event RENAM 8th Conference, pp. 1-4.

[17] Amulothu, Venkata S., Ashish Kapur, and Vishal Shukla. "Facilitating software-defined networking communications in a container-based networked computing environment." U.S. Patent 10,037,220, issued July 31, 2018.

[18] Cziva, R., Jouet, S., White, K. J., & Pezaros, D. P. July, 2015. Container-based network function virtualization for software-defined networks. In 2015 IEEE symposium on computers and communication (ISCC), pp. 415-420.

[19] Bari, M. F., Chowdhury, S. R., Ahmed, R., & Boutaba, R. November, 2013. PolicyCop: An autonomic QoS policy enforcement framework for software defined networks. In 2013 IEEE SDN for Future Networks and Services (SDN4FNS), pp. 1-7.

[20] Hakiri, A., Gokhale, A., Berthou, P., Schmidt, D. C., & Gayraud, T. 2014. Software-defined networking: Challenges and research opportunities for future internet. Computer Networks. **75**: 453-471.

[21] Yan, Z., Zhang, P., & Vasilakos, A. V. 2016. A security and trust framework for virtualized networks and software-defined networking. Security and communication networks. **9**(16): 3059-3069.

[22] Das, T., Caria, M., Jukan, A., & Hoffmann, M. 2013. A techno-economic analysis of network migration to software-defined networking. arXiv preprint arXiv:1310.0216.

[23] Wang, P., Chao, K. M., Lin, H. C., Lin, W. H., & Lo, C. C. November, 2016. An efficient flow control approach for SDN-based network threat detection and migration using support vector machine. In 2016 IEEE 13th International Conference on e-Business Engineering (ICEBE), pp. 56-63.

[24] Mann, V., Vishnoi, A., Kannan, K., & Kalyanaraman, S. April, 2012. CrossRoads: Seamless VM mobility across data centers through software defined networking. In 2012 IEEE Network Operations and Management Symposium, pp. 88-96.

[25] Kaur, S., Kumar, K., Singh, J., & Ghumman, N. S. March, 2015. Round-robin based load balancing in Software Defined Networking. In 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), pp. 2136-2139.

[26] Kim, H., & Feamster, N. 2013. Improving network management with software defined networking. IEEE Communications Magazine. **51**(2): 114-119.

[27] Andrade, L., Borba, M., Ishimori, A., Farias, F., Cerqueira, E., & Abelém, A. October, 2016. On the benchmarking mainstream open software-defined networking controllers. In Proceedings of the 9th Latin America Networking Conference, pp. 9-12.

[28] Yeganeh, S. H., Tootoonchian, A., & Ganjali, Y. 2013. On scalability of software-defined networking. IEEE Communications Magazine. **51**(2): 136-141.

[29] Kim, H., & Feamster, N. 2013. Improving network management with software defined networking. *IEEE Communications Magazine*. **51**(2): 114-119.

[30] Lara, A., Kolasani, A., & Ramamurthy, B. December, 2012. Simplifying network management using software defined networking and OpenFlow. In *2012 IEEE International Conference on Advanced Networks and Telecommunciations Systems (ANTS),* pp. 24-29.

[31] Tsagkaris, K., Logothetis, M., Foteinos, V., Poulios, G., Michaloliakos, M., & Demestichas, P. 2015. Customizable autonomic network management: integrating autonomic network management and software-defined networking. *IEEE Vehicular Technology Magazine*. **10**(1): 61-68.

[32] De Gante, A., Aslan, M., & Matrawy, A. June, 2014. Smart wireless sensor network management based on software-defined networking. In *2014 27th Biennial Symposium on Communications (QBSC),* pp. 71-75.

[33] Moura, H., Bessa, G. V., Vieira, M. A., & Macedo, D. F. May, 2015. Ethanol: Software defined networking for 802.11 wireless networks. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM),* pp. 388-396.

[34] Chourasia, S., & Sivalingam, K. M. April, 2015. SDN based Evolved Packet Core architecture for efficient user mobility support. In *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft),* pp. 1-5.

[35] Bi, Y., Han, G., Lin, C., Deng, Q., Guo, L., & Li, F. 2018. Mobility support for fog computing: An SDN approach. *IEEE Communications Magazine*. **56**(5): 53-59.

[36] Hirsch, M., Mateos, C., Rodriguez, J. M., & Zunino, A. 2019. DewSim: A trace-driven toolkit for simulating mobile device clusters in Dew computing environments. *Software: Practice and Experience*. pp. 1-31.

[37] Singh, M., & Baranwal, G. 2018. Quality of service (qos) in internet of things. In 2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU). pp. 1-6).

[38] Singh, A. K., & Srivastava, S. 2018. A survey and classification of controller placement problem in SDN. *International Journal of Network Management*. **28**(3).

[39] Babu, R., Jayashree, K., & Abirami, R. 2018. FC QoS Review and Open Challenges. International Journal of FC (IJFC). **1**(2): 109-118.

[40] Anawar, M. R., Wang, S., Azam Zia, M., Jadoon, A. K., Akram, U., & Raza, S. 2018. FC: An overview of big IoT data analytics. *Wireless Communications and Mobile Computing*.

[41] Kumari, A., Tanwar, S., Tyagi, S., Kumar, N., Parizi, R. M., & Choo, K. K. R. 2019. Fog data analytics: a taxonomy and process model. *Journal of Network and Computer Applications*. **128**: 90-104.

[42] Naas, M. I., Parvedy, P. R., Boukhobza, J., & Lemarchand, L. 2017. iFogStor: an IoT data placement strategy for fog infrastructure. *In 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*. pp. 97-104.

[43] Mahmud, R., Ramamohanarao, K., & Buyya, R. 2018. Latency-aware application module management for FC environments. *ACM Transactions on Internet Technology (TOIT)*. **19**(1): 9.

[44] Yousefpour, A., Ishigaki, G., & Jue, J. P. 2017. FC: Towards minimizing delay in the internet of things. *In 2017 IEEE international conference on edge computing (EDGE)*, pp. 17-24.

[45] Yousefpour, A., Patil, A., Ishigaki, G., Kim, I., Wang, X., Cankaya, H. C., & Jue, J. P. (2019). FogPlan: a lightweight QoS-aware dynamic fog service provisioning framework. *IEEE Internet of Things Journal*, **6**(3), 5080-5096.

[46] Sarddar, D., Barman, S., Sen, P., & Pandit, R. 2018. Refinement of Resource Management in FC Aspect of QoS. *International Journal of Grid and Distributed Computing*. **11**(5): 29-44.

[47] Baktyan, A., & Zahary, A. 2018. A review on cloud and FC integration for iot: Platforms perspective. *EAI Endorsed Transactions on Internet of Things*. **4**(14).

[48] Yassein, M. B., Aljawarneh, S., Al-Rousan, M., Mardini, W., & Al-Rashdan, W. November, 2017. Combined software-defined network (SDN) and Internet of Things (IoT). In *2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*. pp. 1-6.

[49] Bera, S., Misra, S., & Vasilakos, A. V. 2017. Software-defined networking for internet of things: A survey. *IEEE Internet of Things Journal*. **4**(6): 1994-2008.

[50] Tayyaba, S. K., Shah, M. A., Khan, O. A., & Ahmed, A. W. July, 2017. Software Defined Network (SDN) Based Internet of Things (IoT) A Road Ahead. In *Proceedings of the International Conference on Future Networks and Distributed Systems*, pp. 1-8.

[51] Chemeritskiy, E., & Smelansky, R. 2014. On QoS management in SDN by multipath routing. *International Science and Technology Conference (Modern Networking Technologies)(MoNeTeC)*, pp. 1-6.

[52] Wang, J., & Li, D. 2018. Adaptive Computing Optimization in Software-Defined Network-Based Industrial Internet of Things with FC. *Sensors*. **18**(8): 2509.

[53] Hu, T., Guo, Z., Yi, P., Baker, T., & Lan, J. (2018). Multi-controller based software-defined networking: A survey. *IEEE Access*, *6*, 15980-15996.

[54] Mouawad, N., Naja, R. & Tohme, S. 2018. Optimal and dynamic SDN controller placement. In *2018 International Conference on Computer and Applications (ICCA.)* IEEE. pp. 1-9.

[55] H. K. Rath, V. Revoori, S. M. Nadaf & A. Simha. 2014. Optimal controller placement in Software Defined Networks (SDN) using a non-zero-sum game. *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, Sydney, NSW*. pp. 1-6.

[56] Ali, J., Roh, B. H., & Lee, S. 2019. QoS improvement with an optimum controller selection for software-defined networks. *PloS one*. **14**(5): e0217631.

[57] Zhao, Z., & Wu, B. 2017. Scalable SDN architecture with distributed placement of controllers for WAN. *Concurrency and Computation: Practice and Experience*. **29**(16): e4030.

[58] Akintoye, S. B., & Bagula, A. 2019. Improving Quality-of-Service in Cloud/FC through Efficient Resource Allocation. *Sensors*. **19**(6): 1267.

[59] Son, J., & Buyya, R. 2018. A taxonomy of software-defined networking (SDN)-enabled cloud computing. *ACM Computing Surveys (CSUR)*. **51**(3): 1-36.

[60] Azodolmolky, S., Wieder, P., & Yahyapour, R. June, 2013. SDN-based CC networking. In *2013 15th International Conference on Transparent Optical Networks (ICTON)*, pp. 1-4.

[61] Jararweh, Y., Al-Ayyoub, M., Benkhelifa, E., Vouk, M., & Rindos, A. 2016. Software defined cloud: Survey, system and evaluation. *Future Generation Computer Systems*. **58**: 56-74.

[62] Yen, T. C., & Su, C. S. (2014, April). An SDN-based cloud computing architecture and its mathematical model. In *2014 International Conference on Information Science, Electronics and Electrical Engineering* (Vol. 3, pp. 1728-1731).

[63] Salman, O., Elhajj, I., Chehab, A., & Kayssi, A. (2018). IoT survey: An SDN and fog computing perspective. *Computer Networks*, **143**, 221-246.

[64] Gupta, H., Nath, S. B., Chakraborty, S., & Ghosh, S. K. 2016. Sdfog: A software defined computing architecture for qos aware service orchestration over edge devices. *arXiv preprint arXiv:1609.01190*.

[65] Khakimov, A., Ateya, A. A., Muthanna, A., Gudkova, I., Markova, E., & Koucheryavy, A. June, 2018. IoT-fog based system structure with SDN enabled. In *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems.* pp. 1-6.

[66] Tomovic, S., Yoshigoe, K., Maljevic, I., & Radusinovic, I. 2017. Software-defined fog network architecture for IoT. *Wireless Personal Communications*. **92**(1): 181-196.

[67] Hakiri, A., Sellami, B., Patil, P., Berthou, P., & Gokhale, A. October, 2017. Managing wireless fog networks using software-defined networking. In *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA).* pp. 1149-1156.

[68] Huang, L., Li, G., Wu, J., Li, L., Li, J., & Morello, R. October, 2016. Software-defined QoS provisioning for FC advanced wireless sensor networks. In *2016 Ieee Sensors,* pp. 1-3.

[69] Bi, Y., Han, G., Lin, C., Deng, Q., Guo, L., & Li, F. 2018. Mobility support for fog computing: An SDN approach. *IEEE Communications Magazine*. **56**(5): 53-59.

[70] Mazhelis, O., Luoma, E., & Warma, H. 2012. Defining an internet-of-things ecosystem. In *Internet of Things, Smart Spaces, and Next Generation Networking,* pp. 1-14.

[71] Bansal, S., & Kumar, D. 2020. IoT Ecosystem: A Survey on Devices, Gateways, Operating Systems, Middleware and Communication. *International Journal of Wireless Information Networks*. pp. 1-25.

[72] Sethi, P., & Sarangi, S. R. 2017. Internet of things: architectures, protocols, and applications. *Journal of Electrical and Computer Engineering*.

[73] Gazis, V., Görtz, M., Huber, M., Leonardi, A., Mathioudakis, K., Wiesmaier, A. & Vasilomanolakis, E. August, 2015. A survey of technologies for the internet of things. In *2015 International Wireless Communications and Mobile Computing Conference (IWCMC),* pp. 1090-1095.

[74] Gil, D., Ferrández, A., Mora-Mora, H., & Peral, J. 2016. Internet of things: A review of surveys based on context aware intelligent services. *Sensors.* **16**(7): 1069.

[75] Zaidan, A. A., Zaidan, B. B., Qahtan, M. Y., Albahri, O. S., Albahri, A. S., Alaa, M. & Lim, C. K. 2018. A survey on communication components for IoT-based technologies in smart homes. *Telecommunication Systems*. **69**(1): 1-25.

[76] Rehman, A. U., Syed, A. R., Khan, I. U., Mustafa, A. A., Anwer, M. B., & Ali, U. A. 2020. IoT-Enabled Smart Socket. *Wireless Personal Communications*. pp. 1-19.

[77] Noura, M., Atiquzzaman, M., & Gaedke, M. 2019. Interoperability in internet of things: Taxonomies and open challenges. *Mobile Networks and Applications*. **24**(3): 796-809.

[78] Imtiaz, S., Sadre, R., & Vlassov, V. July, 2019. On the Case of Privacy in the IoT Ecosystem: A Survey. In *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData),* pp. 1015-1024.

[79] Hassan, W. H. 2019. Current research on Internet of Things (IoT) security: A survey. *Computer Networks*. **148**: 283-294.

[80] Ali, F., Khan, M. S., & Akhtar, H. 2019. Security Review in Internet of Things. *Internet of Things and Cloud Computing*. **7**(3): 80.

[81] Granjal, J., Monteiro, E., & Silva, J. S. 2015. Security for the internet of things: a survey of existing protocols and open research issues. *IEEE Communications Surveys & Tutorials*. **17**(3): 1294-1312.

[82] Botta, A., De Donato, W., Persico, V., & Pescapé, A. 2016. Integration of CCand internet of things: a survey. *Future generation computer systems*. **56**: 684-700.

[83] Chang, K. D., Chen, C. Y., Chen, J. L., & Chao, H. C. September, 2011. Internet of things and CCfor future internet. In *International Conference on Security-Enriched Urban Computing and Smart Grid,* pp. 1-10.

[84] Odun-Ayo, I., Okereke, C., & Orovwode, H. E. 2018. CC and Internet of Things: Issues and Developments. 182-187

[85] Atlam, H. F., Alenezi, A., Alharthi, A., Walters, R. J., & Wills, G. B. June, 2017. Integration of CCwith internet of things: challenges and open issues. In *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData),* pp. 670-675.

[86] Belgaum, M. R., Soomro, S., Alansari, Z., Musa, S., Alam, M., & Su'ud, M. M. 2017. Challenges: Bridge between cloud and IoT. In *2017 4th IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS),* pp. 1-5.

[87] Rizwan, M. (2018). Cloud Computing Serving as a Solution to the IoT Generated Data. *Bahria University Journal of Information & Communication Technologies (BUJICT)*, *11*(2), 1-6.

[88] Babu, S. M., Lakshmi, A. J., & Rao, B. T. (2015, April). A study on cloud based Internet of Things: CloudIoT. In *2015 global conference on communication technologies (GCCT)* (pp. 60 65). IEEE.

[89] Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. August, 2012. FCand its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile CC*, pp. 13-16.

[90] Baktyan, A., & Zahary, A. 2018. A review on cloud and fog computing integration for iot: Platforms perspective. *EAI Endorsed Transactions on Internet of Things*. **4**(14).

[91] EL IDRISSI, M., ELBEQQALI, O., & RIFFI, J. October, 2019. From CCto fog computing: two technologies to serve iot–a review. In *2019 IEEE International Smart Cities Conference (ISC2)*, pp. 272-279.

[92] El Idrissi, M., Elbeqqali, O., & RIFfi, J. October, 2019. A Review On Relationship Between IoT–Cloud Computing–FC (Applications And Challenges). In *2019 Third International Conference on Intelligent Computing in Data Sciences (ICDS)*, pp. 1-7.

[93] Kunal, S., Saha, A., & Amin, R. 2019. An overview of cloud-fog computing: Architectures, applications with security challenges. *Security and Privacy*. **2**(4): e72.

[94] Hong, H. J., Tsai, P. H., Cheng, A. C., Uddin, M. Y. S., Venkatasubramanian, N., & Hsu, C. H. December, 2017. Supporting internet-of-things analytics in a FC platform. In *2017 IEEE International Conference on CC Technology and Science (CloudCom)*, pp. 138-145.

[95] Yousefpour, A., Ishigaki, G., & Jue, J. P. 2017. Fog computing: Towards minimizing delay in the internet of things. *In 2017 IEEE international conference on edge computing (EDGE)*, pp. 17-24.

[96] Abuseta, Y. 2019. A FC Based Architecture for IoT Services and Applications Development. *arXiv preprint arXiv:1911.02403*.

[97] Cha, H. J., Yang, H. K., & Song, Y. J. 2018. A study on the design of FC architecture using sensor networks. *Sensors*. **18**(11): 3633.

[98] Aazam, M., Zeadally, S., & Harras, K. A. 2018. FC architecture, evaluation, and future research directions. *IEEE Communications Magazine*. **56**(5): 46-52.

[99] Jin, Q., Lin, R., Zou, H., & Yang, F. June, 2018. A distributed FCarchitecture supporting multiple migrating mode. In *2018 5th IEEE International Conference on Cyber Security and CC(CSCloud)/2018 4th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, pp. 218-223.

[100] Baccarelli, E., Naranjo, P. G. V., Scarpiniti, M., Shojafar, M., & Abawajy, J. H. 2017. Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study. *IEEE access*. **5**: 9882-9910.

[101] Shahzad, M., Panneerselvam, J., Liu, L., & Zhai, X. August, 2019. Data Aggregation Challenges in Fog Computing. In *2019 IEEE SmartWorld, Ubiquitous Intelligence &*

Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), pp. 1717-1721.

[102] Wang, Y., Uehara, T., & Sasaki, R. July, 2015. Fog computing: Issues and challenges in security and forensics. In *2015 IEEE 39th Annual Computer Software and Applications Conference*. **3**: 53-59.

[103] Yakubu, J., Christopher, H. A., Chiroma, H., & Abdullahi, M. 2019. Security challenges in fog-computing environment: a systematic appraisal of current developments. *Journal of Reliable Intelligent Environments*. pp. 1-25.

[104] Madanpalli, S., 2019. *The Roof Computing | Electronics For You*. [online] Electronics For You. Available at: <https://www.electronicsforu.com/technology-trends/must-read/the-roof-computing>

[105] Anatol, B., 2019. *Iiot – Intelligent Iot*. [online] Researchgate. Available at: <https://www.researchgate.net/publication/331473327_IIoT_-_Intelligent_IoT>

[106] Santulli, J., 2016. *P1931.1 - Standard For An Architectural Framework For Real-Time Onsite Operations Facilitation (ROOF) For The Internet Of Things*. [online] Standards.ieee.org. Available at: <https://standards.ieee.org/project/1931_1.html>

[107] Meloni, A., Madanapalli, S., Divakaran, S. K., Browdy, S. F., Paranthaman, A., Jasti, A. & Kumar, D. 2018. Exploiting the IoT potential of blockchain in the IEEE P1931. 1 ROOF standard. *IEEE Communications Standards Magazine*. **2**(3): 38-44.

[108] Madanpalli, S., 2017. *The Impact Of Iot On Cloud Computing, Big Data & Analytics*. [online] Slideshare. Available at: <https://www.slideshare.net/smadanapalli/the-impact-of-iot-on-cloud-computing-bigdata-analytics>

[109] Wang, Y., Skala, K., Rindos, A., Gusev, M., Yang, S., & PAN, Y. 2017. DCand transition of internet computing paradigms. *ZTE COMMUNICATIONS*. **15**(4).

[110] Kukreja, P., & Sharma, D. 2016. A detail review on cloud, fog and dew computing. *International Journal of Science, Engineering and Technology Research (IJSETR)*. **5**(5): 1412.

[111] Cristescu, G., Dobrescu, R., Chenaru, O., & Florea, G. May, 2019. DEW: A New Edge Computing Component for Distributed Dynamic Networks. In *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*, pp. 547-551.

[112] Rindos, A., & Wang, Y. October, 2016. Dew computing: The complementary piece of cloud computing. In *2016 IEEE International Conferences on Big Data and CC(BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom)*, pp. 15-20.

[113] Patel, H. M., Chaudhari, R. R., Prajapati, K. R., & Patel, A. A. 2018. The Interdependent Part of Cloud Computing: Dew Computing. In *Intelligent Communication and Computational Technologies*, pp. 345-355.

[114] Wang, Y. (2015). Cloud-dew architecture. *International Journal of Cloud Computing*. **4**(3):199-210.

[115] Fisher, D. E., & Yang, S. 2016. Doing more with the dew: a new approach to cloud-dew architecture. *Open Journal of CC (OJCC)*. **3**(1): 8-19.

[116] Tefera, G., She, K., & Deeba, F. February, 2019. Decentralized Adaptive Latency-Aware Cloud-Edge-Dew Architecture for Unreliable Network. In *Proceedings of the 2019 11th International Conference on Machine Learning and Computing*, pp. 142-146.

[117] Gushev, M. 2020. DC Architecture for Cyber-Physical Systems and IoT. *Internet of Things*, 100186.

[118] Šojat, Z., & Skala, K. May, 2016. Views on the role and importance of DCin the service and control technology. In *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 164-168.

[119] Wang, Y., & Leblanc, D. October, 2016. Integrating SaaS and SaaP with dew computing. In *2016 IEEE International Conferences on Big Data and CC(BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom)*, pp. 590-594.

[120] Longo, M., Hirsch, M., Mateos, C., & Zunino, A. 2019. Towards integrating mobile devices into dew computing: A model for hour-wise prediction of energy availability. *Information*. **10**(3): 86.

[121] Garrocho, C. T. B., & Oliveira, R. A. R. 2019. Counting time in drops: views on the role and importance of smartwatches in dew computing. *Wireless Networks*. pp. 1-19.

[122] Ray, P. P. 2017. An introduction to dew computing: definition, concept and implications. *IEEE Access*. **6**: 723-737.

[123] Ray, P. P. 2019. Minimizing dependency on internetwork: Is DCa solution? *Transactions on Emerging Telecommunications Technologies*. **30**(1): e3496.

[124] Li, Y., Su, X., Riekki, J., Kanter, T., & Rahmani, R. May, 2016. A SDN-based architecture for horizontal Internet of Things services. In *2016 IEEE International Conference on Communications (ICC)*, pp. 1-7.

[125] Wang, J., & Li, D. 2018. Adaptive computing optimization in software-defined network based industrial Internet of Things with fog computing. *Sensors*. **18**(8): 2509.

[126] Truong, N. B., Lee, G. M., & Ghamri-Doudane, Y. May, 2015. Software defined networking-based vehicular adhoc network with fog computing. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 1202-1207.

[127] Nobre, J. C., de Souza, A. M., Rosário, D., Both, C., Villas, L. A., Cerqueira, E. & Gerla, M. 2019. Vehicular software-defined networking and fog computing: Integration and design principles. *Ad Hoc Networks*. **82**: 172-181.

[128] Gupta, A., 2018. *4 Layers Of The Internet Of Things | Jigsaw Academy*. [online] Jigsaw Academy. Available at: <https://analyticstraining.com/4-layers-of-the-internet-of-things/>

[129] L. Christina1 , T. Nithyakamala. 2018. An Analysis on the Challenges of IoT. *International Journal of Innovative Research in Science, Engineering and Technology*. **7**(8): 118-121.

[130] Arora, H. and Thakur, T., 2019. *Software Defined Networking (SDN)- Architecture And Role Of Openflow.* [online] Howtoforge.com. Available at: <https://www.howtoforge.com/tutorial/software-defined-networking-sdn-architecture-and-role-of-openflow/>

[131] Lessing, M., 2019. *What Is An SDN Controller? Definition - Sdxcentral*. [online] SDxCentral. Available at: https://www.sdxcentral.com/networking/sdn/definitions/what-is-sdn-controller/

[132] Abdullah-Al-Shafi, M., & Bahar, A. N. 2016. Cloud Computing: An Aspect of Information System. *International Journal of Applied Information Systems*. **10** (4): 46-50.

[133] Caria, M., Jukan, A., & Hoffmann, M. (2016). SDN partitioning: A centralized control plane for distributed routing protocols. *IEEE Transactions on Network and Service Management*, *13*(3), 381-393.

[134] Kim, M. S., Kim, Y., Lee, S., Lee, S. & Golmie, N. 2019. A user application-based access point selection algorithm for dense WLANs. *PloS one*, **14**(1).

[135] Raschellà, A., Bouhafs, F., Mackay, M., Shi, Q., Ortín, J., Gállego, J. R. & Canales, M. 2020. A dynamic access point allocation algorithm for dense wireless LANs using potential game. *Computer Networks*. **167**: 106991.

[136] Wu, D., Arkhipov, D. I., Asmare, E., Qin, Z. & McCann, J. A. 2015. UbiFlow: Mobility management in urban-scale software defined IoT. In *2015 IEEE conference on computer communications (INFOCOM)*. IEEE. pp. 208-216.

[137] Hong, K., Lillethun, D., Ramachandran, U., Ottenwalder, B., & Koldehofe, B. 2013. Mobile fog: A programming model for large-scale applications on the internet of things. *In ACM SIGCOMM workshop on Mobile cloud computing*, pp. 15–20.

[138] OneMind Technologies. 2020. *Understanding The Iot Ecosystem: Platforms, Applications, And Solutions*. [online] Available at: <https://onemindtechnologies.com/understanding-the-iot-ecosystem-platforms-applications-and-solutions/>

[139] Gusev, M. May, 2017. A dew computing solution for IoT streaming devices. In *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 387-392.

[140] Adelantado, F., Vilajosana, X., Tuset-Peiro, P., Martinez, B., Melia-Segui, J., & Watteyne, T. 2017. Understanding the limits of LoRaWAN. *IEEE Communications magazine*. **55**(9): 34-40.

[141] Glossary, G., 2018. *Definition Of Mobile Cloud Synchronization - Gartner Information Technology Glossary*. [online] Gartner. Available at: <https://www.gartner.com/en/information-technology/glossary/mobile-cloud-synchronization>

[142] McCormick, Z., & Schmidt, D. C. 2012. Data synchronization patterns in mobile application design. *Vanderbilt University*. pp. 1-14.

[143] Oracle Help Center. 2020. *Using Oracle Mobile Cloud Service*. [online] Available at: <https://docs.oracle.com/en/cloud/paas/mobile-cloud/mcsua/data-offline-and-sync.html#GUID-52F00ED6-BA1C-43FF-A87F-5BD77DF40B53>

[144] Identity, O., 2020. *Context Aware Security, A New Adaptive Security Model*. [online] One Identity. Available at: <https://www.oneidentity.com/context-aware-security/>

[145] Baktir, A. C., Ozgovde, A., & Ersoy, C. 2017. How can edge computing benefit from software-defined networking: A survey, use cases, and future directions. *IEEE Communications Surveys & Tutorials*. **19**(4): 2359-2391.

[146] D. Amendola, N. Cordeschi, and E. Baccarelli. 2016. Bandwidth management vms live migration in wireless FC for 5g networks. In 5th IEEE International Conference on Cloud Networking (Cloudnet), pp. 21–26.

[147] Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K., & Buyya, R. 2017. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Software: Practice and Experience*. **47**(9): 1275-1296.

LIST OF PUBLICATIONS

[1] Ishtiaq Ahammad, Md. Ashikur Rahman Khan, Zayed-Us-Salehin, Main Uddin, Sultana Jahan Soheli. 2021. Improvement of QoS in an IoT Ecosystem by Integrating Fog Computing and SDN. International Journal of Cloud Applications and Computing (IJCAC). Volume 11, Issue 2, Article 4. IGI Global. (Accepted)

[2] Ishtiaq Ahammad, Md. Ashikur Rahman Khan, Zayed-Us-Salehin. 2021. A Review on Cloud, Fog, Roof and Dew Computing: IoT Perspective. International Journal of Cloud Applications and Computing (IJCAC). Volume 11, Issue 4, Article 2. IGI Global. (Accepted)

[3] Ishtiaq Ahammad, Md. Ashikur Rahman Khan, Zayed-Us-Salehin. 2021. Advancement of IoT System's QoS by Integrating Cloud, Fog, Roof, and Dew Computing Assisted by SDN: Basic Framework Architecture and Simulation. International Journal of Ambient Computing and Intelligence (IJACI). IGI Global. (Accepted)