SOFTWARE DEFINED NETWORKING BASED RESOURCE MANAGEMENT AND QUALITY OF SERVICE SUPPORT IN WIRELESS SENSOR NETWORK APPLICATIONS

by

Babedi Betty Letswamotse

Submitted in fulfilment of the requirements for the degree
Philosophiae Doctor (Electronics)

in the

Department of Electrical, Electronic and Computer Engineering
Faculty of Engineering, Built Environment and Information Technology

UNIVERSITY OF PRETORIA

September 2018

SUMMARY

SOFTWARE DEFINED NETWORKING BASED RESOURCE MANAGEMENT AND QUALITY OF SERVICE SUPPORT IN WIRELESS SENSOR NETWORK APPLICATIONS.

by

Babedi Betty Letswamotse

Supervisor: Prof R. Malekian

Department: Electrical, Electronic and Computer Engineering

University: University of Pretoria

Degree: PhD of Engineering (Electronics)

Keywords: Wireless Sensor Network, software defined networking, quality of service, software defined wireless sensor network, resource management, resource management, machine learning

To achieve greater performance in computing networks, a setup of critical computing aspects that ensures efficient network operation, needs to be implemented. One of these computing aspects is, Quality of Service (QoS). Its main functionality is to manage traffic queues by means of prioritizing sensitive network traffic. QoS capable networking allows efficient control of traffic especially for network critical data. However, to achieve this in Wireless Sensor Networks (WSN) is a serious challenge, since these technologies have a lot of computing limitations. It is even difficult to manage networking resources with ease in these types of technologies, due to their communication, processing and memory limitations. Even though this is the case with WSNs, they have been largely used in monitoring/detection systems, and by this proving their application importance.

Realizing efficient network control requires intelligent methods of network management, especially for sensitive network data. Different network types implement diverse methods

to control and administer network traffic as well as effectively manage network resources. As with WSNs, communication traffic and network resource control are mostly performed depending on independently employed mechanisms to deal with networking events occurring on different levels. It is therefore challenging to realize efficient network performance with guaranteed QoS in WSNs, given their computing limitations. Software defined networking (SDN) is advocated as a potential paradigm to improve and evolve WSNs in terms of capacity and application. A means to apply SDN strategies to these compute-limited WSNs, formulates software defined wireless sensor networks (SDWSN).

In this work, a resource-aware OpenFlow-based Active Network Management (OF-ANM) QoS scheme that uses SDN strategies is proposed and implemented to apply QoS requirements for managing traffic congestion in WSNs. This scheme uses SDN programmability strategies to apply network QoS requirements and perform traffic load balancing to ensure congestion control in SDWSN. Our experimental results show that the developed scheme is able to provide congestion avoidance within the network. It also allows opportunities to implement flexible QoS requirements based on the system's traffic state.

Moreover, a QoS Path Selection and Resource-associating (Q-PSR) scheme for adaptive load balancing and intelligent resource control for optimal network performance is proposed and implemented. Our experimental results indicate better performance in terms of computation with load balancing and efficient resource alignment for different networking tasks when compared with other competing schemes.

DEDICATION

I dedicate this work to my siblings (Simon, Rebecca, Matshidiso and Seapei), my aunt Mrs. Molebogeng Yvonne Mogopa, Ms. Basadi Merriam Magano and my daughter Gomolemo Abigail Letswamotse.

A special dedication to my late parents Mr. Pogisho Labious Letswamotse and Mrs. Bailakgabo Magdeline Letswamotse.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere appreciation to my supervisor Prof. Reza Malekian. Your patience and support towards me throughout the progression of this work has been overwhelming. Thank you, Prof., for your kindness, continuous motivation and your technical advice. I will also like to thank Dr. Uche Chude-Okonkwo for mentoring me during the implementation stages of this research. Your wisdom and support are highly appreciated.

I would also like express my gratitude to the University of Pretoria and the Department of Electrical, Electronic and Computer Engineering for providing technical equipment and research support that assisted in achieving this work. I would like to accordingly, acknowledge the National Research Foundation (NRF) of South Africa (grant numbers: IFR160118156967 and RDYR160404161474) for the financial support awarded to me, to see me through this work.

I would also like to express my heartfelt gratitude to Dr. Kgotlaetsile Mathews Modieginyane (my life partner and research partner), thank you for being supportive, for always finding a way to motivate me when I was slacking and for offering assistance when needed. A special thanks to my sister and friend Mmmathebe Esther Lesetedi for your unwavering support. I also extend my gratitude to my siblings (Simon Letswamotse, Rebecca Letswamotse, Matshidiso Mmitloe and Seapei Letswamotse) for the sacrifices you make for me and for your continuous support and motivation.

Last but not least, I would like to thank God for always making everything work together for my good, for providing me with a strong support system and giving me the endurance throughout the advancement of this work.

LIST OF ABBREVIATIONS

6LowPAN	Low-power wireless personal area networks
ACLs	Access Control Lists
API	Applications Programming Interfaces
AODV	Ad hoc On-demand Distance Vector
CABPA	Centralized Adaptive Bandwidth and Power Allocation
CAPA	Centralized Adaptive Bandwidth Allocation
CAR	Committed Access Rate
CBWFQ	Class-Based Weighted Fair Queuing
CCRP	Cloud Computing Resource Pool
CH	Cluster Head
CPU	Central Processing Unit
CQ	Custom Queuing
C-RAN	Cloud-RAN
CSDWSN	Cloud-Assisted SDWSN
DABPA	Distributed Adaptive Bandwidth and Power Allocation
DAlPaS	Dynamic Alternative Path Selection
DASN	Deputy Assistant Secretary of the Navy
DICM	Delay-Inducing Congestion Mitigation
ECCKN	Energy-Consumption-based Connected K-Neighbourhood
FPGA	Field-Programmable Gate Array
FSO	Flow Splitting Optimization

HTTP	Hypertext Transfer Protocol
HWSN	Heterogeneous WSNs
IoT	Internet of Things
ILP	Integer Linear Programming
IWSNs	Industrial WSNs
LEACH	Low-Energy Adaptive Clustering Hierarchy
LLNs	Low power and Lossy Networks
LQRP	Link-Quality Routing Protocol
MCU	Microcontroller Unit
MOS	Mean Opinion Score
MSEECH	Modified SEECH
NBAR	Network-Based Application Recognition
NETCONF	Network Configuration Protocol
NFV	Network Function Virtualization
NS-3	Network Simulator 3
NP	Non-deterministic Polynomial-time
OF-ANM	OpenFlow-based Active Network Management
ONOS	Open Network Operating System
OS	Operating System
OSPF	Open Shortest Path First
OVS	Open Virtual Switch
PDR	Packet Delivery Ratio
PNSR	Peak Signal-to-Noise Ratio
PQ	Priority Queuing
QoS	Quality of Service
Q-PSR	QoS Path Selection and Resource-associating
RAN	Radio Access Network

REST	Representational State Transfer
RF	Radio Frequency
RIP	Routing Information Protocol
RL	Reinforced Learning
RPL	Routing Protocol for LLNs
RTP	Real-time Transport Protocol
SAPS	Situation Aware Protocol Switching Scheme
SDP	Semi-Definite Programming
SDN	Software Defined Networking
SDSN	Software Defined Sensor Network
SDWSN	Software Defined Wireless Sensor Network
SEECH	Scalable Energy Efficient Clustering Hierarchy
SNMP	Simple Network Management Protocol
SSIM	Structural Similarity
TARA	Topology Aware Resource Adaptation
TLM	Traffic Load Minimization
TTL	Time To Live
UDP	User Datagram Protocol
WBSN	Wireless Body Sensor Network
WFQ	Weighted Fair Queuing
WMSN	Wireless Multimedia Sensor Network
WRED	Weighted Random Early Detect
WSAN	Wireless Sensor and Actuator Network
WSN	Wireless Sensor Network
WSRP	Wireless Spectrum Resource Pool
YANG	Yet Another Next Generation

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION 1

1.1 PROBLEM STATEMENT 4

1.1.1 Context Of The Problem 5

1.1.2 Research Gap 5

1.2 RESEARCH OBJECTIVE AND QUESTIONS 6

1.2.1 Research Questions 6

1.2.2 Objectives 6

1.3 APPROACH 7

1.4 RESEARCH GOALS 8

1.5 RESEARCH CONTRIBUTION 9

1.6 RESEARCH OUTPUTS 9

1.7 THESIS OVERVIEW 11

CHAPTER 2 LITERATURE STUDY 13

2.1 CHAPTER OVERVIEW 13

2.2 QOS IN WSNS 14

2.2.1 QoS Metrics 16

2.2.2 QoS Challenges In WSNs 17

2.3 RESOURCE MANAGEMENT IN WSNS 21

2.4 SDN 22

2.4.1 COMMUNICATION INTERFACES 23

2.4.1.1 NORTH-BOUND INTERFACES 25

2.4.1.2 SOUTH-BOUND INTERFACES 25

2.4.1.2.1 QOS CAPABILITIES IN OPENFLOW PROTOCOL 26

2.4.1.3 EAST/WEST-BOUND INTERFACES 28

2.4.2 SDN Controller Models 29
2.4.2.1 CENTRALISED SDN CONTROLLER ARCHITECTURES 31
2.4.2.2 DISTRIBUTED SDN CONTROLLER ARCHITECTURES 32
2.4.3 QoS in SDN Controllers 35
2.4.4 Existing Work on SDN QoS 36
2.4.5 SDN Opportunities In WSNs 40
2.5 SDWSN 41
2.6 QOS IN SDWSN 45
2.6.1 Energy management 48
2.6.2 Resource allocation and utilisation 53
2.6.3 SDN based routing mechanisms 57
2.6.4 Congestion Control 60
2.7 CHAPTER SUMMARY 64

CHAPTER 3 ACTIVE NETWORK MANAGEMENT STRATEGY 66
3.1 CHAPTER OBJECTIVES 66
3.2 SIMULATION TOOLS 66
3.2.1 ONOS 66
3.2.2 NS-3 68
3.2.3 OfSwitch 13 70
3.2.4 Modified SNMP 71
3.2.5 OpenFlow 72
3.3 EXPERIMENTAL SETUP 73
3.4 RESULTS 78
3.5 CHAPTER CONCLUSION 83

CHAPTER 4 DEVELOPING QOS STRATEGIES FOR EFFICIENT SDWSN SYSTEMS 84
4.1 CHAPTER OBJECTIVES 84
4.1.1 RESTCONF and YANG 86
4.2 EXPERIMENTAL SETUP 87
4.3 RESULTS 92
4.4 CHAPTER CONCLUSION 96

CHAPTER 5 DISCUSSION 97

CHAPTER 6 CONCLUSION 100

REFERENCES ……..104

ADDENDUM A CODE SNIPETTS 123

ADDENDUM B ONOS FRAMEWORK 127

LIST OF FIGURES

Figure 1.1 : An overview of WSN Architecture 2
Figure 2.1: SDN Architecture. 23
Figure 2.2: SDN Communication interfaces. 24
Figure 2.3: Centralised Controller Architecture. 31
Figure 2.4 : Flat Distributed Controller Architecture. 33
Figure 2.5 : Hierarchical Distributed Controller Architecture. 35
Figure 2.6: An Overview of SDWSN Architecture 41
Figure 2.7: Basic QoS Operations in SDWSN 43
Figure 3.1: ONOS interactions. 68
Figure 3.2: NS 3 Supported modules. 69
Figure 3.3: NS 3 integration to the OfSoftSwitch13 architecture 70
Figure 3.4: The OFSwitch13 Queue internal structure 71
Figure 3.5: OpenFlow switch architecture. 73
Figure 3.6: An Overview of the ANM Strategy Implementation. 74
Figure 3.7: Transmission Delay vs Number of Sensor Nodes. 79
Figure 3.8: Throughput vs Number of Sensor Nodes. 80
Figure 3.9: End-End Delay vs Packet Size. 81
Figure 3.10: Throughput vs Packet Loss. 82
Figure 4.1: SDWSN architecture with control and management layer. 85
Figure 4.2: RESTCONF protocol layering. 86
Figure 4.3: YANG modules within the ONOS architecture. 87
Figure 4.4: Modelled network system. 88
Figure 4.5: An illustration of abstracted SDN implemented Controller-QoS services. 90
Figure 4.6: Average hop-by-hop delay based on source data rate. 93
Figure 4.7: Received Packet Rate Success on Source Data Rate. 94
Figure 4.8: Associated Packet Drop Percentage per Hop-by-Hop Transmission Delay 95
Figure B.1: ONOS Framework……132

LIST OF TABLES

Table 2.1 : QoS tarameter summary and their service connection mode. 16
Table 2.2: QoS tools and their descriptions. 17
Table 2.3 : Comparisons between SDN Controller Models 29
Table 2.4: Advantages and disadvantages of some of the existing solutions 63
Table 3.1: System parameters and values. 78
Table 4.1: System parameters 89

LIST OF ALGORITHMS

Algorithm 3.1: The implemented status check function. 75
Algorithm 3.2: QoS Procedure for the proposed ANM 77
Algorithm 4.1: The Proposed Q-PSR Scheme. 91

CHAPTER 1 INTRODUCTION

The use of WSNs for mission critical applications have brought to light concerns regarding QoS provisioning since the mentioned applications require performance guarantees from systems in which they are implemented. The rapid growth in the number of users and applications using WSNs has led to WSNs becoming increasingly complex. Hence, this growth has triggered high demands on WSNs to provide QoS requirements for every user and application that uses them, as a means to enhance their operation. Even though WSN technologies have limitations, they possess the potential to support modern high demand network computing applications if they could be optimized accordingly. However, it is yet to be realized as to how WSN systems could support QoS provisioning in modern network application systems, since these technologies suffer in terms of computing capabilities. Hence, this work, makes reference to the need for efficient resource management to realize the required QoS in WSNs.

QoS refers to the capability of a network architecture to support and provide adequate services to particular network traffic. In essence, the objective of QoS is to implement some level of network engineering priority for maintaining efficient network operations by ensuring resource control and availability for different network services or applications. QoS metrics usually include: bandwidth, throughput, latency, delay, jitter, packet loss and coverage. However recently energy consumption and network lifetime have become standard part of QoS metrics. WSN architecture is designed in a manner that groups of specialised, relatively small networking nodes with the communication infrastructure (sensing circuitry, an approximate amount of memory, a microcontroller, a wireless

transceiver and a power source) for monitoring and recording conditions at diverse locations are expected to work together to perform application specific tasks ranging from object, animals and human tracking, intrusion detection, disaster management amongst others.

Figure 1.1 illustrates a WSN architecture with cluster communicating with the base station.

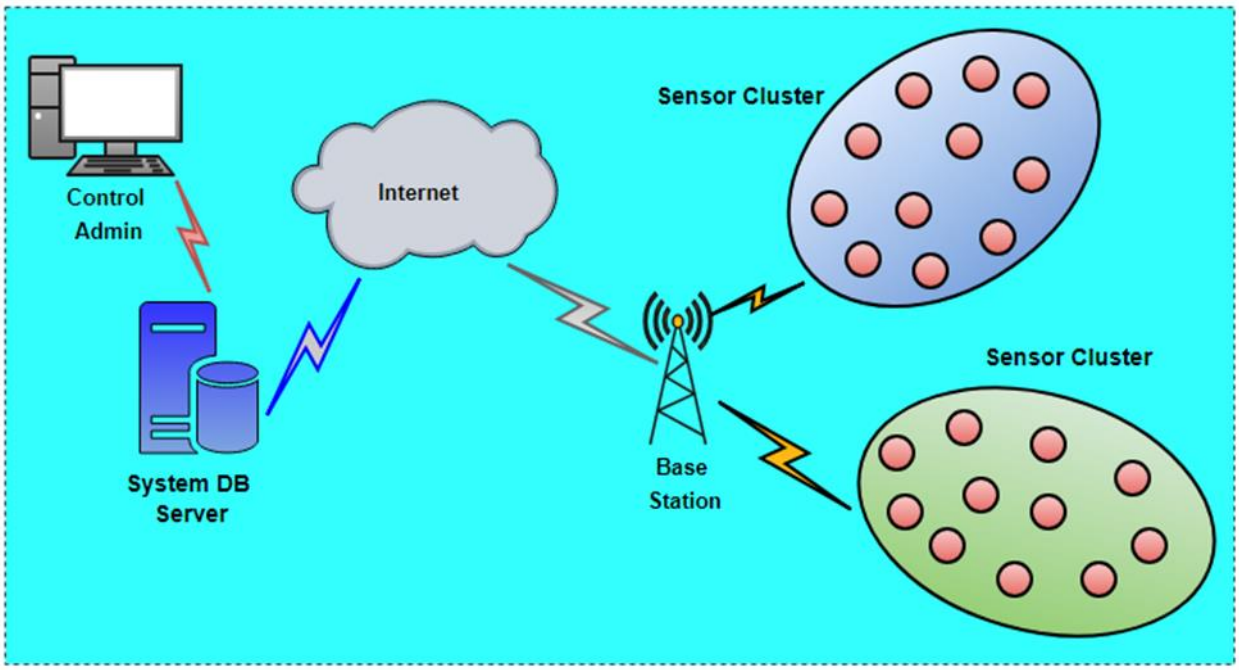

Figure 1.1 : An overview of WSN Architecture

Providing QoS for WSNs is a relatively challenging task because WSNs are severely constrained with regards to memory, processing capacities, communication capabilities and energy [1, 2]. To appropriately optimize WSN application technologies, a thorough understanding of challenges associated with their operations must be made. Computing constraints for these technologies includes, but are not limited to the following:

- *Incompatibility*: Sensor nodes are vendor specific hence they are not compatible with sensor nodes from different vendors due to the differences at the higher layers of the protocol stack [3, 4].
- *Configurability, maintainability and manageability*: Once deployed, WSNs are difficult to configure, maintain and manage since sensor nodes can be deployed in

unfriendly territories or locations, it becomes impossible to replace their power sources and they are configured manually hence there will not be any modifications to their configuration [5, 6].

- *Resource underutilisation*: Since WSNs are application specific, they tend to underutilise resources [7, 8, 9].

- *Security and privacy concerns*: In this aspect, WSN tend to suffer intrusion since wireless communication can be disturbed or attacked using intelligent intrusion methods. As a results, this can lead to Denial of Services (DoS) attacks, resources limitations and data alterations. Given these security issues and deployment locations that are not secured, the whole network can be easily accessible and compromised [10, 11, 12, 13].
- *Computing and communication limitations*: Limited computing or transmission resources may lead to high delays, poor system responses due to insufficient processing memory, and loss of data due to traffic congestion. High loads on sensor nodes results in high loss sensor energy. Network reliability can also be affected as methods for extending sensor communication capabilities are installed or incorporated to the network [14, 15, 16, 17].
- *Energy limitations*: Sensor batter power depletes with time due to operations on the sensor nodes. As a results sensor nodes may be discarded out of the network infrastructure due to energy depletion. Energy is therefore, the most critical challenge is sensor networks as it directly affect the lifetime of the system [18, 19, 20].
- *Sensor fault tolerance*: A well-developed network infrastructure must be resilient to faults and also promote efficient data communication. The infrastructure must be able to survive and transmit meaningful data even when some sensors are discarded from the network due to physical challenges [21, 22, 23].

However, it is yet to be realized as to how WSN systems could support QoS provisioning in modern network application systems, since these technologies suffer in terms of computing capabilities. Hence, this work, makes reference to the need for efficient resource management to realize the required QoS in WSNs.

Nevertheless the solution to these challenges or limitations can be provided by introducing SDN in WSNs. SDN is a paradigm that permits management of network services through abstraction of lower level functionalities. The basic idea of SDN is to separate the data layer from the control layer in network devices and to centralize control of the network operation [24]. SDN is envisioned to reduce the complexity of network configuration and management of WSN by introducing evolvability and simplification to the network. SDN promises flexibility, centralized control and open interfaces between nodes, enabling an efficient adaptive network [25].

SDWSN is a network computing paradigm for applying SDN strategies to WSNs with the purpose of enhancing their technological applications. This networking paradigm, promises to evolve how WSN systems operate for diverse application through the concept of programming. The SDWSN concept is aimed at, dedicating the controlling functionality of a WSN system to a global controller for compute intensive tasks and therefore maintain the underlying sensor nodes to only perform data forwarding. The controller manages numerous network devices, services, topology, traffic paths, and packet handling (QoS) policies. Hence, SDWSN is envisioned to; reduce network complexity, simplify device configuration and improve network management by introducing adaptive computing evolution and simplicity to the network.

1.1 PROBLEM STATEMENT

This subsection provides a detailed description of experienced challenges within the proposed area of research, especially for mission-critical application systems. These experienced challenges are described in the context of how QoS can improve WSN network operations using software-oriented strategies. Furthermore, the criticality of implementing QoS requirements to modern network computing systems is provided with the aim to support the systematic opportunity that can be achieved from the proposed research approach.

1.1.1 CONTEXT OF THE PROBLEM

WSNs have rapidly progressed over the years, they are now applied in health care systems, home automation, security surveillance, disaster management and more. This rapid growth for WSN systems that are competent to meet sensitive application demands, requires necessary operational strategies to fulfil these tasks. This brings us to the need to develop and maintain highly responsive systems that promotes flexibility and network services abstraction. These system must also be able to support large numbers of users and critical applications with efficiency and reliability. However, these technical requirements, suggest very sophisticated WSN systems that entail capable resource access and management capabilities. Network QoS is an integral aspect of communication application systems, since it directly impact internal and external networking operations. However, network QoS must be intelligently and flawlessly implemented so that it promotes the efficiency of the system as well as improve the overall performance of the system. QoS and efficient resource allocation and utilization are amongst some of the most important requirements that users and applications demand.

The use of WSN in ad hoc applications has increasingly become popular due to their capacities to be deployed in inaccessible area and their nature of affordability. The rapid technological growth requires that these systems be equally capable to meet modern technological directions. To achieve this, WSN application systems must comprise of smart devices with efficient resources to sustain current technological needs. This implies that, the manner in which these systems are designed and developed, must be able to support on-demand techno logical transitions for better computational performances. Thus, the planning phase must incorporate well-articulated strategies for developing highly competitive and reliable computing systems.

1.1.2 RESEARCH GAP

Sensor nodes are severely limited in terms of resources and the network lifetime and data accuracy depend on energy usage, therefore it is of paramount importance to provide

abundant Quality of Service. It is not easy to maintain the desired QoS provisioning because WSNs recurrently go through changes, the changes may be due to the network topology change, load imbalance, faulty nodes and unstable wireless links. Consequently it is important to develop QoS techniques that are dynamic, energy saving and resilient to frequent changes. Accordingly this research proposes the use of SDN in WSNs as a means to solve the problems experienced in WSN, to provide the overall desired QoS and efficient resource management.

1.2 RESEARCH OBJECTIVE AND QUESTIONS

This subsection provides a list of questions that came about when conducting this study, with the objectives that have been developed to address these questions as well as provide a direction towards the implantation of the proposed approach.

1.2.1 RESEARCH QUESTIONS

- What is the correct way to synchronize the network global-view at the controller in order to ensure efficient resource management and utilization?
- Can SDN facilitate efficient resource allocation and utilization in WSN?
- What can be done to prolong the Network lifetime for SDWSN?
- Would the OpenFlow protocol be able to fulfil the QoS requirements in such a constrained network?

1.2.2 OBJECTIVES

1. To develop a SDN oriented status check function that will facilitate efficient network knowledge acquisition and reporting.
2. To exploit the capabilities of check function to improve network resource allocation and utilisation.

3. To develop an OpenFlow based Active Network Management (OF-ANM) QoS scheme using SDN strategies to efficiently manage sensor traffic congestion.
4. To formulate and develop a QoS Path Selection and Resource-associating (Q-PSR) scheme for adaptive load balancing and intelligent resource control for optimal network performance.

1.3 APPROACH

This work will focus on developing software defined methods that can be implemented to improve and facilitate the provisioning of QoS and resource allocation and management in WSNs. The approach will then be implemented as an effort to facilitate the guaranteeing of QoS in WSN application or production networks. Based on the proposed approach, this work implemented the following plan:

1. Literature Study
    a) Carried out a detailed literature study on previous and current approaches of WSN application technologies.
    b) Technical challenges on these application technologies are analysed and discussed in an effort to formulate better methods for designing and implementing efficient WSN systems.
    c) Ideas for applying SDN strategies in WSN application technologies were also suggested with the aim to improve QoS implementation in WSN systems.
    d) Based on the compared WSN approaches, efficient software-oriented strategies for implementing QoS in these systems were identified.
    e) A constructive research methodology was then developed from these investigations, which forms the core aspect of the proposed study.
2. Develop software informed and reinforced learning motivated network techniques that supports efficient QoS assurance in SDWSN systems.
    a) A resource-aware OpenFlow-based Active Network Management (OF-ANM) QoS scheme that uses Software Defined Networking (SDN)

strategies was developed as a means to apply QoS requirements for efficient network management in WSNs.

b) To further improve the performance of the system, a QoS Path Selection and Resource-associating (Q-PSR) scheme for adaptive traffic load balancing was developed.

c) All the proposed strategies were developed with functional cooperation to the ONOS controller to achieve efficient network performance.

3. Implementation

a) The proposed approach was implemented using NS-3, with an ONOS controller implemented as the central intelligence of the system.

b) Other related strategies were carefully studied and implemented for comparison purposes against the proposed approach.

c) Achieved results from the proposed approach were critically analysed and evaluated against other related strategies, to establish its improvement aspect.

4. System Analysis and Discussion of Results

a) Using all the proposed strategies, an actual SDWSN system was simulated, wherein achieved results were analysed and discussed as a means to validate its application potential.

b) Achieved implementation strategies from the proposed methodology are compared to other existing results, wherein critical analysis of these systems are made.

c) Results a critically analysed and discussed in terms of how the new developed strategies contribute to the area and knowledge in generally.

1.4 RESEARCH GOALS

The main goal of this research is to develop adaptable strategies that use SDN programmability for ensuring efficient resource management and ensuring QoS provisioning in Wireless Sensor Networks. These strategies apply flexible QoS requirements, perform

load balancing, efficient resource alignment and ensure congestion avoidance within the Software Defined Wireless Sensor Network.

1.5 RESEARCH CONTRIBUTION

Subsequent to the fact that provisioning of good QoS is a need in any computing network, this work will enormously contribute to the area of WSNs largely on guaranteeing of communication and processing stability of these technologies. If successful Application Programming Interfaces (APIs) for ensuring QoS can be developed and implemented on WSNs especially for resource allocation and management, the overall WSN performance can be improved. The candidate also emphasize that upon rich QoS provisioning in WSNs, greater resource utilization can be experienced. This work will also largely contribute to the Research and Development (R&D) aspect of the development or improvement of techniques and protocols for ensuring QoS.

Furthermore, the candidate highlights that this work will open up programmable opportunities for handling sensor communication services as well as sensor node connectivity and coverage services of WSN technologies. Since QoS directly reflects the performance of any computing system, realization of good QoS will bring about satisfaction to customer needs especially on productive networks. Lastly but not concluding, this work will directly contribute by increasing new research opportunities towards the improvement of quality guaranteeing services as well as optimizing QoS protocols in WSNs.

1.6 RESEARCH OUTPUTS

B. B. Letswamotse, R. Malekian, Chi-Yuan Chen and K. M. Modieginyane, “Resource-aware QoS Support for Efficient Congestion Control in Software Defined Wireless Sensor Networks.” IET Networks. In press {Accepted}

B. B. Letswamotse, R. Malekian, Chi-Yuan Chen and K. M. Modieginyane, "Software Defined Wireless Sensor Networks (SDWSN): A Review on Efficient Resources, applications and technologies," Journal of Internet Technology. In press {Accepted}

K. M. Modieginyane, B.B. Letswamotse, R. Malekian, A. M. Abu-Mahfouz, "Software defined wireless sensor networks application opportunities for efficient network management: A survey", Computers & Electrical Engineering, 2017, ISSN 0045-7906, https://doi.org/10.1016/j.compeleceng.2017.02.026.

K. M. Modieginyane, R. Malekian and B.B. Letswamotse, "Flexible Network Management and Application Service Adaptability in Software Defined Wireless Sensor Networks" Journal of Ambient Intelligence and Humanized Computing. In press {Accepted}

A.C. Jose, R. Malekian, B.B. Letswamotse, "Improving smart home security, integrating behaviour prediction into smart home", International Journal of Sensor Networks, {Accepted}

B. B. Letswamotse, K. M. Modieginyane, and R. Malekian, "Adaptable QoS Provisioning for Efficient Traffic-to-Resource Control in Software Defined Wireless Sensor Networks" {Submitted, to Springer}

K. M. Modieginyane, R. Malekian and B.B. Letswamotse, "Increased Network Knowledge Using a Status Update Algorithm for Monitoring Applications in Software Defined Wireless Sensor Networks". {Submitted, to Springer}

B. B. Letswamotse, K. M. Modieginyane, and R. Malekian, "SDN based QoS provision in WSN technologies," in Southern Africa Telecommunication Networks and Applications Conference (SATNAC), George, South Africa, Sep. 2016.

K, M. Modieginyane, B.B. Letswamotse and R. Malekian, "Optimized Network Provisioning in Resource Constrained Wireless Sensor Networks for Monitored Environments: A Software Defined Networking Approach, George, South Africa, Sep. 2016"

1.7 THESIS OVERVIEW

This work provides the technical scope of implementing efficient network QoS in SDN based WSN application systems. The importance of assuring QoS in computing networks is thoroughly discussed and methods for applying this networking aspect are outlined. The inception of SDN in WSN application systems is discussed with the effort to develop strategies for implementing QoS requirements in these systems. This technical design for ensuring network QoS is developed to facilitate efficient network resources sharing in SDWSN systems. This forms the fundamental goal of the research. Related research outputs and developments are carefully studied and analysed to build towards the foundational objectives of this work. As a result, efficient methods and algorithms using SDN strategies are developed to ensure this technicality. The importance of network QoS is therefore aligned to the proposed approach of this work. Technical achievements from this work, are thus discussed with regards to the proposed and developed algorithms in this work. The impact of scientific contribution of this work is explained as associated with all related areas of applications in SDWSN applications. The entire thesis is organised as follows:

- ***Chapter 2***: This section provides critical analysis of related work within the scope of the proposed approach. From these analysis, strong foundations and directions are developed to support the formulated research objective, so as to ultimately achieve the goal of this research.
- ***Chapter 3:*** In this chapter an SDN programmable OpenFlow-based strategy that ensures QoS provisioning by efficiently managing network resources and managing and controlling network congestion in WSNs is developed and implemented.
- ***Chapter 4:*** In this chapter controller-based QoS strategies to actively manage network resources by implementing flexible SDN strategies to perform resource alignment based on different networking tasks are formulated and implemented, programmability strategies to perform path computation activities to control sensor path traffic are applied and service-oriented abstraction functionality to perform scheduling activities for various QoS requirements are developed and implemented.

- ***Chapter 5***: This section provides detailed discussions of all the implemented strategies and what they bring to the proposed research. Discussions are made along existing approaches and technologies within the area.
- ***Chapter 6***: This section provides general research conclusion with details relating to the developed methods and all achieved results. The relevance of this research is provided with its contributions in the area of SDWSN.

CHAPTER 2 LITERATURE STUDY

2.1 CHAPTER OVERVIEW

This chapter reviews the general WSN limitations with regards to QoS and Resource Management. Moreover, it reviews the concept of SDN, its challenges and benefits and how QoS and Resource Management issues were addressed using SDN. Finally the concept of SDWSN is introduced together with the existing literature on SDWSN.

In Section 2.2 the importance of WSNs QoS provisioning and the challenges faced by sensing technologies are discussed.

Section 2.3 provides a brief discussion on resource management in WSNs.

In Section 2.4 an extensive discussion on SDN, the potential of SDN, centralised and distributed controller models, communication interfaces, an in-depth discussion on different OpenFlow versions, QoS capabilities of some of the SDN controllers and finally some of the work done in providing QoS in SDN networks is provided.

Section 2.5 discusses SDWSN, the benefits that SDN paradigm brings to WSNs, description of how an SDWSN architecture should handle QoS requirements and finally, it discusses the existing work on SDWSN provisioning sub sectioned into: energy management, congestion management and resource management.

Finally, section 2.6 summarises what has been achieved in this chapter

2.2 QOS IN WSNS

Over the years providing the desired QoS in WSNs and at the same time reduce energy consumption and hence extend the lifetime of the network has been the focal point and the greatest challenge that researchers in the networking field faced.

Generally, guaranteeing a certain QoS level in WSNs is a difficult matter due to the constraints in available resources and capabilities, such as energy, memory, bandwidth and processing capabilities, unpredictable nature of wireless links and often huge number of sensor nodes in WSN [26]. Current development of WSNs indicates that they are becoming increasingly complex and consequently users and applications require more bandwidth, lower delays and increasing reliability [27]. Therefore, QoS support has become one of the key factors of WSNs.

Authors in [28] highlighted that the basic QoS for achieving energy efficiency and prolonging network lifetime are found in the correlation between deployment, coverage and connectivity since QoS is affected by topology and coverage and these two determine connectivity.

Meeting the desired QoS requirements also depend on the type of data delivery model. A delivery model could either be continuous, query driven or event driven. In the continuous model, the sink node receives data from the sensor nodes at a predetermined rate. Query based applications provide interactions between the sink node and the sensor nodes, whenever the sink node require information on a certain event occurrence, it generates and submits a query to the sensor nodes. These applications are delay tolerant, despite the fact that they are mission critical, they are also non end-to-end. The event driven applications are interactive, mostly delay tolerant and non-end-to-end. Sensors detect a certain event occurrence and have to act appropriately since the sensor nodes and sink nodes are affected by certain events and

the data submitted by sensor nodes are essential and have to be sent to the sink node in a quick and reliable manner [29].

Even though guaranteeing QoS is one of the most important measures in computing networks, its potential has been studied for the past decade but the solutions for QoS provision in WSNs are very limited, since most researches focused on single challenges such as energy, routing, network topology and other inherent problems instead of providing overall QoS Provisioning. In addition to this, sensor programmability in WSNs has not evolved quite enough to clearly address and improve QoS guarantee in these technologies, hence it remains a challenge to develop new efficient QoS mechanisms which will support this approach. SDWSN promises to evolve how wireless sensor network systems operate for diverse application systems through the concept of programming.

For production networks to realize optimum network results especially for services or processes that are customer aligned, some service quality needs to be implemented at all costs. In this regard, there should not be any compromise in sustaining or optimizing some critical QoS parameters especially those that forms the backbone of a competitive and productive network experience. The following QoS parameters are some of the most critical in terms of adhering to an optimum network performance:

- Energy Efficiency
- Network Lifetime
- Network Latency
- Reliability
- Bandwidth
- Network Coverage and Connectivity
- Packet Loss and Packet Delay Variation

Table 2.1 gives a summary of QoS parameters and their respective supported service modes unto which they apply, such. These are some the critical QoS parameters that are negotiated either on local or on end-to-end devices.

Table 2.1 : QoS Parameter summary and their service connection mode.

QoS Parameters	Mode of Service		Point of Negotiation	Value Determination	
	on-connection	*non-connection*		*Negotiated as per-connection*	*Not negotiated during connection*
Security/ Protection	both		local	Yes	
Priority	both		local	Yes	
Network Resilience	connection		n/a		Yes
Transit Delay	both		end-to-end	Yes	
Throughput	connection		end-to-end	Yes	

QoS provisioning is the most important aspect in real-time, multimedia and mission critical WSNs that require guaranteed performance in order for them to function properly. However, due to the increasing number of users and applications, soon it will not be sufficient to have QoS only as an option in WSNs, but rather to become a standard in WSNs.

2.2.1 QoS Metrics

QoS tools are most essential in networking since their main purpose is to manage, classify, control and prioritize as well as to direct network traffic. Even though QoS tools maintain the same goal in terms of customizing the network's capability to improve the overall network performance, they use different approaches to achieve this. With the understanding that, network traffic is not equally created or managed, some traffic must be prioritized over others. Without proper network management and control, it is difficult to achieve or maintain the best network performance, which leads to a better network experience. These aspects of network computing highlights the importance of network service quality or QoS implementation as a critical part of networking. In this regard, Table 2.2 presents the critical tools to be implemented in a network computing environment to achieve best results.

Table 2.2: QoS Tools and their descriptions.

QoS Measures	Description	Tools/Mechanisms
Classification	Flow types are identified and marked if required.	Access Control Lists (ACLs), policy-based routing, Committed Access Rate (CAR), and Network-Based Application Recognition (NBAR).
Congestion management	Flows are queued and serviced in various ways to provide special treatment to certain flows.	Priority Queuing (PQ), Custom Queuing (CQ), Weighted Fair Queuing (WFQ), and Class-Based Weighted Fair Queuing (CBWFQ).
Scheduling and Queueing	A queue is prevented from filling, to allow high-priority traffic to enter the queue.	Weighted Random Early Detect (WRED)
Shaping/ Policing	Limiting the bandwidth that a flow(s) uses by monitoring and regulating flows to ensure that they do not violate their traffic contract.	Leaky Bucket and Token Bucket
Link efficiency	Providing a mechanisms for alleviating delay experienced on lower-speed links.	Compressed Real-Time Protocol header

2.2.2 QoS Challenges In WSNs

The challenges faced in QoS provisioning in different WSN technologies are discussed in this subsection.

Maintaining the desired QoS provisioning in WSNs is challenging because of the recurrent changes in the network topology due to faulty, dead nodes or unstable wireless links which leads to inherently inaccurate available state information for routing. Sensor networks incur bandwidth limitations and load imbalance since the sensed data is collected from several sensor nodes to a base station. On many occasions routing in sensor networks fulfil delivery requirements at the expense of energy efficiency. In this regard, even though multi-hops are energy efficient, it incurs an overhead that may delay packet delivery. Even though buffering in routing is beneficial as it assists in obtaining numerous packets before forwarding them, there is a buffer size restriction which increases the delay variation that packets transmitted on different paths and even on the same path experience thus hindering the assurance of QoS requirements [30].

IWSNs are time critical, if information is delayed the possibility of wrong decisions being made exist. Recurrently changing network topology is a challenge since IWSNs incur unstable wireless links, faulty and dead nodes. The wireless links are affected by signal interference, dust and noise amongst others [31].

Wireless Body Sensor Networks (WBSNs) employ sensor nodes hence the risk of packet drop and errors exist. Moreover, in these systems the probability of packet loss is increased by in an on body dynamic path loss and patient's mobility, these factors also introduce dire repercussions to the WBSN channels. Channel characteristics can also differ with body size and posture. The network topology might change frequently due to link failure, power failure, and node mobility these unstable conditions make routing and medium access very challenging. Unreliable nature of wireless links may be disastrous in emergency packet transfer since some packets in WBSN are time critical and require real time attentions. Hence it increases complexity of QoS support. WBSNs may unnecessarily consumes much needed energy when processing redundant data. They also experience and traffic imbalance since data flow from many sensor devices to a small number of sinks. The heterogeneous environment in WBSN makes QoS support more complex and challenging [32].

Wireless Sensor and Actuator Networks (WSANs) are bandwidth constraint as a result some data transmissions may experience large delays, resulting in poor QoS provisioning. Consequently, memory constraints lead to data packets being dropped before nodes send them to the destination successfully. The platform heterogeneity makes it difficult to utilise resources in an efficient manner and to achieve real-time and reliable communication between various nodes. In WSANs some nodes may be mobile, this leads to dynamic changes in the network topology since sensor/actuator nodes may be added or removed at runtime. This may be due to dead nodes, energy saving nodes and node mobility. Various applications may need to share the same WSAN, actuating both regular and irregular information. This requires WSANs to support service differentiation in QoS provisioning [33].

Wireless Multimedia Sensor Networks (WMSNs) present high traffic and require more bandwidth to process multimedia data. Some applications may need various sensors to monitor the events and to capture images and videos of objects in motion. These applications produce heterogeneous data from different types of sensors at varying sampling rates based on different QoS limitations and delivery models. Hence, these heterogeneous WMSNs may impose significant challenges for the provision of QoS and WMSN routing protocols. Sensor nodes consume a large amount of energy when communicating and a significant amount of energy is also wasted when processing redundant data. This pose significant challenges for QoS provisioning. The network topology may change recurrently due to node failures or link failures caused by mobility. QoS provisioning is a challenge since nodes are added or removed from the network. WMSNs use radio as the communication medium and wireless links. Both are not reliable due to inherent features associated with them. Wireless links are affected by environmental factors in their surroundings [34].

Providing good QoS level for robotic sensor networks such as drone-based sensor networks proved to be challenging since these networks operate in more extreme environmental conditions such as earthquakes, chemical disaster environments and flooding. These networks experience QoS constraints in terms of data traffic since packets may be droppe

before they reach their destinations due to congestion, broken communication links caused by node failures or node mobility, and bit errors caused by noise, interference, distortion or bit synchronization. Moreover, mobility pose another challenge to QoS provisioning since drones fast mobility leads to unpredictable links and there are delays resulting from path planning and navigation algorithms when mobile drones are relaying information. Finally drone-based sensor networks may incur connectivity challenges due to their fast mobility [35, 36].

The involvement of human participation in crowd sensing poses QoS challenges since certain trust issues arise. Participants are likely to provide incorrect or even fake data to the system. Meanwhile, mischievous users may deliberately infect the sensing data for their own advantage. The lack of control mechanisms to guarantee source validity and data accuracy can lead to information credibility issues. Redundant sensing may lead to data inconsistency. One may find a group of collocated smartphones with different sensing and computing abilities running the same algorithm and sensing the same event can obtain different inference results, thus causing inconsistency issues. Delivering the sensed data from distributed participants to the backend server is another challenge due to a range of crowd sensing characteristics, such as the low bandwidth of wireless communication, frequent network partitioning caused by participants mobility, and numerous energy-constrained devices [37].

Since traditional SDN protocols were essentially designed for wired networks, it remains a challenge to effectively apply them as they are in WSNs. Routing protocols for wired network technologies, have the advantage of physical energy sources and hard-connected infrastructure, whereas the opposite is experienced in wireless network technologies. In WSNs, routing plays an integral part of the whole system architecture, connectivity, network traffic, performance and lifetime. Other aspects of routing in WSNs directly relates to, QoS provisioning, energy management, scheduled processes, etc.

Consequently, a well-planned WSN system must be sensitive on the manner of routing protocols used based on the actual system application. Therefore, an opportunity to effectively use traditional SDN protocols in WSNs, is somehow by means of modifying these protocols to well suit the type of routing applied by WSNs. Other methods of using these protocols could be achieved in terms of developing the same protocols but with the focus for wireless technologies or by integrating wireless network technologies with other systems such as in hybrid architectures. Accordingly, QoS mechanisms designed for WSNs technologies should be scalable, dynamic, energy efficient and robust. The addition or removal of sensor nodes should not affect the QoS of a WSN system.

2.3 RESOURCE MANAGEMENT IN WSNS

Due to the resource constraints experienced in WSNs, leading to resource underutilization it became important to employ algorithms and techniques to ensure efficient utilization and management of network resources. Researchers all over the world came up with various solutions.

Authors in [38] proposed a resource management protocol (RMP) for resource constrained WSN applications, the objective of this protocol was to improve the efficacy by borrowing resources from a resource management server that has unlimited access to the network's resources in a manner that is general and sufficiently expandable to solve the resource limitations problem in some of the Wireless Sensor Networks. Their experiments indicated that the RMP is both practical and effective in solving the resource constraints experienced in ZigBee applications.

Authors in [39] proposed the Distrinet Adaptive Resource Management Architecture (DARMA), an insubstantial component-based service platform for WSN. DARMA provides basic, yet versatile, Service Level Agreements (SLAs) customized to meet the sensor network environments' needs. Their approach provides autonomic service managers to

enhance the manner in which WSN resources are used to satisfy multiple, concurrent service requests. They implemented the proposed architecture to prove its suitability by using Loosely Coupled Component Infrastructure (LooCI) and the SunSPOT platform. Evaluations show that DARMA is capable of enhancing the manner in which multiple services share components. Moreover fault tolerance is achieved though using the DARMA SLA language, thus sensor nodes lifetime is extended.

2.4 SDN

SDN is a networking model that decouples the control logic and data logic to enable flexible control and configuration of underlying network devices. This paradigm allows network administrators to automatically and dynamically manage and control a large number of network devices, services, topology, traffic paths, and packet handling (QoS) policies using high level programming languages and Application Programming Interfaces (APIs) [40].

Simplifying the network management process has been main motive behind the adoption of SDN. In SDN network administration problem is altered into a network programmable problem. The application of SDN in WSN is envisioned to simplify the management of these complex networks. Deployment of new protocols, network control and management solutions on existing infrastructure become as easy as installing a program on a computer due to the flexibility introduced by SDN [41].

In Figure 2.1, the data plane and the control plane are separated and the architecture comprises of three distinct layers: Application, Control and Infrastructure layers. Users and business applications that make use of SDN communication facilities are place on the application layer via the north bound APIs, the control layer offers the combined control functionalities that manages the network forwarding behaviour through the southbound APIs. Finally, the forwarding elements that are responsible for switching and forwarding packet are placed on the infrastructure layer [42].

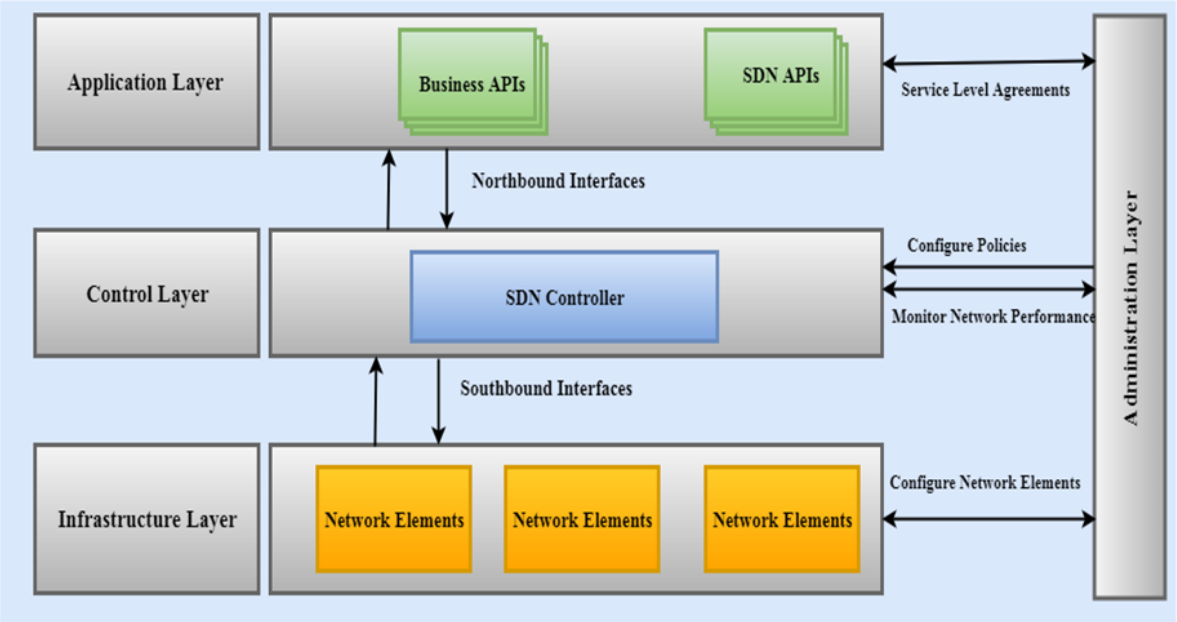

Figure 2.1: SDN Architecture.

OpenRoads is a testbed that lets multiple networking experiments to be carried out simultaneously in a production network, it envisioned the move from wired to wireless infrastructure and uses OpenFlow as a way to control access points and switches in the network data path [43].

2.4.1 COMMUNICATION INTERFACES

In SDN communication interfaces provide a communication channel between the networking components of the system. Figure 2.2 illustrates the nature of communication that these interfaces offer. Whereby northbound APIs facilitate the communications between control layer elements and the elements (applications) in the application layer. East/westbound APIs provide the communication between control layer elements and finally the southbound APIs provide communication between the control layer elements and the underlying forwarding layer elements. The detailed discussions on these APIs is presented in the following subsections.

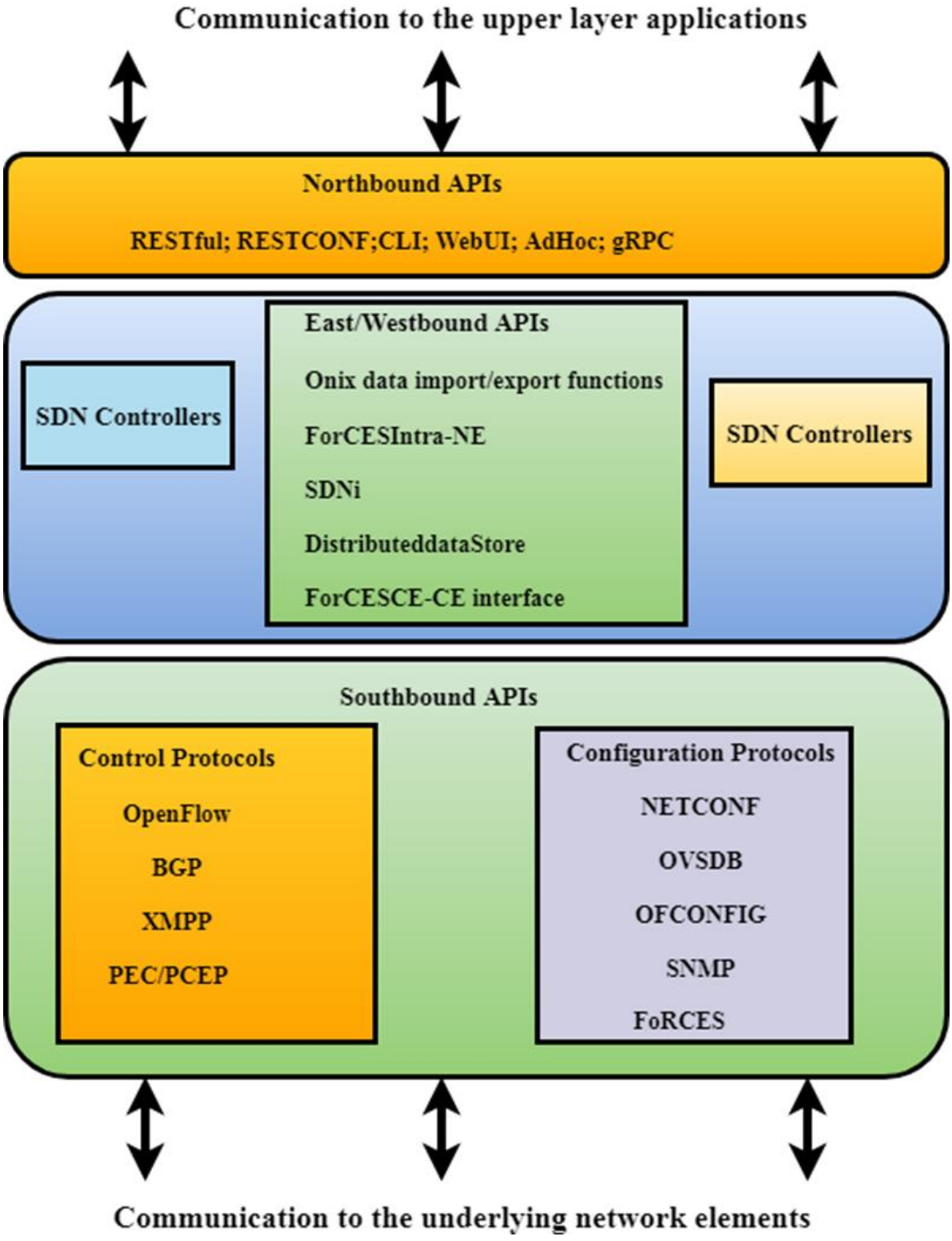

Figure 2.2: SDN Communication interfaces.

2.4.1.1 NORTH-BOUND INTERFACES

The northbound interface is a software interface between the control layer and the application layer, it enables the communication between SDN applications and the controller modules. This interface consist of protocols that ensure policy enforcement and conflict resolution. Unlike in southbound interfaces, the northbound interface does not have a standardised interface. The northbound interfaces that are supported by many of the controllers include: REST/ RESTFUL API [44], RESTCONF [45], gRPC [46] and Adhoc API. Prichard et al. [47] explored the possibility of improving the northbound interface communications by addressing metadata limitations in REST. They also worked toward identifying potential platforms for developing a metadata framework.

2.4.1.2 SOUTH-BOUND INTERFACES

The southbound interface is a communication channel that is situated between the control layer and the infrastructure layer. It enables the communication between data plane elements and the controller, in that manner, the data plane elements are enabled to discover the topology of the network, define network flows and execute northbound API requests. This interface consists of protocols that determine how the forwarding elements are configured and managed by the controller in terms of inserting forwarding rules in the flow tables and how the network statistics are collected. The southbound protocol that already exist include: OpenFlow [48], NETCONF, Simple Network Management Protocol (SNMP) [49], Open VSwitch Database Management Protocol (OVSDB) [50], Forwarding and Control Element Separation (FoRCES) [51], Border Gateway Protocol (BGP) [52], OpenFlow-Config (OF-Config) [53], Path Computation Element/Path Computation Element Communication Protocol (PCE/PCEP) [54] and Extensible Messaging, and Presence Protocol (XMPP) [55]. OpenFlow is the most commonly used southbound interface, as a result this protocol is discussed more in subsection 2.4.1.2.1 below.

2.4.1.2.1 QOS CAPABILITIES IN OPENFLOW PROTOCOL

OpenFlow is the most popular standard protocol used to facilitate the communication between the control layer elements and the forwarding plane elements. An OpenFlow-enabled switch performs packet forwarding, switching and look-ups, it comprises of three parts: a flow table, secure (OpenFlow) channel and OpenFlow protocol. An OpenFlow-enabled switch keeps numerous flow tables that are holding a list of flow entries in which each flow entry consists of header\rule field, counters/statistics field, and an action field which contains a set of instructions to apply to packets matching the values in the header/Rule field [56].A secure channel is an interface that uses Transport Layer Security (TLS) for encryption, it connects the OpenFlow-enabled forwarding layer elements to a remote controller. The switches/ data plane elements are configured and managed by the remote controller over this interface.

Moreover, the controller receives switch state information and network events from the switch and also sends packet-out messages through the secure channel [57]. Controller-to-switch, asynchronous, and symmetric are the only message types that are supported by an OpenFlow protocol, all these messages are forwarded through the secure channel [58]. Controller-to-switch messages are established by the controller and are used to directly extract the switch state information. Asynchronous messages are established by the switch and are used to inform the controller with network events and modifications to the switch state. Symmetric messages are established by either the controller or the switch and are conveyed without requisition.

Each OpenFlow version brought along some QoS-related changes and features implemented in various versions. In OpenFlow 1.0 [59] an OpenFlow enabled switch provides a basic QoS support through the enqueue mechanism. This mechanism forwards packets via a queue attached to a particular port. Depending on its port, an OpenFlow switch can have one or more queues attached to a port and be used to map flows on it. Flows mapped to a specific queue are treated in accordance to the queue's properties and configuration (e.g. min rate or max rate), this equips the controller to query the queue information of a switch. The

forwarding behaviour of the queue is used to provide QoS support but it is determined by the configuration of the queue which is out of scope for OpenFlow 1.0.

In OpenFlow 1.1 [60] the enqueue mechanism is modified by adding the set-queue action which puts the queue id for the packets. When output action is used to send the packet to a port, the queue id decides which queue attached to that port is used for packet forwarding. The previous versions of OpenFlow only exposed an abstraction of the single table to the controller, OpenFlow 1.1 supports an adaptable pipeline with multiple flow tables. The visibility of multiple flow tables has advantages that are beneficial for QoS provisioning. Moreover, it performs VLAN tagging and matching and MPLS labels and traffic classifications. OpenFlow 1.1's tagging support has detailed actions for adding, removing and modifying VLAN headers. Similarly, it supports adding, removing and modifying actions for MPLS shim headers.

The QoS related enhancements in OpenFlow 1.2 [61] include: the controller responsibility change mechanism to support multiple controllers incase of failover; improving the flexibility of VLAN matching; ability for a controller to query all the queues in a switch; a new experimenter queue property; a new max-rate queue property and there is also a flow action included for rewriting the DiffServ CodePoint bits part of the IP ToS field in the IP header

Prior versions of OpenFlow only allowed a MPLS shim header to be inserted after all VLAN tags in the packet. OpenFlow 1.3[62, 63, 64, 65] eliminates this limitation by allowing the order of the tagging operations to determine the final order of tags in a packet. OpenFlow 1.3 also introduces meter tables consisting of meter entries for defining per flow meters. The per-flow feature is based on a new flexible meter framework, which comprehends the capabilities of multiple metering functionalities to define complex meters. As opposed to queues attached to a port, per-flow meters are directly attached to flow entries and are used to measure and control the rate of packets. Per-flow meter can be coupled with per-port queues (Set-Queue) to execute complex QoS frameworks since per-flow meters permits

OpenFlow to execute various simple QoS functionalities such as rate limiting the packets forwarded to the controller.

OpenFlow 1.4 [66, 67] introduces QoS support for distributed control systems by providing the flow monitoring feature/framework which allows a controller to keep track of all the modifications made by other controllers to any subsets of the flow tables in real time. Consequently a controller is enabled to define numerous monitors, each selecting a subset in the flow table. Each monitor includes a table id and a match pattern that defines the observed subset. When there are changes in the flow tables in terms of added, modified or removed flow entry in one of the subsets defined by the flow monitor, an event is conveyed to the controller to notify it of the change.

In OpenFlow 1.5 [68, 69] the meter instruction which was used for metering in prior versions is replaced with meter action. Consequently, multiple meter can be attached to the same flow entry and meters can be used in group buckets. Other QoS functionalities include the extension that gives the controller the knowledge of all the switch to controller status connection through the controller connection status. This allows the master controller to observe the network states of its subordinate controllers and also detect the control network partitions.

2.4.1.3 EAST/WEST-BOUND INTERFACES

East/West-bound interfaces are the protocols that coordinate the interactions between controllers in distributed SDN architectures. The controllers in the distributed SDN architectures use this interface to notify the other controllers of the changes experienced on their side of the network, this helps the other controllers in building the global network view.

These interfaces also provide a failover mechanism for when one controller fails the SDN controllers can exchange information about their capacities so that the network load can be shared fairly. There are a number of proposals which defined the interactions between controllers such as Onix data import/export functions [70], ForCES Intra-NE [71], SDNi

[72], distributed dataStore [73] and ForCES CE-CE interface [74].

2.4.2 SDN CONTROLLER MODELS

Since the emergence of the SDN paradigms researcher came up with different controller models. The models are categorised into single centralised controller and distributed controller models. Distributed controller models are designed to be either flat or hierarchical. Table 2.3 compares the SDN controller models in terms of consistency, scalability, privacy resilience and failure.

Table 2.3 : Comparisons between SDN Controller Models

Parameters	**Centralised**	**Flat Distributed**	**Hierarchical Distributed**	**Narrative**
Consistency	Yes	No	Yes	It is apparently easy to maintain the network consistency in centralised and hierarchical distributed models because only the master controller has the global view whereas in flat distributed model all the controllers have the global state, this can cause problems if the controllers end up building different global states.
Scalability	No	Yes	Yes	Centralised model does not scale well for large networks. Both the flat and hierarchical distributed models are scalable, however, the flat distributed provides the robust scalability since as the network grows the network load can be shared fairly amongst the controllers.

Privacy	Yes	No	Yes	In a centralised model the single controller is the only element that has the network knowledge. Similarly, in hierarchical distributed models the top layer controllers are the only ones with the overall network knowledge, the lower level controllers only have the knowledge of their controlled domains. Hence both centralised and hierarchical models provide an elementary network abstraction. In flat distributed models controllers share the network knowledge amongst each other and makes the network abstraction complex.
Resilience	No	Yes	No	In hierarchical distributed and centralised model, maintaining normal network operations during adversities is impossible since only the master controller has the overall control and the global view of the network. The flat distributed model on the other hand has a strong failover mechanisms.
Failure	No	Yes	No	Both the centralised and the hierarchical distributed models experience the single point of failure challenge, whereas in a flat distributed model it becomes easy to maintain the network operations because any controller can take over from the failed controller

These distinct SDN controller models are discussed in the following subsections.

2.4.2.1 CENTRALISED SDN CONTROLLER ARCHITECTURES

Centralised SDN controller architectures provide single point and better control over the consistency of the network state. However, the controller has to update OpenFlow switches more often than the traditional routers, this may impose higher overload since the updating has to be done from a single point controller and also incurs considerable flexibility, scalability and robustness limitations for large scale networks. Moreover, physically centralised controllers suffer a single point of failure, meaning if the controller fails the whole network will not work. Even with the limitations of the centralised controller models a substantial number of researches employed the single controller model but with enhancements. Figure 2.3 depicts the centralised controller model whereby a single controller has the dataStore for the global network view and manages the operations of the entire network.

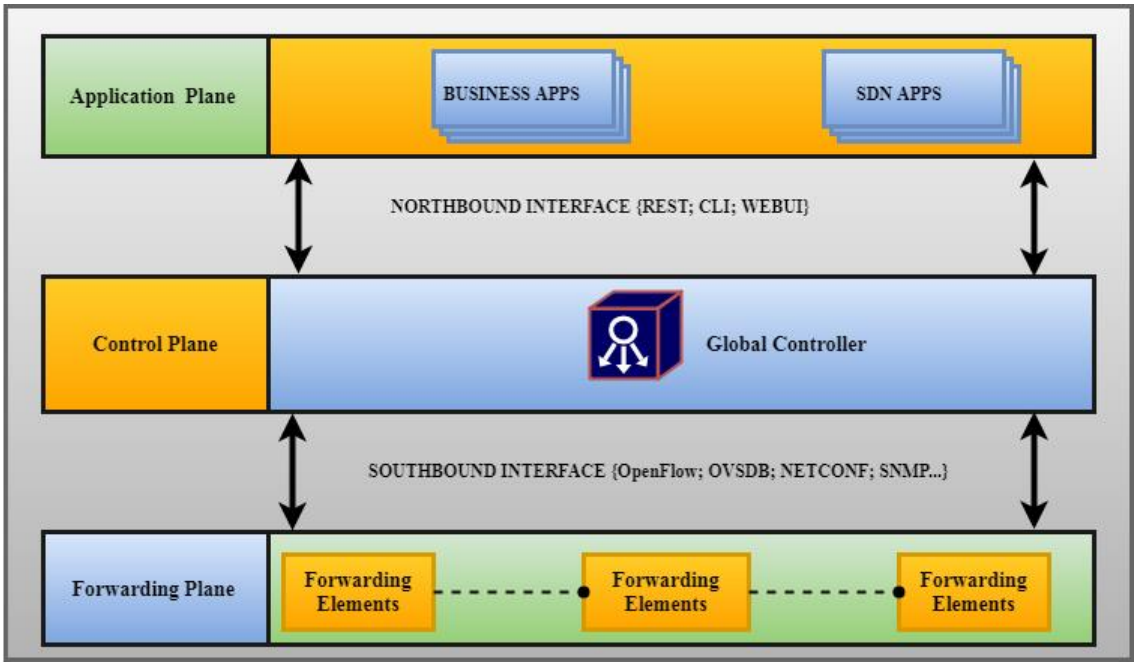

Figure 2.3: Centralised Controller Architecture.

2.4.2.2 DISTRIBUTED SDN CONTROLLER ARCHITECTURES

Distributed SDN controller architectures were designed to help mitigate the scalability and single failure point challenges experienced in Centralised SDN Controller architectures. These distributed architectures were designed with the ability to be robust and more responsive in handling local and global event in an efficient manner. Distributed SDN controller architectures provide multiple point control over the operation and management of the network, the network load is shared equally amongst the controllers. Moreover, when one controller fails or crashes another controller can take over the domain that was managed by the failed controller.

Distributed SDN Controller models are classified into hierarchical and flat distributed models. Flat distributed model controllers all have the whole network' global view, whereas in hierarchical distributed model either one controller is logically centralised, meaning that only one controller has the overall control and dataStore for the global view of the whole network while other controllers are given control tasks by the main controller. Another way is more than one but not all controllers having the global view of the whole network.

A flat distributed controller model is also known as horizontal architecture since the controllers are placed horizontally in a single level, making the control plane just a single layer. Each controller has the local view of the domain it is responsible for. Since each controller has the same responsibilities and only has a partial view of the network it needs to communicate with the other controllers in order to know their network view and hence build a global view based on that information. In that regard, this model is equipped with east/westbound interfaces for communicating with other controllers so that they can notify each other directly of any changes in their network states. In order to maintain a reliable global network state each controller has to notify the other controllers of those changes so that they can update their global network view.

The controllers in this model can acquire the network state information either by polling or by publish and subscribe.

- Polling: each controller can query the other controllers periodically for new network state information even when no change has happened on the domains controlled by the other controller. Hence this method may be inefficient.
- Publish and subscribe: each controller can either publish or subscribe the network state information to/ from other SDN controllers. If a controller is a subscriber then it can subscribe for the network information from the other controllers whereas when the controller is a publisher it will notify the subscribers always when there's a change in the information they are subscribed to.

The flat distributed models have the advantage of providing resilience to failure since if one controller fails the other controllers can take over its responsibilities. However, managing the controllers proves to be a laborious task. An example of a Flat distributed model controller includes Open Network Operating System (ONOS) [75]. Figure 2.4 depicts the flat distributed architecture.

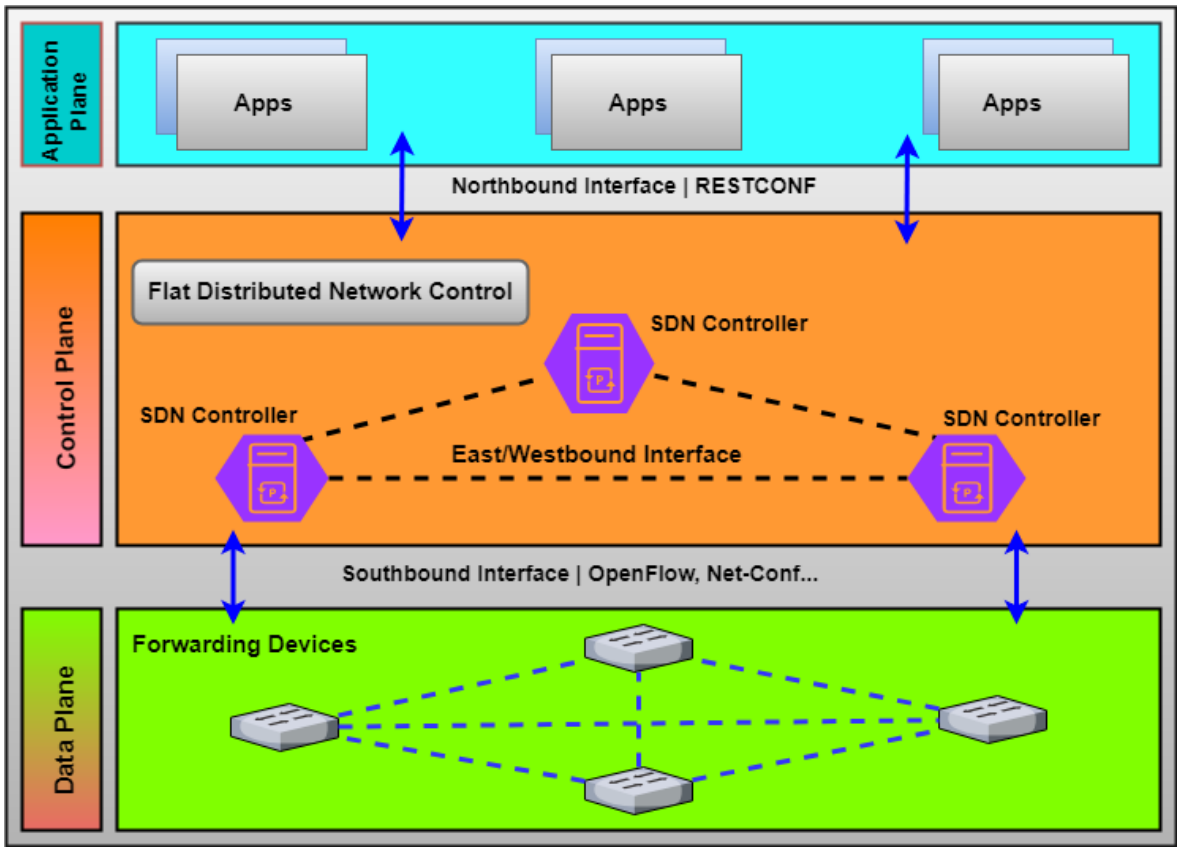

Figure 2.4 : Flat Distributed Controller Architecture.

In a hierarchical distributed controller model, the controllers are vertically placed, the hierarchical model follows a top-down control approach amongst the SDN controllers. Unlike in flat distributed model where the control layer is a single layer, the control layer in hierarchical models is multi-layered. In this model only top layer controller(s) has the global view of the whole network. The controllers in the lower layers only have a partial view of the network (i.e. their local network states).

The top layer controllers coordinates the operations of the lower level controllers in order to maintain the connectivity throughout the whole network and hence if any controller in the lower levels needs execute any inter-domain operation it has to query the network state information from the top layer controller. Due to this kind of interaction between the controllers it is safe to say that the hierarchical model is a client-server model. Where the top layer controller assumes the position of a server and the lower layer controllers that of clients. Furthermore, lower layer controllers do not have an East/west interface as in a flat distributed model. Therefore the lower level controllers do not have the ability to communicate amongst each other.

Even though this model has the advantage of improving the controller-to-switch latency and the easier way of managing the controllers it still has the same single point of failure problem that is experienced in centralised controller models. Figure 2.5 illustrates the hierarchical model whereby the control plane is partitioned into two layers. The top layer contains the root controller which has the dataStore for the global network view and the bottom layer contains local controller which only have the partial knowledge of the network.

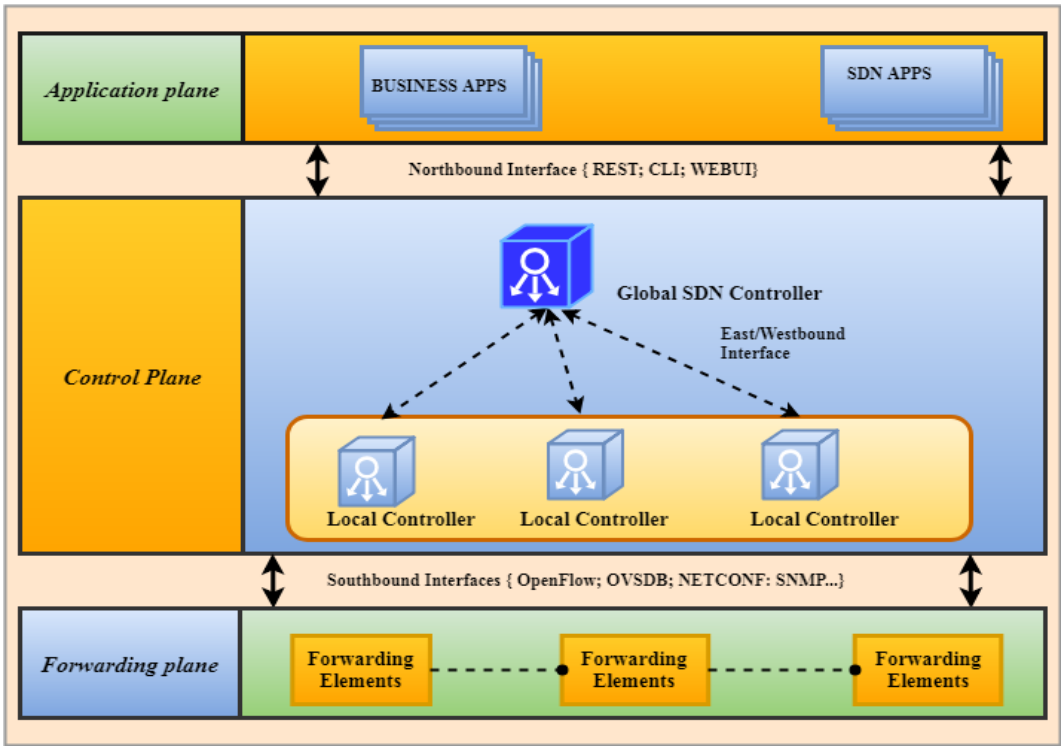

Figure 2.5 : Hierarchical Distributed Controller Architecture.

2.4.3 QoS IN SDN CONTROLLERS

QoS as a critical aspect in computer of must be well planned for, for it to serve its purpose in improving resource management and the overall network performance. OpenDayLight [76] promotes the assurance of QoS implementation in computing networks through its Network Intent Composition (NIC) by aligning QoS related functionalities to the communication processing. This functionality is achieved through the use an OpenFlow-Renderer (OF-Renderer) mechanism. QoS actions are then performed in terms of mapping QoS attributes to different data processes or flows.

The Floodlight [77] SDN controller, implements QoS policies and services within the network through the use of REST APIs as well as enabling computing modules. These modules are implemented as either tools or services within the controller. These services can be interpreted as QoS policies or actions to apply definite types of application services. The

policies are then enforced to ensure quality network services. These operations, can also be performed on the switching level of the network architecture.

2.4.4 EXISTING WORK ON SDN QOS

Architectural extensions to the Ofelia testbed were proposed by [78] in an attempt to enable it to run QoS related experiments and also offer user-friendly unified QoS support towards its experiments. Moreover, in order to extend the feature pallets of the Ofelia Testbeds, a QoS management platform that will be fully integrated with the already existing platform was proposed.

Authors in [79] proposed two distributed OpenFlow-based architectures (Hierarchically and Full distributed control planes) implementing new topology aggregation/ link summary strategies to discover network topology and obtain network state information and associated optimization formulation of end-to-end QoS provision over multi-domain SDNs. Their simulation results demonstrated that the fully distributed control plane architecture closely approaches the global optimum and nicely scales to large networks whereas the hierarchically distributed control plane scales well to networks with inter-domain distance < 2000Km.

In order to diminish the impact of interference in enterprise WLAN authors in [80] leveraged the concept of SDN and OpenFlow to reorganize the enterprise WLAN architecture. They proposed an OpenFlow based architecture where in-depth down link packet scheduling can be performed by inserting suitable instructions in corresponding Access Points (APs). Hence based on the proposed architecture, a downlink packet scheduling algorithm was proposed to obtain high Packet Reception Rate (PRR) in order to enhance the efficiency of Distributed Coordination Function (DCF). They demonstrated the capability of the proposed algorithm by simulation and their results showed that the network throughput is increased significantly and the retransmission rate is minimized. Furthermore the advantage to their solution is that it does not require any modifications to already existing clients.

In the effort to introduce SDN to mobile networks [81] proposed mobile extension of SDN (meSDN) framework, it was used to implement a Wireless Local Area Network (WLAN) virtualization service that adequately ensures that airtime shares are allocated to network slices using pseudo TDMA (pTDMA), a time division multiple access like airtime scheduling. The results from the prototype they implemented on android phones demonstrate that the framework authorizes WLAN virtualization application-aware QoS and improve energy efficiency. However, a deviation from the intended throughput ratio was experienced, this was due to increased aggregate throughput on some of the schedules.

Work such as in [82] a solution based on OpenFlow technology was proposed. HiQoS applies multipath routing and queuing mechanisms to utilise the available capacity in the network. Experimental results show delay reduction and increased throughput. Furthermore, HiQoS recovers from failure very quickly.

[83] Proposed QOF, an OpenFlow based QoS framework which provides QoS guarantee based on network measurement and QoS-based routing strategy. According to their results; QOF can find a route that is suitable to meet the QoS requirements of application in terms of network status information, and can increase resource utilization.

In [84] authors introduced OpenQoS, a novel controller for video streaming with end-to-end QoS support. They proposed dynamic routing to fulfil the required QoS. Their experimental results showed that OpenQoS outperforms HTTP-based-multi-bitrate streaming under congested network traffic. Open QoS does not depend on transport protocols, it guarantees a perfectly consistent video delivery with little or no artefacts experienced by the end user regardless of which transport protocol was used. Moreover, OpenQoS reduces the disadvantageous effects on other flow types.

Scalability and flexibility pose as a challenge to OpenFlow so the authors in [85] proposed OpenQFlow, a novel SDN-based architecture that is a scalable and flexible modified version of OpenFlow. OpenQFlow provides SDN-based QOS enhancements and has a flow state table for fine-grained flow tracking. They decoupled classification from tracking by

separating forwarding rules table into three tables: flow state table, forwarding rules table and QoS rule table. They also developed a two tier flow-base architecture that is derived from their complete version of ETA (CETA) based packet scheduling algorithm which offers performance guarantee and fairness on both granularity levels of micro and aggregate flow at the same time. They implemented the OpenQFlow prototype on an off-the-shelf microTCA and the performance evaluation results reveal that the high performance OpenQFlow data plane implementation is flexible on a user friendly programming environment.

Some researchers recommended a merge between SDN and NFV. Network Virtualization promises a wide range of benefits which include effective resource allocation, allowing operators to inspect their respective networks before making any adjustments and equipment sharing- allows competing customers to share resources in a remote and controlled manner. Moreover network virtualization promises to provide a secure and realistic environment to deploy and evaluate experimental protocols in production networks [86]. Some researchers have ventured into merging SDN and network virtualization and some of the work is discussed below.

Authors in [87] proposed an SDN architecture that uses OpenFlow protocol V1.0 and network virtualization to develop and test various QoS routing algorithms, an emulation tool was used to perform experiments, the results for RTP/UDP streaming service with regards to PNSR, SSIM and MOS show that the SDN controller guarantees QoS for a high priority real streaming flow as compared with typical best effort engines.

Authors in [88] presented FlowVisor: a network virtualization layer built and deployed in the Stanford University production network. They also built a research platform using switch-level virtualization, their platform allows numerous experiments to run alongside production traffic while still providing isolation and hardware forwarding speed. According to the authors, FlowVisor only virtualizes switch forwarding logic, the traffic's flow space and associated hardware resources. However it does not virtualize other network resources that could be virtualized such as device configuration, links and address space.
Yang et al. [89] proposed OpenRAN - a software defined RAN architecture, as an effort to meet QoS requirements for SDN deployments in C-RAN using WSRP, SDN controller and CCRP. Their approach claimed to have achieved some extant of virtualization and programmability. Authors in [90] proposed a strategy called elastic tree model has been reported to have significantly reduced power usage as well as improve handling of traffic load without compromising QoS requirements.

2.4.5 SDN OPPORTUNITIES IN WSNS

SDN is regarded as a potential mechanism to improve guaranteed QoS in WSNs, due to its nature of programmable control. Software oriented application interfaces could be developed and implemented to offer better QoS support since the dynamic nature of SDN processing is to allow and support flexible network experience. Either or with a combination of other wireless networking technologies could offer a great platform for improving the commonly used QoS tools for WSNs application systems, thereby improving network experience (in customer perspective) as well as the overall network performance. It is evident that most network design strategies, take into consideration QoS support and guarantee as an integral part of networking. A good network design should not leave out QoS requirements unaccounted for, as this would lead to poor network operations.

It is possible that, through centralization of the control plane; more advanced routing algorithms could be developed, thereby improving resource and network management in WSNs. Instances such as writing controller applications that could be used to periodically collect information on sensor nodes components such as; battery levels, current memory usage, etc. so as to maintain of implement energy-wise routing strategies. If these routing strategies are fully implemented especially in critical aspects of the network, some level of improvement in terms of the overall network performance will be achieved. In a SDWSN architecture, compute intensive tasks such as; function virtualization and service automation, provisioning of network QoS, traffic re-routing, etc. are directly executed or run on the central controller since it is a fully resourced component of the system, thereby not causing the underlying network devices to use more compute power. Therefore this network aspect, will also improves the network lifetime by preventing rapid energy depletion of sensor nodes.

Other factors that affect efficient wireless network QoS are; network topology and deployment, sensor connectivity and communication, as well as general constraints enforced on the network. SDWSN has the potential to improve these aspects by introducing some

level of state and function processing using simple Application Programming Interfaces (APIs).

2.5 SDWSN

SDWSN refers to a networking paradigm integrates SDN into WSNs thereby decoupling the control mechanism of the sensor nodes from the forwarding plane. In essence SDWSN is a networking computing model that uses SDN techniques in WSNs applications systems as a means to simplify and enhance WSNs' systems operations [91]. In a SDWSN architecture, compute intensive tasks such as; function virtualization and service automation, provisioning of network QoS, traffic re-routing, etc. are directly executed or run on the controller since it is a fully resourced component of the system, thereby not causing the underlying network devices to use more compute power. Therefore this network aspect, will prevent rapid energy depletion of sensor nodes and prolong the network lifetime.

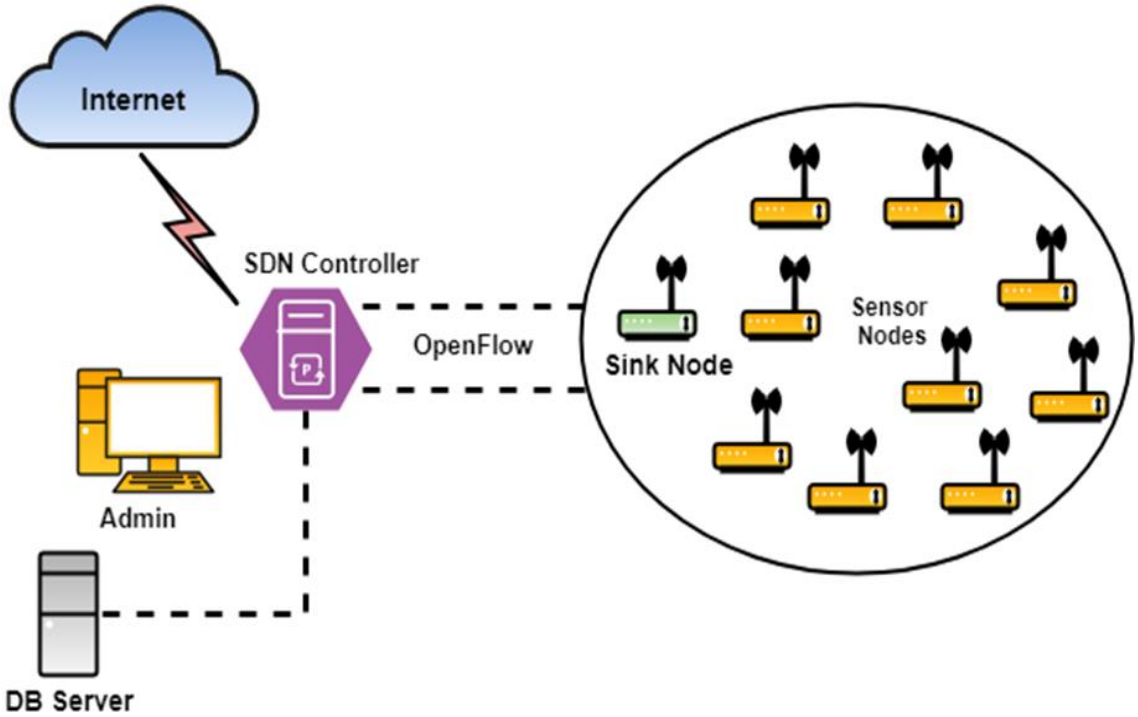

Figure 2.6: An Overview of SDWSN Architecture

In Figure 2.6, the overview of a SDWSN network is illustrated. The sensor nodes will be carrying out forwarding functions only and the network intelligence will be based on the SDN controller. The controller has the distributed data storage that enables it to store the global view of the entire network. Moreover, the compute intensive tasks such as traffic management, QoS management, resource allocation, path computations and others will be done by the controller.

A perfect example of how SDN is envisioned to improve WSNs is exemplified in [92] where the authors proposed smartly management of WSNs through the applications of SDN, they proposed that the controller be situated at the base station. The controller makes all the routing decisions and has the global view of the entire network, it collects the state information by receiving monitoring messages and the sensor node serve as forwarding devices. Moreover, the controller calculates the routes by using the location of all the sensors. Their proposal envisioned that some of the WSN inherent issues can be solved by applying SDN to smartly manage WSNs.

Basically QoS in SDWSN as depicted in Figure 2.7, is performed in three steps:

- Firstly a flow is regulated to ensure that it does not violate traffic contract and then placed in a queue.
- Then, the QoS requirements of the flow are conveyed to the controller. For the controller to provide the QoS requirements, it has to examine the network traffic and QoS parameters and check whether the available resources are adequate to guarantee the requested QoS.
- Then the controller conveys QoS guarantees by providing a path that will meet the QoS requirements to the network node flow table.
- And finally flows are monitored by traffic policing functions evoked by the controller to ensure that they do not violate the agreed traffic contract.

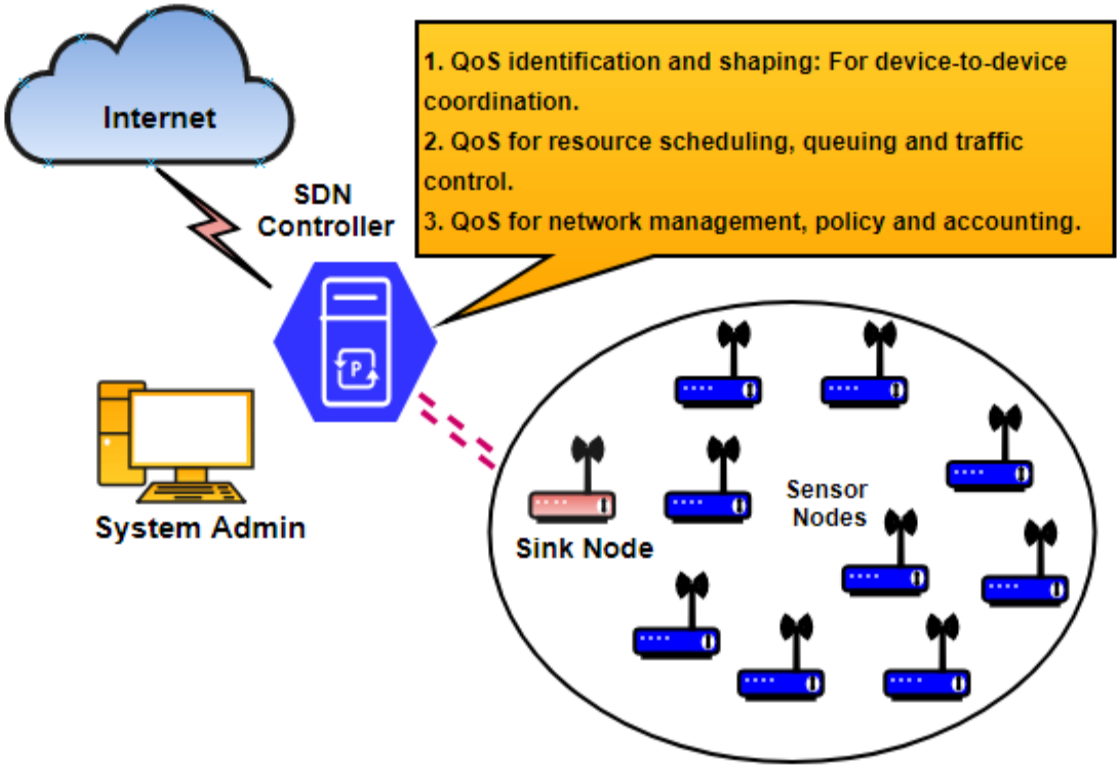

Figure 2.7: Basic QoS Operations in SDWSN

One of the first attempts at combing the SDN and WSN technologies is found in [93], they proposed a SDWSN architecture and addressed the main technical challenges for its core component i.e. Sensor OpenFlow. Their intentions were to transform classical WSNs into flexible, versatile and easily manageable networks but since OpenFlow was specifically designed as a wired protocol, they experienced great challenges in designing Sensor OpenFlow and this also presented significant disparities. However, they proposed solutions to these challenges experienced when designing Sensor OpenFlow.

Researchers in [94] developed a TinySDN based SDWSN framework called IT-SDN, with emphasis on the ensuring efficient controller communication with network devices. IT-SDN framework consist of three components: Neighbour discovery, controller discovery and southbound protocols. Neighbour discovery protocols obtains and maintains the nodes neighbour information. Controller discovery protocol defines the node that serves as the next hop to reach the controller. Southbound protocol serves as the communication interfaces

between forwarding nodes and the controller. The most distinguishing feature of IT-SDN is the clearly defined boundaries between these protocols and the OS independence specification.

In [95], Authors presented a concept of Software Defined Sensor Networks and introduced numerous key technologies for making possible the attainment of this concept. They also introduced their ongoing project DASN project which is envisioned to build a wide-area sensor network that simplifies users' complex sensing demands in japan. They also presented their initial achievement: successful design and implementation of a software-defined sensor node prototype, whose architecture comprises a core board with a MCU, an FPGA unit, and an RF module, a sensor board, and a 32-bit CPU card, connected to each other.

SDN also introduces evolvability, simplicity and cost effectiveness, in [96] they designed and developed a low-cost, low-power software defined acoustic model, the node to node data transmission and self-organizing networking performances are represented by the underwater experiment results that indicated that the modem has the capability of functioning in different underwater sensor network applications.

TinySDN - a hardware independent TinyOS [97] based architecture that permits implementation of multiple controllers within a SDWSN is presented in [98]. Moreover, a protocol was proposed to facilitate the communication between the controller and sensor nodes. TinySDN is said to have achieved the SDN-provided communication flexibility, while the memory overhead it introduced is negligible. The experimental results suggested that TinySDN does not introduce considerable delay in packet transmission except on first generated packets.

In [99] Spotled architecture with distributed hierarchical controllers was proposed. This architecture only allowed each local controller to respond to the sensor nodes in its domain of control and the master (Global) controller administers the operation of the entire network. They employed cluster heads to serve as local controllers, which were managed by the global controller. These local controllers shared information on their network state with the

controller, then then controller used the information to build the global network view. Their experimentation results suggested that the proposed architecture has the ability to reduce the control traffic in SDWSNs and hence reduce the data flow installation time and the initial communication latency.

In [100], the authors proposed TinySDM, a Software Defined Measurement Architecture for WSNs. TinySDM has a number of abilities: 1) offers support for performing varied measurement tasks 2) permits tailor-made specifications on numerous measurement tasks and 3) lets new measurement tasks to be efficiently deployed. TinySDM was implemented on the TinyOS platform and they used testbed containing 60 nodes to assess its performance. The implemented task were: end-to-end delay, PAD, Domo and Sympathy. Their results suggest that TinySDM is adaptable, efficacious and effortlessly programmable.

In [101], wireless SDN based flow-based programmable nodes were designed and implemented in WSN network. These nodes used OpenFlow as their communication protocol within the Software-Defined Sensor Network (SDSN) platform. According to their experimental results, it was reported that their platform was able to achieve computing functionalities such as; network topology discovery, sensor data acquisition and processing, with efficient flow forwarding. They also reported their platform to have achieved a lower packet loss rate due to its forwarding capability.

2.6 QOS IN SDWSN

Over the years researchers have worked extensively attempted solving the QoS limitations and provide the desired QoS requirements for WSNs, they came up with various methodologies ranging from QoS aware routing protocols, layered and cross layered QoS support approaches and middleware layer QoS support. Nevertheless, providing the overall QoS support with both layered and cross layer approach is a challenge since some techniques meet a certain level of QoS but at the expense of the energy. The layered approach meet QoS requirements on individual layers. Cross layer approach seems to have a lot of potential but

most of the proposed cross layer approaches are not implemented in real networks and therefore lack real world experimental proof.

In WSNs, the importance of guaranteed and enhanced QoS which should not be compromised is emphasized in accordance with the network dynamicity and performance. The shift towards SDN based WSNs stands to be an efficient approach for optimized QoS, since programmable applications could be developed to implement and handle QoS requirements pertaining to sensor nodes within the controller. Wang et al. [102] highlights that, QoS requirements are taken as inputs by the local manager of their proposed system to allocate a spectrum band which is ruled by the global manager for different flows. They further indicate that since a local manager is programmable, QoS policies could be updated by the system admins for different network functions.

SDN is regarded as a potential mechanism to improve guaranteed QoS in WSNs, due to its nature of programmable control. Software oriented application interfaces could be developed and implemented to offer better QoS support since the dynamic nature of SDN processing is to allow and support flexible network experience. Either or with a combination of other wireless networking technologies could offer a great platform for improving the commonly used QoS tools for WSNs application systems, thereby improving network experience (in customer perspective) as well as the overall network performance. It is evident that most network design strategies, take into consideration QoS support and guarantee as an integral part of networking. A good network design should not leave out QoS requirements unaccounted for, as this would lead to poor network operations.

The SDN architecture exposes rich possibilities to transform resource constrained networks such as WSNs in that it possess the capability to partition network operations depending on which services needs to run or be executed where in the network. QoS in WSNs has no to little research achievements or outputs that have transformed to actual practices in application networks. However, SDN presents limitless possibilities to change this, since developers could take the advantages brought about by the SDN programmability. Network services such as QoS guarantee and automation could immensely benefit from the SDN

paradigm since these services may be software controlled and automated according to the network requirements. In particular QoS requirements could be implemented using reprogrammable strategies as a means to enable on demand system responses.

Efficient QoS support could even transform WSNs in terms of improving them by reducing their limitations especially in processing. WSNs could benefit from SDN optimized QoS activities through the use of programmable strategies to initiate and guarantee QoS activities within the network. QoS activities such as service scheduling and bandwidth allocation for critical data or applications, could be automated by the network infrastructure accordingly. Software oriented strategies could also be used for the resource prioritization of network intensive tasks or services depending on the technicalities required to successfully execute them.

SDN bears the capacity to evolve WSNs in terms of how the perform computation by introducing simple programmable abstractions to their network operations. A novel SDN based framework called SensorSDN that was envisioned to meet the specific requirements of diverse WSNs was proposed in [103]. This architecture was also envisioned to operate in various IoT systems. Firstly, they proposed a new control plane services to promote automatic topology discovery, node mobility, sensor virtualisation as well as management of network policies. Additionally, to meet the requirements of various sensor network packets an SDN based tailor-made flow tables were proposed for existing Low-Rate Wireless Personal Area Networks (LR-WPAN) technologies. Finally, a programmable MAC Layer was proposed to support fine-grained flow processing.

An SDWSN architecture was proposed in [104], where a network of 24 nodes and a single base station was designed. The authors highlighted how in network processing and performance predictions were performed. The results of their experiments demonstrated significant improvements with regards to throughput and received packets as compared to the traditional WSN. However, their proposed system has shown to be power inefficient since the power consumption increased by 3 % as compared to the traditional WSN.

2.6.1 ENERGY MANAGEMENT

The network lifetime as well as energy policies implemented in WSNs has the aspect of being treated as QoS guarantees, in that they reflect how and what methods of deployment were taken in to account. Therefore, the essence of QoS guarantee in computing networks is of critical factors especially in application systems that needs to maintain quick and accurate data processing. As part of the strategy for SDWSN, the separation of network control from the device wireless data forwarding, could effectively benefit the process of improving the network lifetime and to some level, slower the high rate of energy consumption in network critical devices, thereby allowing efficient operation of the entire network.

The work in [105] a general SDWN architecture where cluster-based sensor nodes with forwarding and switching capabilities only are managed by an SDN controller that is situated at the base station was proposed, in an effort to address the energy consumption problem. OpenFlow was used as the communication protocol between the controller and the switching elements. The results of the experimentations demonstrated that the proposed architecture minimized the energy consumption as predicted. The realization of efficient WSNs technologies is still a heavily engaged matter in that these systems are extremely limited in terms of performance. Recent work sees SDN as a positive paradigm to revolutionize WSNs application systems for greater performance.

Authors in [106] proposed a multi-tasking energy efficient routing algorithm for SDWSNs. In order to make the network convenient, their algorithm chooses control nodes to allocate various tasks. This control nodes selection is conceived as a non-deterministic polynomial-time (NP)-hard problem considering the remaining energy of the nodes and the communication distance. As a solution to the NP-hard problem, they used a non-linear particle swarm optimization to create a cluster structure so as to reduce the communication distance. Their simulations suggest that their algorithm is energy efficient and thus prolongs the network lifetime.

To overcome the flexibility issues experienced in WSNs, the work in [107] proposed an SDWSN based energy efficient routing algorithm which used the information collected from the sensor nodes to establish the distance queue. The benefit of this routing algorithm is that the sensor nodes only transmit data to the closest neighbouring nodes which eliminates the transmission overhead and thus leads to energy savings. Their simulation results as compared to leach revealed that the proposed routing algorithm increases the traffic of the entire network, consumes energy uniformly and therefore, prolonged the network lifetime.

In [108] an energy efficient algorithm based on energy shell and dynamically fine-tuning communication radius was proposed. The higher priority to relay data is assigned to nodes in higher energy shells but preference is given to nodes with the highest relative received signal strength. The experimentation results suggested that the proposed algorithm outperformed AODV and DSR in terms of energy efficiency and networking success rate.

An SDN based sleep scheduling algorithm SDN-ECCKN that reduces the over-all transmission time of a network was proposed in [109]. SDN-ECCKN selects the most appropriate node to go to sleep by taking into consideration the residual energy of the sensor and the number of alive neighbors, by doing so it ensures that the connectivity of the network is maintained and the network lifetime is maximized. All the computations are performed by the controller execute all the computations instead of the sensor nodes and hence, there is no sensor-to-sensor transmissions in their algorithm. Their simulations demonstrated that SDN-ECCKN achieved noteworthy enhancements in terms of network lifetime, number of functional nodes and number of isolated nodes as compared to ECCKN.

Authors in [110] investigated how to design an energy efficient reprogramming strategy with guaranteed quality-of-sensing for a sensing task. They proposed an Integer Linear Programming (ILP) based algorithm which has a low computation complexity to address the following issues: 1) reprogramming sensor selection and 2) program distribution routing. They proposed an efficient ILP based heuristic algorithm which has a low computation complexity to address the high computational complexity of solving ILP. The results demonstrated that the proposed algorithm performs close to optimal. Furthermore the

authors validated the efficiency of the proposed algorithm by comparing it with a two-phase algorithm that considers the 2 issues mentioned above separately and the IPL based heuristic algorithm proved to have significantly outperformed the two-phase algorithm.

An energy efficient SDWSN with wireless power transfer was introduced by [111], where they proposed an optimal placement of a small number of energy transmitters and energy-efficient scheduling of these transmitters. They investigated a trade-off between maximum energy charged in the network and fair distribution of energy for assigning the energy transmitter. A binary integer linear programming problem was formulated while satisfying the limitations on minimum energy charged by each sensor. For given energy charging tasks, they proposed an energy-efficient scheduling scheme for energy transmitters, where the optimal solution was found with the help of branch and bound algorithm. Finally the authors highlighted that, compared to traditional WSNs with wireless power transfer; energy consumption can be minimized considerably by scheduling energy transmitters in SDWSNs.

In an effort to achieve energy efficiency, the authors in [112] proposed software-defined clustered sensor networks (SDCSN) - a cluster-based SDWSN architecture with multiple base stations which serve as both the Controller hosts and cluster heads. In this architecture, the controller shares information with other SDN domains, moreover the controller manages the network and security of its controlled domain. This approach was also extended to provide an interconnection of multiple SDN domains through border controllers. Hence, leading to a compact security model where sensor traffic flow can be collaboratively controlled.

An SDN based energy optimisation architecture for multitasking WSNs was proposed in [113]. The authors pointed out that the optimisation can be achieved through controlled sensor scheduling and management with quality of sensing for all sensing tasks. The minimum energy sensor activation problem was formulated as a Mixed-Integer with Quadratic constraints Programming (MIQP) problem by coupling sensor activation, task mapping and sensor scheduling. MIQP is then reformulated into a low computation complexity formulation through linearization in the form of Mixed-ILP (MILP).

Additionally, an online algorithm was developed to address operational dynamics such as sensor admittance and departure during Software Defined Sensor Network (SDNS) runtime. The results revealed that the proposed online algorithm achieved approximately the same energy optimization as a global optimisation with lower control overhead and rescheduling time.

In IWSNs, data transmission delay compromises the reliability of the system and also causes unnecessary energy consumption. Sensor nodes deployed in in hot spots areas tend to utilise large amounts of energy due to their heavily engaged operations. As a result, efficient energy minimization strategies must be implemented to prolong the network lifetime. Modified SEECH (MSEECH) - a mechanism for enhancing energy efficiency in Industrial WSNs (IWSNs) using SDN AND NFV was proposed in [114]. They used the global view of the network and the central control to monitor IWSNs. The topology of IWSNs and node modes were controlled by exploiting the programmability of SDN and the instant deployment capabilities of NFV. Furthermore, they proposed advanced algorithms for the controller. To prove its efficiency they compared MSEECH with LEACH and SEECH. Their results suggested that, MSEECH was more energy efficient than its two comparatives in WSNs with high data aggregation compression ratio and slightly less efficient than SEECH in WSNs with low data aggregation compression ratio.

In an attempt to achieve energy efficiency and smart management in WSN, a generic base station architecture for SDWSNs was proposed in [115]. Moreover, they proposed an SDN based clustered architecture for WSNs where the controller is situated at the base station for generating the routing table for the cluster-head, controlling the data sensing of the sensor nodes, and providing more feasible and flexible and convenient management function for the users. Finally, they validated the efficiency of their proposed architecture by comparing its performance and energy consumption with LEACH Protocol. The authors explained the system overview but did not explain their experimentations. However, they discussed and compared the proposed architecture with LEACH protocol and came to the conclusion that the proposed architecture is outperformed LEACH with regards to performance and energy consumption.

An effort to improve traffic control in SDWSN was proposed in [116]. In this work, a sleep-scheduling scheme with overhead-based low control flow entry was proposed to deal with this limitation. This strategy is applied in a flow-table by implementing specific controller actions on flow-entries to achieve this. The main purpose for this was to find ways to deal with the dynamic nature of operation in WSN application systems, spanning their node failures, data transmission and energy management. Their simulations were performed on NetTopo simulator [117], which is provides functionalities to visualize network architecture and perform algorithms analysis. They have reported that, their proposed scheme, produced better results in terms of executing controller messages compare to software defined energy-consumption-based connected k-neighbourhood (SD ECCKN). They also proclaimed that, their scheme produces the highest parameter of k for sleep-scheduling operations in ECCKN.

In [18], the realization of efficient WSNs technologies is still a heavily engaged matter in that these systems are extremely limited in terms of performance. Recent work sees SDN as a positive paradigm to revolutionize WSNs application systems for greater performance, promoting energy efficiency and hence extending the lifetime of the network.

In the work such as [119], Reinforced Learning (RL) based energy efficient and flexible SDWSN prototype for monitoring systems was proposed. In this prototype, the controller centrally manage the complex computation while the forwarding elements are responsible for computations that are low in complexity. The results of their experimentation suggested that the RL-based prototype has the ability to enhance the self-learning capabilities of environment monitoring WSNs with QoS and significantly increase energy efficiency by effectively hindering unnecessary loads transmissions, minimizing the quantity of cross-plane communications and improving the load balancing in SDWSN.

Wang *et al.* [120], proposed and implemented an SDN-based wireless routing protocol for a Multihop wireless network in an effort to improve wireless traffic routing and sensor network lifetime. Their proposed protocol was implemented in OPNET. The efficiency of this protocol was tested for various network parameters and for certain network loads. It is indicated on their arguments, as informed by their experiments that, their SDN based routing

protocol achieved better results in terms of prolonging network lifetime compared to OLSR, since it uses less energy. They also reported that, their experiments indicated improvements in achieving shortest paths as well as achieved disjoint multipath for network nodes routing, thereby, increasing network lifetime as compared to OLSR and AODV for a certain value of traffic load.

2.6.2 RESOURCE ALLOCATION AND UTILISATION

In any network architecture, traffic control and resource management are key. However, to achieve greater network performances, critical aspects of the network architecture such as; traffic routing, resource sharing and reservations, packet queuing, application or services assignments, transmission acknowledgements, etc. must be well facilitated and managed. To realize this requires the best networking strategies that support QoS implementation towards efficient resources and traffic management. Therefore, implementing QoS requirements in network computing application systems, must not be compromised at all cost.

However, implementing QoS requirements in any system must not hinder or terminate other network processes or distort various traffic flows that are already administered as network state. To realize this, technical operations must be developed and applied on the network, to ensure corporative performance of all network services. This analogy makes networking a fundamentally sensitive and complex aspect of technology. SDN provides meaningful technicalities to orchestrate the network in separated planes. This can allow efficient understanding on the entire network and provides the opportunity to manage the network in isolate operations. QoS assurance and support can greatly benefit from this strategic orchestration through service abstraction. Hence, SDN provides modularity operations as a way to introduce and advance network flexibility [121] as well as promote innovation.

Research in [122] proposed a Flow Splitting Optimization (FSO) algorithm as an effort to minimize sensor traffic load. They defined this problem as the Traffic Load Minimization (TLM) problem. The main argument, as part of their objective, was that the transmission of effective packets with optimal path selection in SDWSN is a challenge. Hence, their work

conducted a similarity concept to establish different sensor nodes packets. Their approach uses a selection of relay sensor nodes, which have some resource capacity, as well as the transmission of specific splitting flows. They reported to have solved the TLM problem by applying a Levenberg-Marquardt algorithm. Their proposed FSO algorithm was implemented in the NS-2 simulator, with numerous experiments to justify the effectiveness of the proposed algorithm. They have reported that, their FSO algorithm improved packet delivery ration with optimal packet transmission. We also make an observation based on the results from their work that, there is an indication of some great achievements.

Load balance centralised control routing protocol (LRCC) mechanism to construct the forwarding routes was proposed in [123]. The proposed mechanism is divided into three phases i.e. neighbour maintenance where every element in the network is required to discover its neighbours and preserve the neighbour list using a cushioning algorithm, topology collection where every sensor node sends their network status report to the controller; forwarding rules generation and distribution where a load balancing Floyd algorithm is used by the controller to compute the routing rules then the routing rules are distributed to all the sensor nodes. Their proposed mechanism implemented on their SDWSN system illustrated significant improvements with regards to bandwidth and packet loss. This was achieved due to their proposed mechanism's ability to avoid using unstable links as part of the forwarding routes. However, the authors highlighted the scalability issues of their system.

There is a general consensus that resource management (allocation and utilisation) in WSNs has significant effect on QoS support [124]. In that regard, numerous efforts have been made in the past to deal with sensor networks resource management since these types of networks have serious competence issues. In [125], authors exploited SDN programmability to achieve simplified resource management and they also indicated resource access and manipulation improvements.

An SDN based model for resource management in Heterogeneous WSNs (HWSNs) that was intended to support the selection of sensor allocation strategies in an on-demand manner was

proposed in [126]. Their proposed model takes the different applications requirements and the sensors' characteristics into consideration when choosing an appropriate allocation strategy. The sensor allocation strategies can differ according the goals set to be achieved. The authors conducted preliminary tests to validate the proposed model to show that the proposed model can handle the dynamic selection of allocation strategies.

Researchers in [127] proposed a heuristic algorithm to solve the task allocation problem experienced in shared SDWSNs. Their algorithm exploited of routing information and sensor battery measurements to improve energy efficiency and resource utilization among sensor networks. The algorithm is said to have attempted to avoid transmission bottlenecks by ensuring that there are no load imbalance over relay nodes. They compared their proposed algorithm with the traditional WSN architecture that tends to reduce energy consumption during task allocation. Their results suggested that their algorithm achieved promising results with respect to the network lifetime and application acceptance ratio. It also improved QoS for accepted applications.

One of the critical issues in WSNs, is how limited these technologies are in terms of energy. This is a major challenge as it directly impacts on the overall performance of the system. Authors in [128] proposed a centralized algorithm with a focus in resource allocation to reduce energy consumption in a SDWSN system. Their system uses a Semi-Definite Programming (SDP) relaxation technique as a programming model to solve the no-convex problem which negatively affect efficient resource allocation.

Based on their aim to reduce energy utilization, they formulated the energy consumption issue as an optimization problem. In their work, a centralized adaptive bandwidth and power allocation (CABPA) algorithm was proposed in an effort to lower energy consumption by network nodes. They claim to have achieved this by, allocating energy and bandwidth efficiently. To validate their proposed approach, they said to have considered centralized adaptive bandwidth allocation (CABA) and centralized adaptive power allocation (CAPA) together with their distributed adaptive bandwidth and power allocation (DABPA).

Their experimental results, their proposed Centralized Adaptive Bandwidth and Power Allocation (CABPA) indicated a significant energy utilization reduction and efficient bandwidth utilization, whereas their Distributed Adaptive Bandwidth and Power Allocation (DABPA) also indicated better bandwidth utilization compared to the Centralized Adaptive Bandwidth Allocation (CAPA) algorithm, which in their report claim it used more power. However, they have also stated the trade-off experience in facilitating efficient power allocation and bandwidth utilization.

A Cloud-Assisted SDWSN (CSDWSN) where SDWSN capabilities were optimized using cloud servers was proposed in [129]. The focus of this work was to merge development amongst CSDWSN service providers as a means to provide resource and revenue sharing approach with some coalition.

SDWSN was also used in Smart Grid WSNs [130] where all sensing nodes are managed by a central, fixed position OpenFlow controller which can openly select routing protocol according to the network topology changes. Compared to already existing WSN protocols their solution is said to reduce network complexity, energy consumption in sensor nodes and hence prolongs the network lifetime.

In [131], an SDN based load balancing scheme was proposed for 5th generation cellular architecture in an effort to solve the high resource consumption issue experience in these systems. Their approach was focused on implementing this technique without any means of affecting the QoS as perceived by the network user. As much as it is important to deal with factors that leads to high resource constraints in WSNs, QoS requirements and implementation strategies (especially for customer bound or production network systems) should not be disregarded or simple treated as this could lead to several dissatisfaction. High QoS guarantee could be achieved in SDWSN, since the SDN paradigm proposes responsiveness, flexibility, scalability, robustness, etc. in WSNs systems.

Since WSNs are susceptible to faulty links and link failures, a lightweight flow management model was proposed in [132] to address the issue. This model was envisioned to reconfigure flow entries in the flow table once a link failure or fault occurs. The authors highlighted that the objective of the proposed model is to minimise the circulation of control messages within the network.

CoBWSN-A constrained application protocol (CoAP)-based SDWSN control plane was proposed in [133].the authors comprehensively specified the communication infrastructure, the control plane protocol, and the basic functions in controllers. They further extensively described and demonstrated the implementation of CoBWSN in Contiki OS using Cooja. The experimental results indicated that the overhead introduced by CoBWSN is negligible compared to the benefits that SDN flexibility brings to the network and therefore this proved its potential. However, their proof of concept lacked extensive experimentation.

2.6.3 SDN BASED ROUTING MECHANISMS

Authors in [134] proposed an SDWSN focused Situation Aware Protocol Switching Scheme (SAPS) for real-time support of application-specific requirements. Their proposed scheme consists of two phases: The decision phase- where supervised learning algorithms are employed to determine the appropriate protocol to be deployed; and the protocol deployment phase- where protocols are deployed at the sensor nodes, according to application-specific requirements. Their simulation results suggested that SAPS outperformed some of the already existing schemes with regards to delay, energy consumption and throughput in different scenarios. However, their scheme did not yield the best results with regards to Packet Delivery Ratio (PDR) and since the controller did not receive the network state in real-time it questions the real-time support that the authors aimed to achieve.

Wei et al. [135] proposed a centralised SDWSN system whereby a single controller managed many sensor nodes. They proposed load balance centralized-control routing protocol (LCCR) in order to implement centralised control. The sensor nodes share their topology

and link quality with the controller, the controller then configures and distributes the routing tables of every node based on load balance. The experimentation results demonstrated that LCCR achieved improvements in terms of bandwidth and packet loss since it avoids selecting links with higher packet loss ratio and links with unstable performance.

Anadiotis *et al.* introduced Software-Defined Wireless SEnsor (SD-WISE) networking solution which they reported to have complemented the application of SDN to WSNs with some operational enhancements compared to its competing solutions [136]. Based on their conducted experiments, they reported this solution to have shown impressive support to NFV thereby leveraged the underlying OS to achieve critical network functionalities such as geographical routing.

In [137] SDN Cluster Head (SDNCH) architecture is used to administer routing functions and security rules to the border controllers. Their implementation used ODL controller for managing and monitoring traffic flow in IoT environments. They further proposed a routing protocol that governs the routing functions of the clustered system. They also employed OpenFlow capabilities and Network Virtualisation to create SDN-controlled virtual nodes.

Researcher in [138], proposed μSDN, which they formulated as an architecture that uses SDN strategies to extend Ad hoc On-demand Distance Vector (AODV) and Link-Quality Routing Protocol (LQRP) strategies to facilitate communication between the control plane and the forwarding plane. They claim that their proposed architecture exhibit almost the same scale of energy utilization as well-known protocols, except some slight increase of signalling overhead compared to both AODV and LQRP protocols. They also advocate that, μSDN has the potential to be applied in SDN based WSNs, since it can be easily used to configure routing policies without changing the actual hardware.

Zhu et al. [139], proposed a node selection localization algorithm using SDN computing strategies. They designed this node selection localization algorithm based on Cramer-Rao Lower Bound (CRLB). Their experimental results demonstrated that, the proposed algorithm

achieved great node localization results. It was also reported that, through this proposed algorithm, significant energy consumption minimization for node localization was achieved.

The authors in [140] proposed a routing approach permits administrators to maintain the global view of each routing path and also simplifies the procedures such as troubleshooting and network provisioning by coupling link state routing with OpenFlow. Since all the routing decisions will be made by the controller the nodes would not have to negotiate amongst each other. Their proposed architecture promises to provide a fault-tolerant system design as well as the possibility of an increase in network utilization.

In WMSNs it is important to take certain QoS parameters into consideration during route selection since lost packets needs to be retransmitted hence using extra energy. Moreover, throughput can be affected by delay in data transmission. Finally, control overhead should be kept at a minimum otherwise it will cause energy wastage. To deal with such challenges, authors in [141] proposed a QoS-aware routing mechanism for OpenFlow-enabled WMSNs. The mechanism consists of a framework and routing algorithms on SDN controller. The framework is located at the application layer of SDN architecture and comprises five components: link state detection, flow classification, flow management, QoS-aware routing algorithms, and per-flow routing policy.

An SDN based algorithm was developed by Han *et al.* [142]. This algorithm was used to determine a link as well as QoS flow requirements. This application, was reported to have effectively reduces throughput as well video traffic delays. The routing algorithms are achieved in two steps. First, it finds the practical paths that fulfil QoS requirements of a flow. If no appropriate path that suits the required QoS is found, the proposed algorithms will be select the path relying on flow types. They evaluated the performance of the proposed QoS mechanism against OSPF and RIP. The Results suggest that QoS-aware routing mechanism achieved 43% throughput more than RIP for video data transmission, and minimized the delay by 30% and 54% as compared to RIP and OSPF for audio data transmission respectively. Hence, they argued that the QoS-aware routing algorithm is advantageous in satisfying QoS for different flow types.

Authors in [143] proposed a routing algorithm for a hybrid WSN architecture where a traditional WSN using MTE routing protocol is deployed together with a partial Software Defined Sensor Network (SDSN). The aim was to prove that if the controller has the global view of the network it can significantly increase the network life time of the WSN. The programmable nodes functionalities include topology discovery, sensor data acquisition and processing. OpenFlow flow table with detailed definitions of the match field, action field, next hop field, Time To Live (TTL) field for every entry. According to their results, it is reported that, the programmable data plane with flow table forwarding reduces packet loss ratio and processing latency due to the use of a local controller. Hence, they emphasized that if the state information is available the network performance can be improved and the network lifetime prolonged regardless of the network topology.

2.6.4 CONGESTION CONTROL

For large scale WSNs, congestion is the main challenge experienced. If the network was not designed to scale up with the continuous growth, then it might be congested during high traffic volumes. As a result, this then leads to extra energy consumption hence network lifetime may be shortened. Recently, researchers have proposed ideas to use SDN strategies to ensure QoS provisioning and provide congestion avoidance mechanisms. In [144], an SDN approach that exploits state information to support QoS provision in WSNs was proposed. By the time of this work, an approach to provide QoS support in SDWSN was proposed as its main contribution. This work is based on the previously proposed Software Defined Networking solution for WIreless SEnsor Networks (SDN-WISE) [145], whose aim was to simplify network management.

They simulated their network using the OPNET simulator. Sensor nodes states using the SDN-WISE mechanism are exploited to report network activities as a means to apply QoS requirements. Based on this mechanism, sensor nodes can efficiently convey stateful information in terms of congestion levels in the network to the controller. By making use of this state, the controller could assign rules for different packets drop probabilities to different

traffic flows depending on the current level of congestion at the sensor nodes along the path going from the source sensor node to the sink as a means to avoid congestion. According to their results, these sensor reports have been used to determine the efficiency of their system through applying some required levels of QoS.

SDN-WISE is a stateful SDN solution for WSN systems, which was intended for reducing the amount of information exchange between the SDN controller and the adjacent sensor nodes and also for enabling these sensor nodes to be programed as finite state machines. From that work, they reported that their system approach increased network elasticity and also provided simplified network programmability since it gave the system developers freedom to use programming languages of their choice when implementing the SDN controller. In our view this adds to be a great improvement in the aspect of WSN network programmability especially with the ability to use any high level programming language when implementing the SDN controller, since the controller build language choice is one of the aspects highlighted in this paper.

Congestion in WSNs occur when the transmitted traffic load exceeds the capacity that sensor nodes can handle. Congestion can occur due to factors such as varying transmission rate, channel contention, buffer overflow, many to one communication nature, concurrent transmission amongst others [146] [147]. Congestion leads to increased packet loss ratio, increased latency, increased delay, low throughput, wastage of node energy, increased retransmissions which lead to more energy wastage and hence deteriorates network performance and affect the reliability of the network [148] [149]. In that regard, this proves the importance of controlling congestion since it results in the degradation of QoS support.

Researchers in [150] used SDN strategies to introduce some traffic delay in WSNs to reduce packet losses, which occur during congestion scenarios. This approach exploits the delay- and loss-tolerant nature of WSNs as an opportunity to advance the introduction of this delay. To achieve this, they used SDN strategies to implement a Delay-Inducing Congestion Mitigation (DICM) algorithm that delays network-generated traffic, with a directive to reduce packet loss. They used TinyOS, which supports 6LowPAN traffic to program their

simulated TellosB [151] sensor nodes. Based on their proposed approach, it is reported that, the DICM reduced packet loss because of the introduced delay. They also claim that, their framework can be used in a wider range of networking devices due to its basic OpenFlow features.

In [152], a traffic and resource-aware model- SDN-TAP was formulated as a framework to advance wireless sensor programming to adapt flow-tables dynamically. Within the SDN-TAP framework, network formation and network operation phases are implemented to flow-path update and topology discovery. This model was developed on the focus of health WSN application system. A traffic monitoring algorithm was also implemented with a purpose to send an alarm to the controller when detecting congestion situations. This action is performed to invoke the controller to write new flow rules to those congested nodes. Based on their reported results, the SDN-TAP framework had improved network reliability and reduced packet loss ratio quite significantly.

Researchers in [153] proposed a SDN driven framework for WSN applications- *WARM*, executed on a web browser to developed and manage sensor applications. They used TinySDN to facilitate a connection between sensor nodes running on the TinyOS which uses the nesC (network embedded systems C) language. Their framework controller was implemented using Python language. Their experiments indicate that, WARM, simplifies programming, decreases overhead and is practicable. However, when compared with Terra, TinyDB and Swiss QM for memory utilization, WARM seemed to be using the large amount of RAM with minimum ROM. On the hand, apart from other best-case performances by WARM, it is unclear as to how the authors mentioned a decreased overhead whereas in their statement, WARM used the largest RAM amount when compared to Terra, TinyDB and Swiss QM for memory use. Table 2.4 discusses the pros and cons of some of the solution mentioned above.

Table 2.4: Advantages and Disadvantages of some of the existing solutions

System Overview	Pros	Cons
Wang et al.	Allocation of spectrum band for different flows. QoS policies can be dynamically modified for different active network functionalities. Innovations for improved QoS are possible in this system.	Applying QoS requirements on the local manager could be beneficial, however their proposition does not detail how QoS could be applied for different purposes.
Galluccioz et al.	Simplified network programmability such that the network is more flexible. Sensor nodes are programmed as finite state machines. Reduce information exchange between the controller and end devices.	Since system developers have the liberty of using programming languages of their choice, security for such systems becomes a serious concern, as attackers can use that many platforms to mimic the system.
Di Dio et al.	Status reports for traffic congestion are issued by sensor nodes and the sink to the controller. This process suggests an improved way of traffic handling.	The sink could also be overloaded by aggregated data in a system where network devices are less resourced in terms of memory and processing capacity.
Yuan et al.	The system allows a global view of all routing paths as performed by the controller. Proposed a fault-tolerant system whit the focus of increasing network utilization.	The issue that there is no negotiation between sensor nodes is somehow a lack sing aggregation and redundancy are sometimes relevant the system's data integrity.

Jayashreel and Princy	A general SDWSN framework was proposed. The system was reported to have reduced the energy consumption significantly. Sensor nodes saves a lot of energy since intensive tasks are performed on the controller and OpenFlow switch.	To some extent, non-OpenFlow systems might not be favoured by this framework. Since prioritizing the switching functionality through OpenFlow might require many propriety devices to be used for this process.
Ejaz et al.	A wireless form of energy transfer was presented. A form of scheduled energy transmission was used to power network devices, which could be a solution for energy-limited sensor nodes for increased network lifetime.	This system could be efficient for as long as there is reliable connectivity amongst the transmitters and network device. Otherwise for limited network reach, some end devices could still be energy starved.

2.7 CHAPTER SUMMARY

In this chapter an overview of QoS in WSNs, Resource management in WSNs, SDN and SDWSN was provided. In Section 2.2, provisioning of QoS in WSNs was provided. It was found that QoS provisioning is an important aspect of system design yet providing QoS is challenge for these constrained architectures. In Section 2.3, resource management in WSNs was studied. It was found that resource allocation and utilization plays an important role in QoS provisioning. In Section 2.4, the SDN concepts were introduced. The study provided insights on communication interfaces, controller placements and also highlighted some of the work done on providing QoS in SDN networks, finally, it provided the opportunities for SDN in WSNs. The study found the importance of controller placements in maintaining the network consistency, scalability, privacy, resilience and failover mechanism. Moreover, it was found that in each version of OpenFlow new QoS support enhancements were added. Furthermore, it found that some of the controllers are designed with some limited QoS

support. In Section 2.5, the concept of SDWSN was introduced. From the surveyed work, it was found that SDN programmability has brought many benefits to WNSs and that combining SDN with WSNs has the potential to help overcome the limitations faced in WSNs. In Section 2.6, the existing work for providing QoS in SDWSNs was surveyed. It was found that managing SDWNS networks is less complex as compared to WSNs. This gives hope on the possibility of providing overall QoS for these networks.

CHAPTER 3 ACTIVE NETWORK MANAGEMENT STRATEGY

3.1 CHAPTER OBJECTIVES

A resource-aware OpenFlow-based Active Network Management (OF-ANM) QoS scheme that uses SDN strategies is proposed and implemented to apply QoS requirements for managing traffic congestion in WSNs. This scheme uses SDN programmability strategies to apply network QoS requirements and perform traffic load balancing to ensure congestion control in SDWSN.

Section 3.2 discusses the tools that are used in the simulations.

Section 3.3 describes the experimental setup of the implemented system.

Section 3.4 discusses the simulation results and the achievements of the implementation.

Section 3.5 summarises the chapter.

3.2 SIMULATION TOOLS

3.2.1 ONOS

ONOS is an SDN controller platform designed specifically to meet the scalability, high-availability and performance requirements of operator networks. With this design, ONOS projects itself as a network operating system which offers a control plane for SDN, managing

network elements and executing software modules to provide communication services to end-users and neighboring network domains. ONOS does not just control a single networking element, it maintains the entire network. Hence, it can make network configuration and management, and deployment of new services, software and hardware much easier. ONOS provides capabilities ranging from APIs and abstractions, resource allocation, CLI, WEBUI and system applications, it supports running multiple controllers in a clustered mode where they share network state among each other.

ONOS only extracts the characteristics of the networking elements that are of interest to the core operating system, in that manner some networking layers and protocols used for configuring and controlling the networked elements are hidden from the core operating system. The numerous southbound support provided by ONOS include: NETCONF, SNMP, BGP, TL1, OpenFlow, CLI, P4 and RESTCONF amongst others. ONOS also provide ground-breaking northbound abstractions that permits deployments of new applications in a plug and play manner. Applications can be plugged in to execute on-box using built-in interfaces or off-box using REST or gRPC APIs. Hence, by design, these northbound abstractions turn the complexity of configuration, management and control applications creation, deployment and operation into a simple task. Figure 3.1 describes how ONOS interacts with the underlying network element with the assistance from its providers.

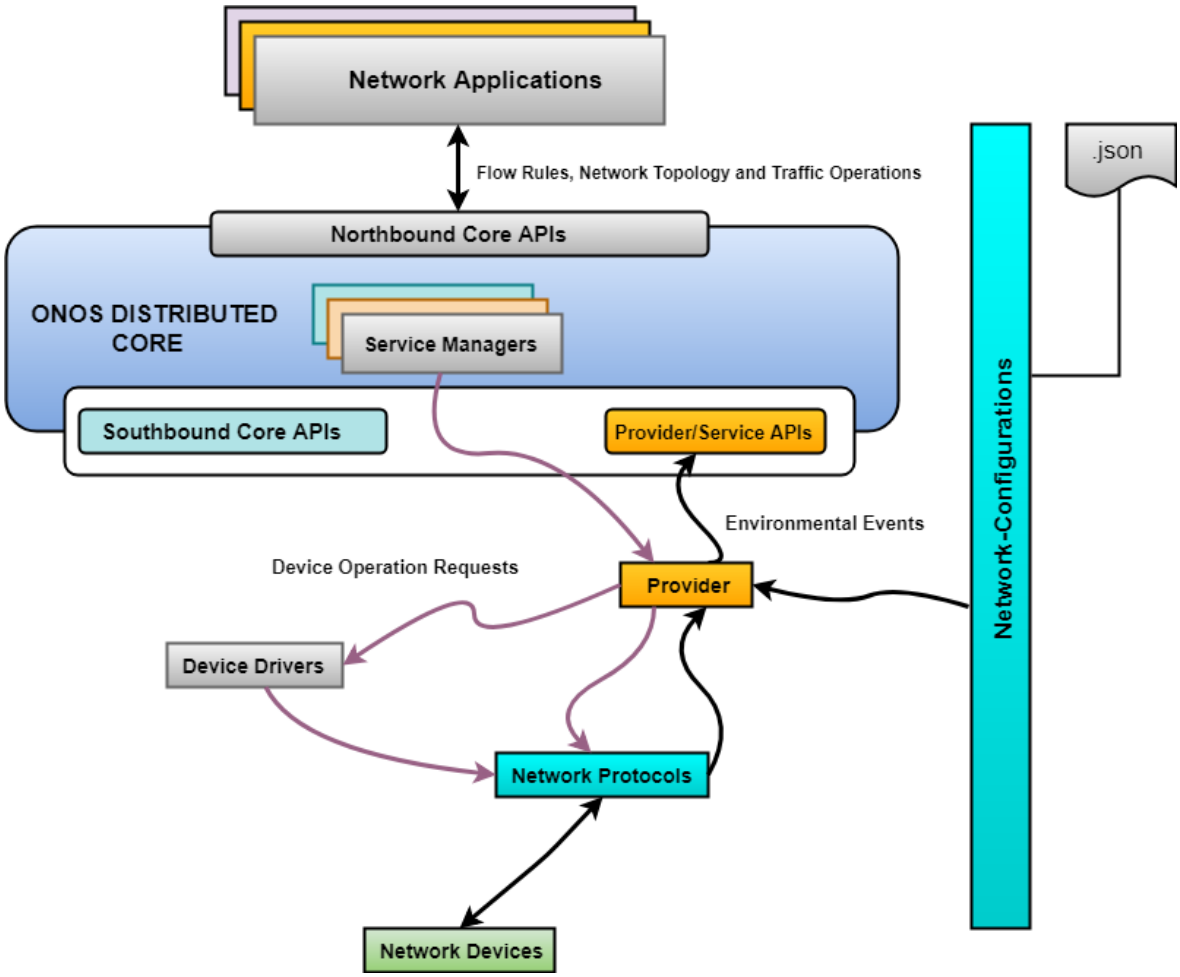

Figure 3.1: ONOS interactions.

3.2.2 NS-3

NS-3[154] is a discrete-event network simulator for internet systems and mainly intended for research and educational use. NS-3 allows the use of C++ and Python and also allows real network integration through its emulation mode. In NS3, diverse modules like 802.15.4, 6LoWPAN and RPL are already integrated for Wireless Sensor Networks. It also works with operating system such as Window via Cygwin and Linux. NS-3 has an OpenFlow switch model for supporting OpenFlow based simulations but the module provides OpenFlow version 0.8.9 which is very outdated and lacks many of the features that the most recent

versions of OpenFlow (V 1.0 – V 1.5) have. One of the disadvantages of this NS-3 OpenFlow module is that it does not offer the support for external controller implementation. However, the latest versions of NS 3 provide an OpenFlow 1.3 SoftSwitch module which enables the NS 3 discrete event simulation to connect to the external controller. Figure 3.2 illustrates the modules which are NS 3 supported.

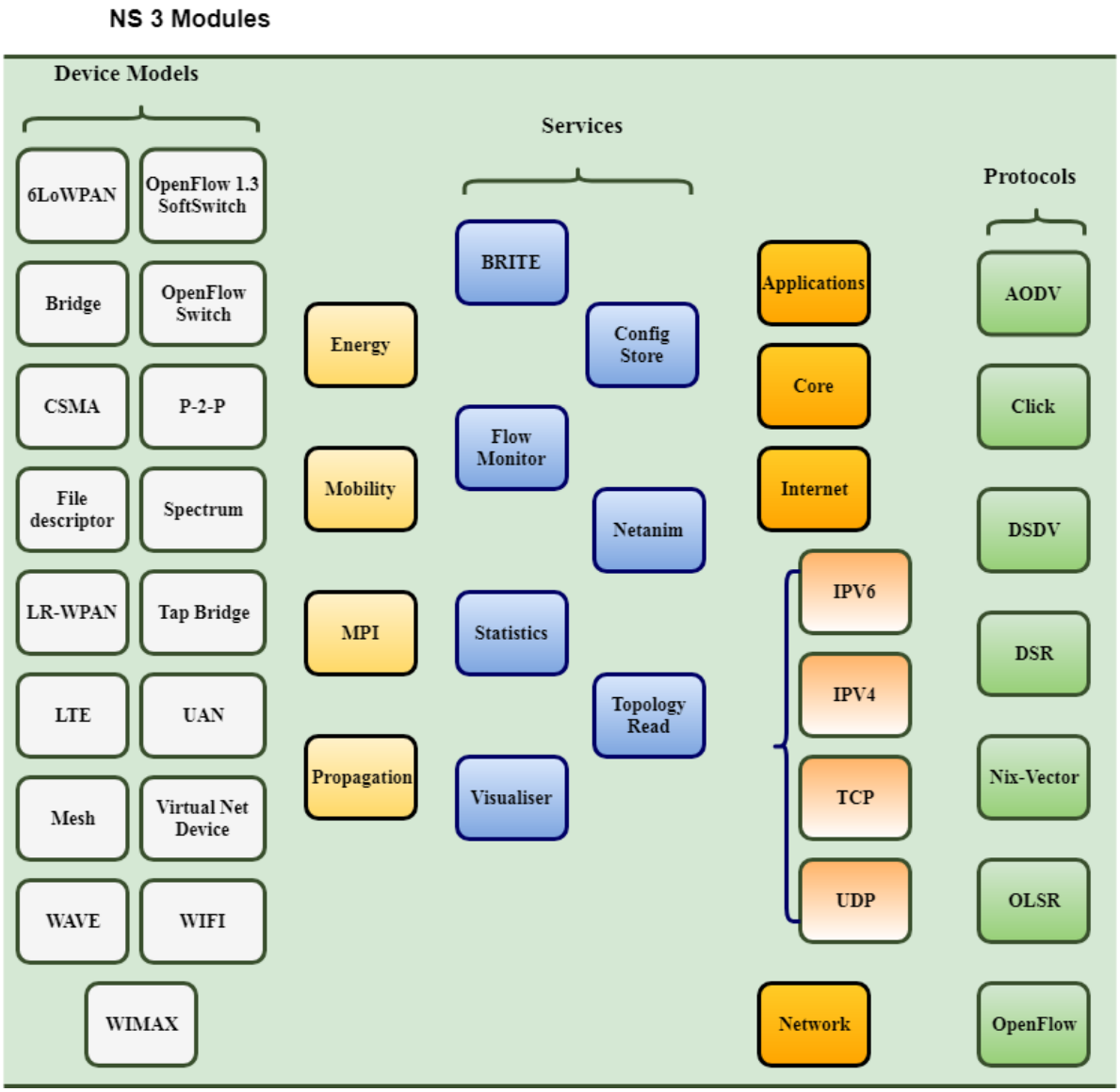

Figure 3.2: NS 3 Supported modules.

3.2.3 OFSWITCH 13

OfSwitch 13 [155] is an OpenFlow switch module that supports OpenFlow 1.3 to improve the NS-3 SDN implementation support. This module also supports the implementation of any desired controller logic for network management. The OfSwitch module is composed of the switch network device, the controller application interface, the OpenFlow channel and the external OfSoftSwitch13 library. The function of the switch network device is to facilitate the interconnection of ns-3 nodes using CSMA channels and devices. The components of each switch device include a set of ports with each of the ports is connected to a CSMA device. The connection between the switch and the controller is either by the switch's own point-to-point channel or the shared CSMA channel. Figure 3.3 depicts the NS 3 integration in to the OpenFlow SoftSwitch framework.

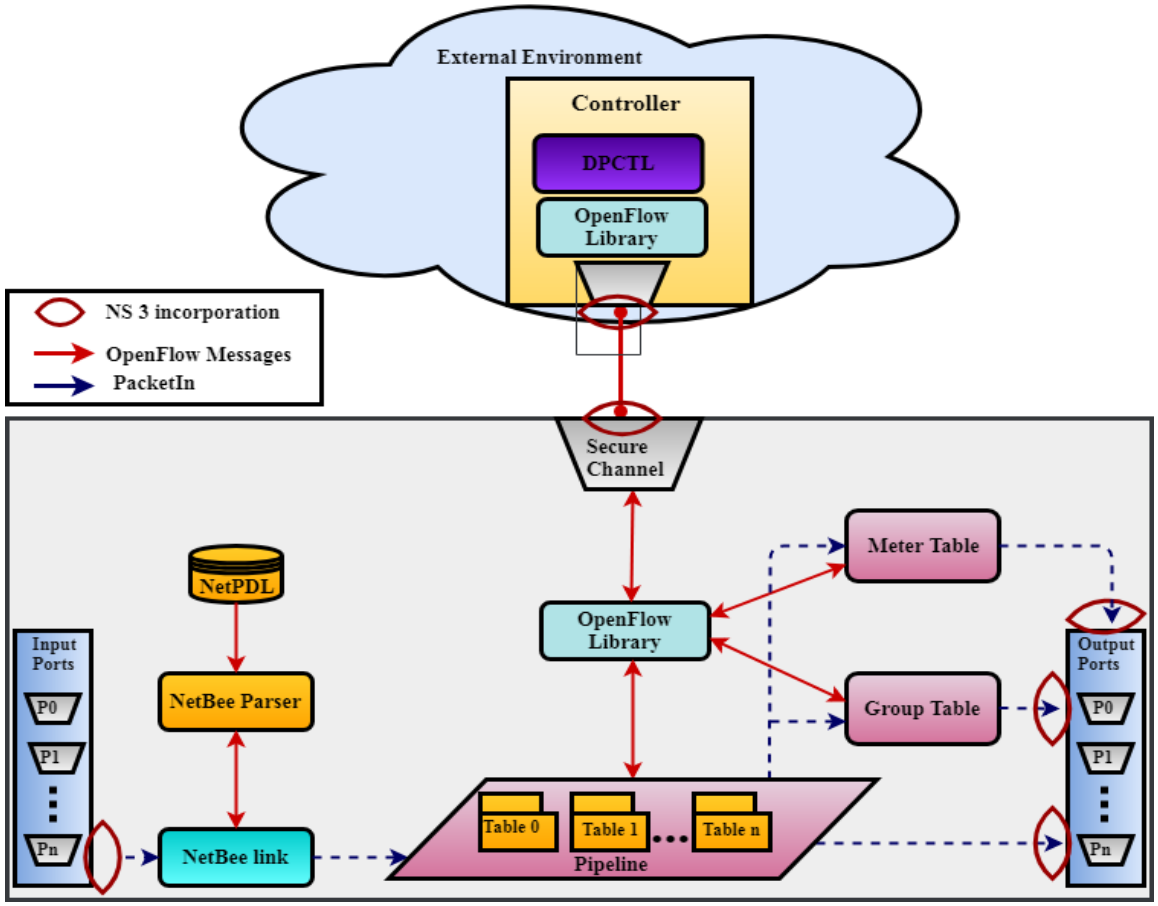

Figure 3.3: NS 3 integration to the OfSoftSwitch13 architecture.

In order to support simple controller implementation functionalities, this module includes an OpenFlow 1.3 based controller interface and an OpenFlow channel. The OfSoftSwitch13 library renders the implementations of OpenFlow pipeline implementation for OFSwitch13 including, input and output ports, and group, meter and flow tables. OFSwitch13 converts the internal messages in to and from OpenFlow 1.3 by making use of the OFLib library services. It also uses Netbee library that is enhanced to incorporate within the OfSwitch 13 module to decode and parse incoming packets. The module maps the simulator and the library by deriving time related functions as irrelevant, the same strategy is applied to ensure that packets are sent and received directly to and from the NS 3 environment.

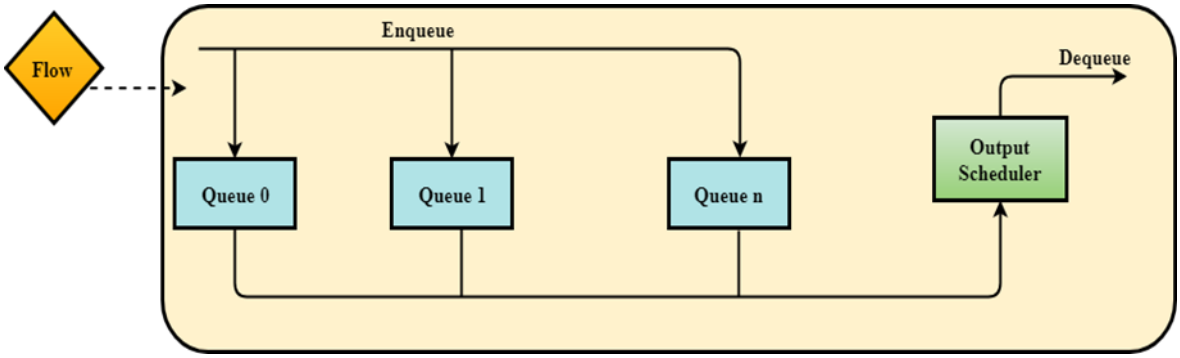

Figure 3.4: The OFSwitch13 Queue internal structure

Packets are received by each switch from an input port, they are then directed to the OfSoftSwitch library for pipeline processing and finally the appropriate actions are executed to the output port based on the action set gathered from the OfSoftSwitch. Figure 3.4 depicts the processing of the flow once it enters the pipeline.

3.2.4 MODIFIED SNMP

Modified Simple Network Management Protocol (SNMP) also known as IPv6 over Low-power wireless personal area networks (6LowPAN)-SNMP [156] is the extended modification of SNMP, motivated to achieve the goals in RFC 4919 (which describes how LowPANs could benefit from IPv6 networking) and RFC 4944 (which describes the frame

format for IPv6 packet transmission and the method of forming IPv6 link-local addresses and stateless auto-configuration of addresses on IEEE 802.15.4 networks). Modified SNMP aims to provide native communication of SNMP messages on 6LowPAns, it is highly efficient with regards to resource usage and it is also fully compatible with the SNMP standard. Modified SNMP uses SNMP header compression and provides extended protocol operations to minimize the number of SNMP messages generated among the SNMP entities.

3.2.5 OPENFLOW

OpenFlow is a standard protocol used for southbound communications between SDN controllers and network devices. Newer versions of OpenFlow have IPV6 capabilities and hence makes it easier to use OpenFlow for IPv6 based networks. Figure 3.5 depicts the components of an OpenFlow switch which includes OpenFlow protocol, an OpenFlow channel and flow, group and meter tables. The OpenFlow channel provides a secure channel for communicating with the controller through the OpenFlow protocol.

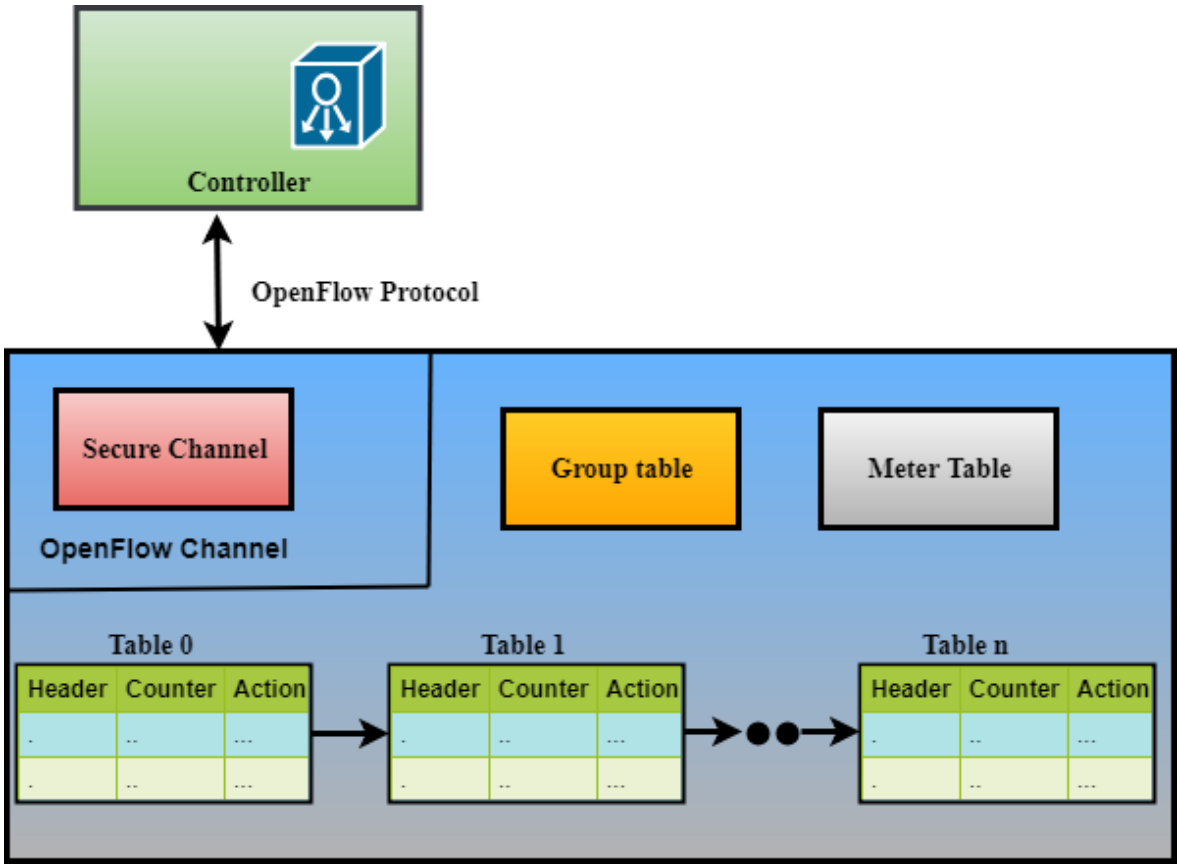

Figure 3.5: OpenFlow switch architecture.

3.3 SIMULATION SETUP

Our approach for using SDN strategies to improve WSN QoS provision by avoiding congestion and efficiently managing the network resources, aims at implementing a central controller (ONOS) that configure and manage the entire network. Our implementation consists of a number of sensor nodes which are clustered and multiple base stations that will act as cluster heads as shown in Figure 3.6. The controller will have direct communication with these base stations. The controller performs QoS tasks such as Path computations, to provide high priority flows with sufficient network resources using Active Network Management (ANM) strategy to successfully forward prioritised flows.

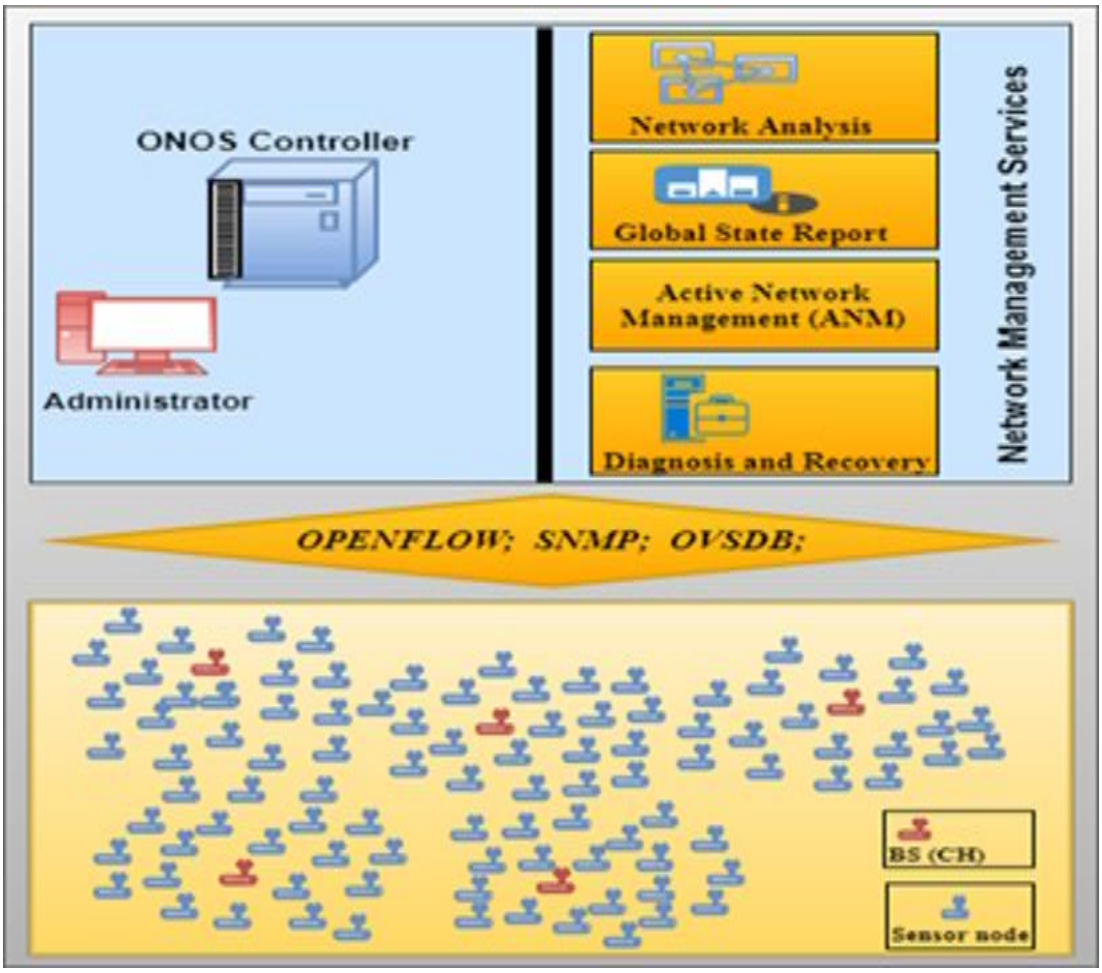

Figure 3.6: An Overview of the ANM Strategy Implementation.

The Controller has an ANM strategy with path computation capability which is motivated by RFC 7567 (IETF Recommendations Regarding Active Queue Management) [157]. The ANM strategy collects link state information i.e.: list of one hop neighbours, channel conditions from source to neighbours, queue length and packet arrival rate; and node state information i.e.: node residual energy. From the collected statistics, a global network view is built and the path computation functionality computes the most optimal paths with the help of the status check function. Algorithm 3.1 describes the check status operations.

Algorithm 3.1: Status Check Function

```
Fetch: GlobalState
NodeState: {SensorID}
Fetch {active/ inactive
NeighbourhoodList
Battery
LinkQuality
Bandwidth }
Fetch: shortestDistanceNeighbour
{
If Node is active
Check: NodeState
If NodeState Negative
Report parameter
If parameter negligible
Select for path computation
Calculate energy efficiency probability function
Else if Node is inactive
Elim.Sensor: SensorID.discard
State.upd: UpdateState.Controller
State.upd: UpdateState.BS
end
```

Algorithm 3.1: The implemented status check function.

Selecting whether a node is appropriate for relaying the data is done through nearest neighbour metrics. If the node has the lowest threshold T (n), that node is therefore considered appropriate.

$$T(n) = \begin{cases} \dfrac{S}{1 - S\ (\Delta C\ mod\ \frac{1}{S})} \end{cases} .if\ n \in f \tag{3.1}$$

Where ΔC is the current best option, S is the desired distance between source node and relay node. Once the node is selected the network status is updated on both cluster and controller level. When the first packets were transmitted only the transmission distance was considered for the threshold computations. However, to avoid unbalanced energy, the probability of being elected was set as a function of node's current energy state relative to the energy of the neighbourhood. The threshold will then depend on the residual energy of the nodes.

$$T(n) = \begin{cases} \dfrac{S_i}{1 - S_i\ \left(\Delta C\ mod\ \frac{1}{S_i}\right)} . \dfrac{E_{Res}}{E_{Tot}} \end{cases} if\ n \in f \tag{3.2}$$

Where S_i is the distance-based selection probability, E_{Res} is the current energy state of the node and E_{Tot} is the initial energy state of the node. f is a set of nodes lined up for selection. The energy efficiency probability of the node will then increase once the threshold is lowered.

$$E_{tot}(t) = \sum_{n=1}^{N} E_{Res}(t) \tag{3.3}$$

Where E_{tot} is the total energy of the neighbourhood. Using these probabilities, paths are computed by selecting the most appropriate node for data transmission.

The ANM strategy operates in such a manner that the Controller queries the network status in time intervals. However, in cases where there is a change in the network the sensor nodes notify the controller of the changes. The path computation functionality finds the shortest path, by taking note of constraint paths (these constraint path computations depend on the QoS requirements of each flow) then use a resource-aware SDN functionality to divert traffic

from the constraint paths which are bound to be congested. This functionality is achieved through the OpenFlow protocol. When a sensor node experiences a high flow arrival rate or the queue length is reaching near the threshold the controller injects a flow attribute into that sensor node to notify the neighbouring sensors to divert the traffic from the path that is at risk of being congested. The operation of the proposed strategy is illustrated in Algorithm 3.2.

Algorithm 3.2; Proposed OF-ANM QoS Procedure

```
Task → Get nodeID ()
Task → Get.LinkStateInformation
if (flowArrivalRate == high)
inject.flowAttribute:
modify.flowForwardingRate
 else if (queueLength >
maximumThreshold)
     Calculate remaining capacity
end_if
if (remaining capacity > 25%)
    Divert traffic to avoid
congestion
Task → Compute.newPaths
     → Get.listOfOneHopNeighbours
     → Get.channelCondition (between
source and one hop neighbours)
     → Get.pathConstraints
     → Get.newPaths
Task → update.flowTable
end_if
 else
transmit.data (destinationID
→get.Route)
end_procedure
```

Algorithm 3.2: QoS Procedure for the proposed ANM

Based on our system model Table 3.1 illustrates the parameters used in our experiments.

Table 3.1: System parameters and values.

Number of Sensor Nodes	100
Distance between Sensor Nodes	Random
Simulation time	60 s
Protocol	OpenFlow 1.3
Switch	OfSwitch 13
Controller	ONOS
Controller Placement	Centralised

3.4 RESULTS

To validate the efficiency of the proposed scheme. The implemented system uses a number of southbound protocols i.e. Modified SNMP, OpenFlow and our proposed scheme (OpenFlow-Based ANM strategy). Given the resource-aware and QoS guarantee of our system, our system model takes careful consideration of the overall communication of the system. In that regard, our implementation focus on efficiently managing network resources as well as traffic routing. Our system uses SDN strategies to enhance QoS oriented network requirement from the central controller since it well resourced. System performances are illustrated in figures bellow in terms of measurement plots. Path computations and flow tables updating given any risk events occurring on the network are performed from the ONOS controller which deploys an active network management procedure that considers network QoS requirements.

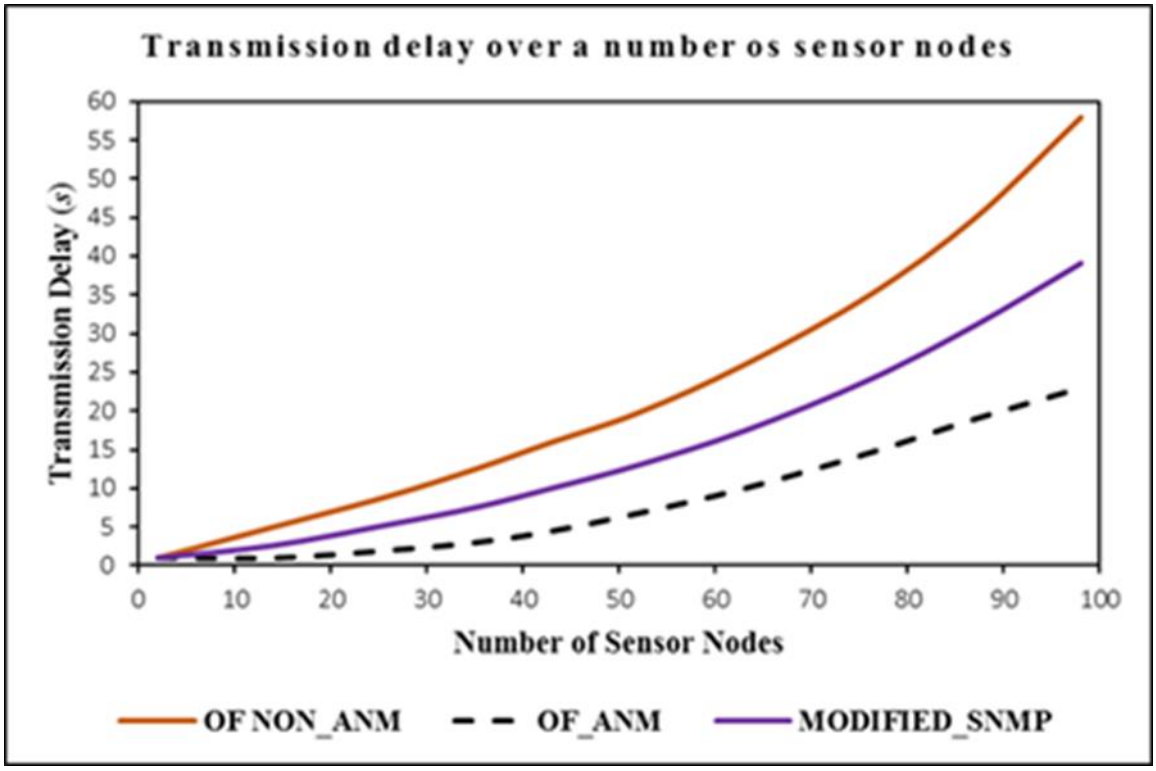

Figure 3.7: Transmission Delay vs Number of Sensor Nodes.

Figure 3.7 presents a transmission delay time in terms of communication given a number of sensor nodes implemented on our system. Based on our experiments, the proposed scheme has the least delay compared to both the NON-ANM OpenFlow and MODIFIED SNMP methods. The proposed scheme has the lowest transmission delay time for a higher number of sensor nodes. On average, OF_ANM produces just below 25% delay, whereas the MODIFIED SNMP and NON-ANM OpenFlow produces around 40% and just above 55% transmission delays. Given this achievement, our proposed approach offer greater communication speed and hence, the network status can be achieved in optimal time.

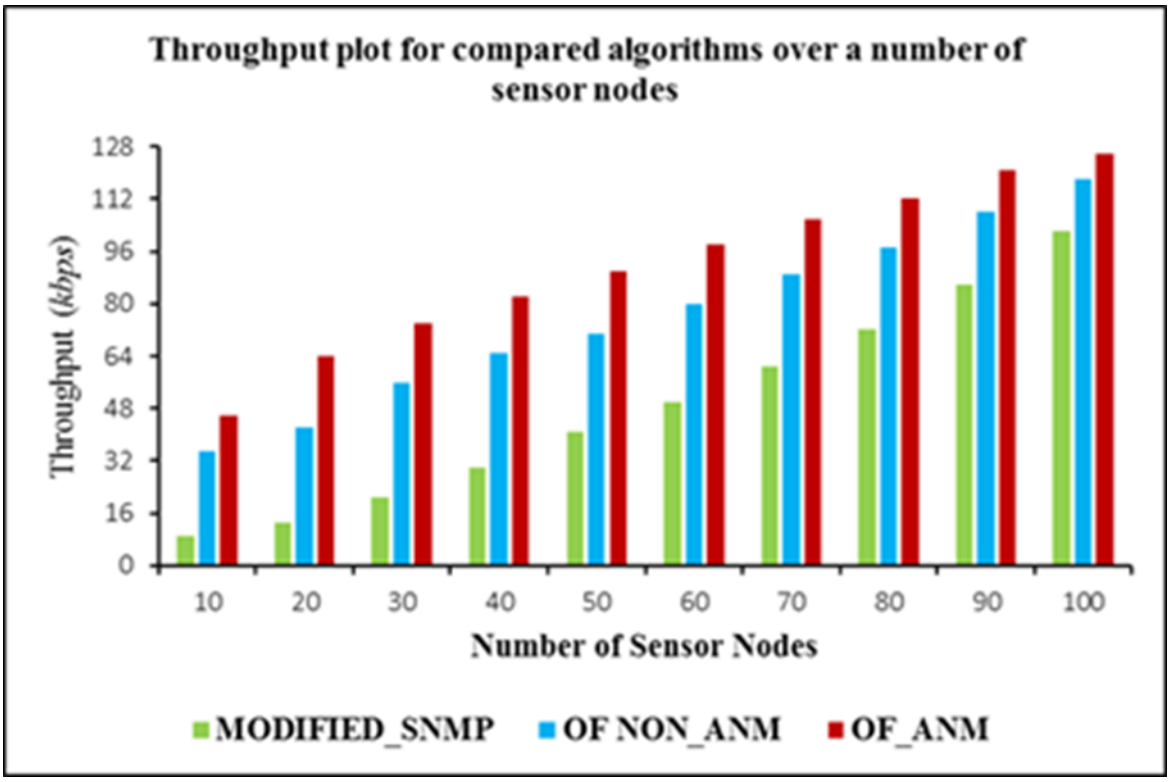

Figure 3.8: Throughput vs Number of Sensor Nodes.

To get the understanding in terms of throughput computation over a number of sensor nodes, results are given in Figure 3.8. Our experiments indicate that, the proposed scheme produced best results. Based on our simulation statistics, OF_ANM performed best compared to both NON-ANM OpenFlow and MODIFIED SNMP in achieving higher throughput computation. The highest throughput computation can be seen between 50 and 100 sensor nodes. Seemingly, NON-ANM OpenFlow also produced better throughput results compared to the MODIFIED SNMP as it indicated higher throughput computation for higher number of sensor nodes. The MODIFIED SNMP has the least throughput computation of all these procedures given our system settings. Based on the performance of the proposed scheme, we have achieved efficient system information integrity and this is good for building better information knowledge of the network.

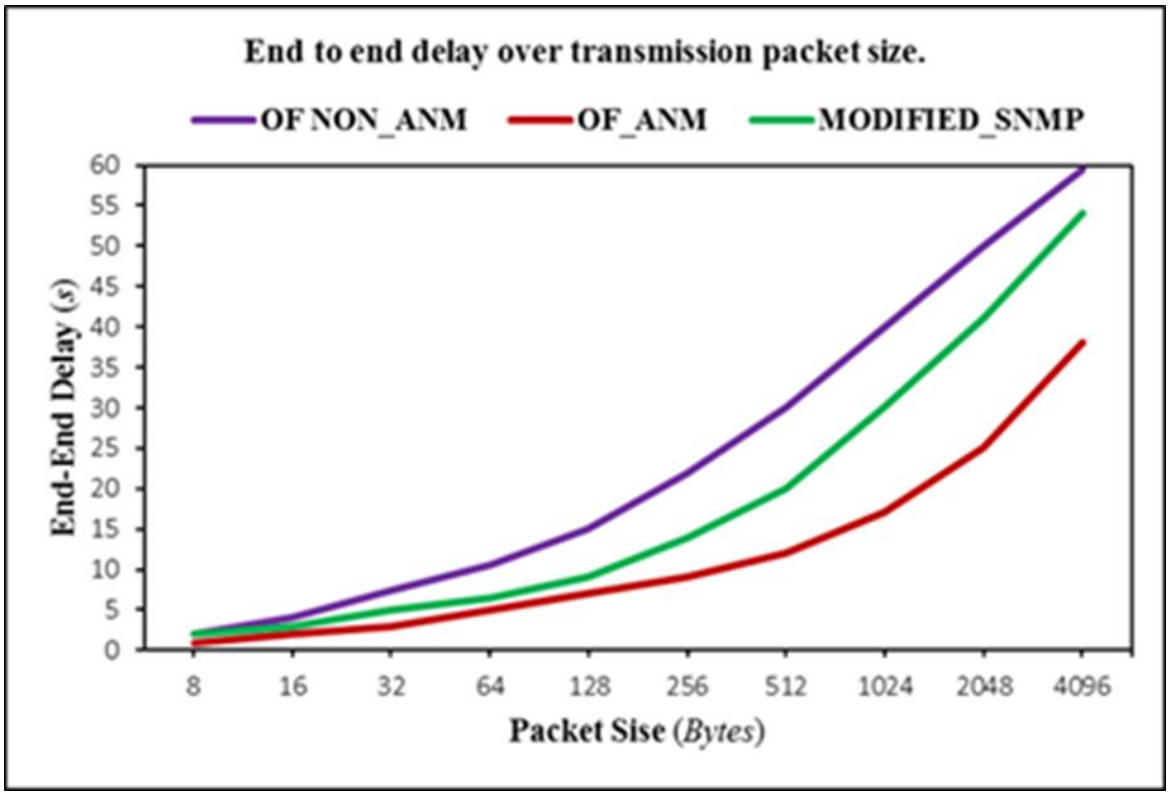

Figure 3.9: End-End Delay vs Packet Size.

Again, to capture our system's efficiency, end-end delay computation is performed for various sensor traffic packet sizes. Figure 3.9 illustrates that the proposed scheme produces the lowest end-end delay compared to both the NON-ANM OpenFlow and MODIFIED SNMP. In this figure, it can be seen that, OF_ANM can handle large sensor packet sizes with minimum delay compared to its counterparts. Given the simulation time as set in our experiment, the MODIFIED SNMP produce better delay compared to the NON-ANM procedure. This shows QoS efficiency of the proposed scheme compared to the other two procedures. This is significant as it also improves the data integrity of the system in terms of information retrieval especially when computation time for data retrieval is key. On average, OF_ANM produces just about 20 seconds delay for high traffic packet sizes, whereas

MODIFIED SNMP and NON-ANM produced 37 and 40 seconds end-end delay respectively. This is also significant as information integrity of the network and the data acquisition speed are greatly improved.

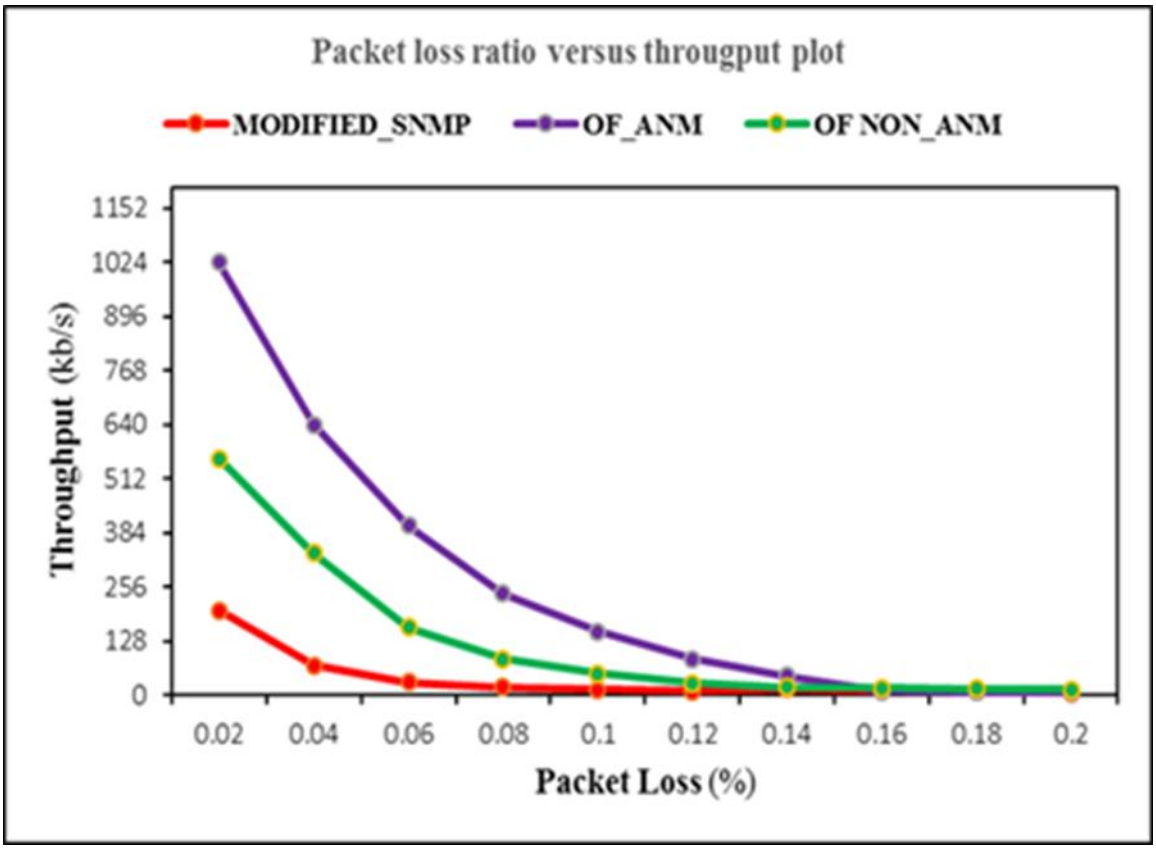

Figure 3.10: Throughput vs Packet Loss.

Figure 3.10 illustrates performance comparisons for the compared southbound protocols. Based on our experiments, as seen in figure 6, the proposed scheme (OF_ANM) produces the best results in terms of traffic packet loss. On average our proposed scheme, produces 20% packet loss for higher throughput, whereas NON-ANM OpenFlow and MODIFIED SNMP produces 30% and 35% packet loss respectively. With these achieved results, our proposed approach improves system reliability in terms of packet delivery ratio.

3.5 CHAPTER CONCLUSION

This work proposed an SDN based QoS scheme that aims at avoiding congestion and managing the network resources in an efficient manner. NS-3 was used for modelling the WSN and ONOS controller was used for configuring and managing the network. The interaction between NS-3 and the ONOS controller was facilitated by the OfSwitch 13 module. The co-operation between the implemented controller and the proposed scheme provides active QoS operations within the network. Our proposed scheme i.e. OpenFlow-based ANM strategy achieved the best performance with regards to throughput, transmission delay, end-end delay and packet loss as compared to the other implemented schemes. It significantly improves data acquisition speed, provides information knowledge of the network to the controller in optimal time and hence, improves the reliability of the system with regards to packet delivery ratio.

CHAPTER 4 DEVELOPING QOS STRATEGIES FOR EFFICIENT SDWSN SYSTEMS

4.1 CHAPTER OBJECTIVES

In this work, flexible SDN strategies are applied to perform traffic balancing as well as resource control and alignment. Resource control and alignment is performed to facilitate and provide compute platform for critical network services. Traffic control and association is performed to avoid transmit path congestion as well as to apply alternative path options computation to enhance packet routing.

Section 4.2 describes the setup of the experiment.

Section 4.3 discusses and analyses the experimental results.

Section 4.5 summarises the work done in this chapter.

Figure 4.1 illustrates an SDWSN model with control and management service layer that facilitated SDN and QoS functionalities.

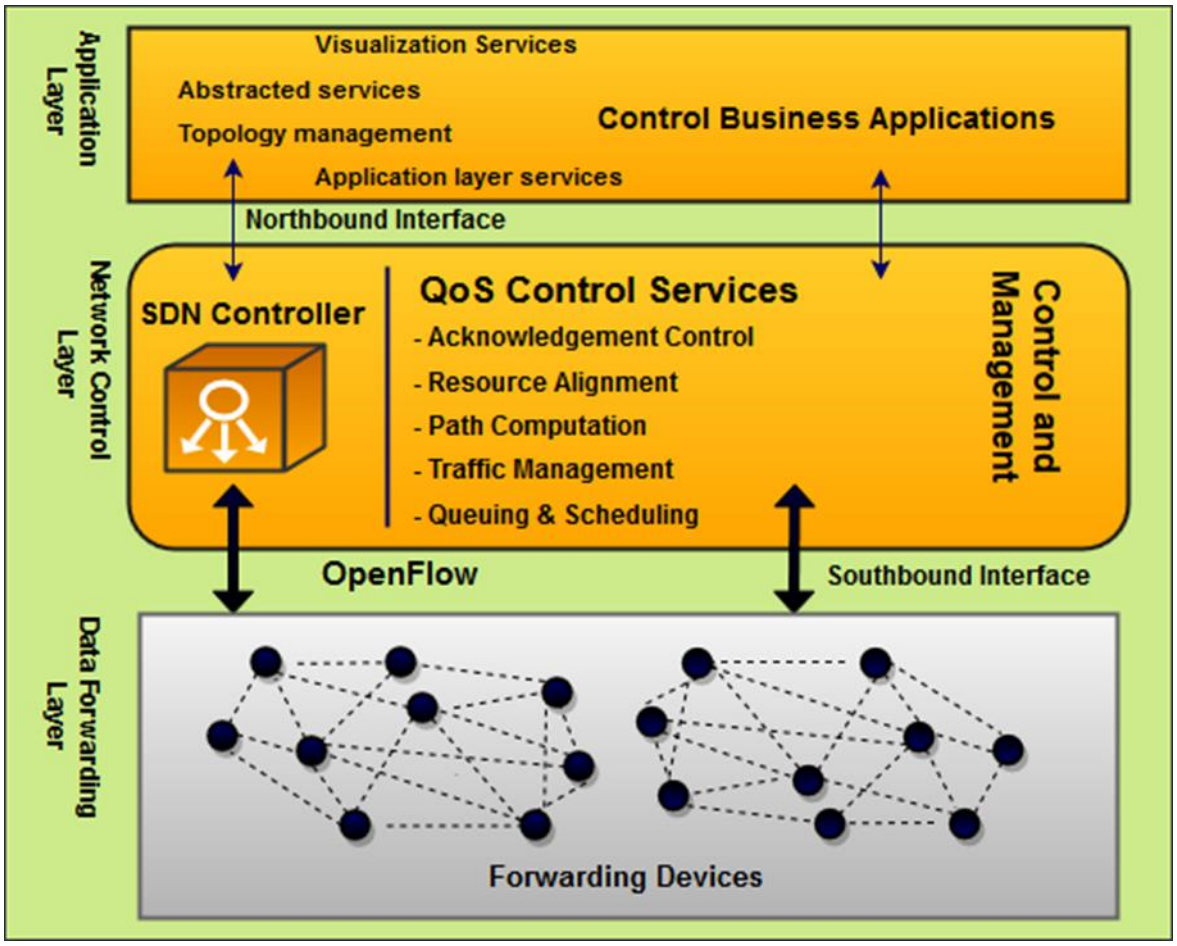

Figure 4.1: SDWSN Architecture with Control and Management Layer.

The SDN controller performs required QoS services in terms of different network activities. The SDN controller implemented in this work is ONOS. The ONOS controller implements top layer network services through the northbound interface. Communication form the controller to the forwarding devices is achieved through the northbound interface, which facilitates the OpenFlow (OF) communication protocol. Intent-based network top-layer applications interfaces the ONOS controller and all the QoS schemes through the northbound interface.

The northbound interface facilitates Controller-services abstraction to access and operate on underlying forwarding devices. This work proposes a QoS Path Selection and Resource-

associating (Q-PSR) scheme for adaptive load balancing and intelligent resource control for optimal network performance. The scheme uses SDN granularity strategies to execute flexible QoS services from the ONOS controller. Abstracted services are finalised with execution reports for any process state to the ONOS controller, for logging and service association.

4.1.1 RESTCONF AND YANG

RESTCONF [158] is a stateless HTTP based protocol used to describe how a YANG specification is mapped to a RESTful interface. It utilises the data store defined in NETCONF to render a programmatic interface for acquiring the data defined in YANG models. RESTCONF was not developed with the intention to replace NETCONF but rather to provide resource-aware device abstraction compatibility and an additional interface that is REST-like oriented. RESTCONF uses HTTP methods to implement the equivalent of NETCONF operations, enabling basic CRUD operations on a hierarchy of conceptual resources. Figure 4.2 depicts the RESTCONF protocol layering.

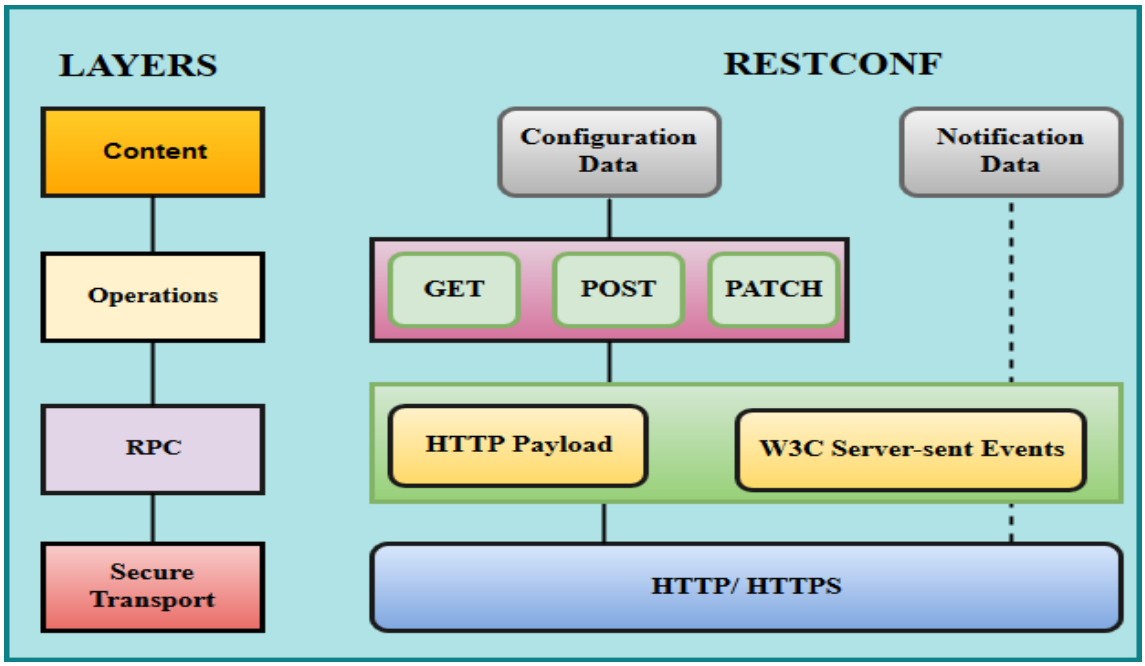

Figure 4.2: RESTCONF protocol layering.

YANG is a data modelling language used to model configuration data, state data, Remote Procedure Calls (RPCs), and notifications for network management protocols [159]. It was initially designed with intended use for modelling data for NETCONF. However, it is also able to support other configuration protocols such as SNMP and RESTCONF. A YANG module describes hierarchies of data that is useful for NETCONF and RESTCONF-based operations, such as configuration, notifications, RPCs, and state data. This allows a complete description of all data sent between a NETCONF client and server. Figure 4.3 describes the YANG operations within the ONOS framework.

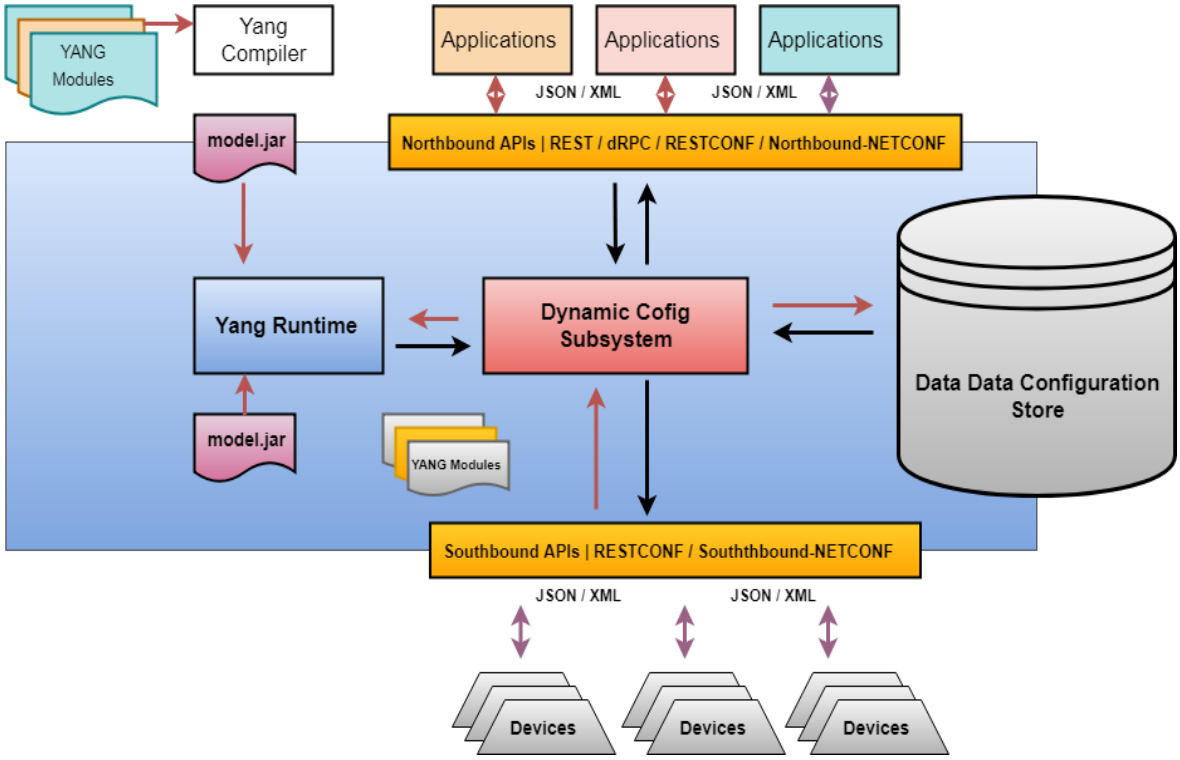

Figure 4.3: YANG Modules Within the ONOS Architecture.

4.2 SIMULATION SETUP

The designed system uses ONOS (Junco {version 1.9.0}) controller as its core SDN controller. ONOS supports application extensions, programmable abstractions and core

distribution. This makes it well suited for SDN implementations that involves active network operations. This controller is implemented in a separate virtual machine instance- Controller-VM to the one modelling the wireless network. The modelled network is then implemented on another virtual machine instance- Network-VM. The network is implemented using Network Simulator 3 (NS-3). The network is implemented in terms of clusters. Each sensor cluster has a base station, which is implemented as a Cluster Head (CH). The OFSwitch-13 module facilitates communication between the Controller-VM and the Network-VM, which is an OpenFlow 1.3 switch module for SDN developments using NS-3.

In Figure 4.2, a description of the modelled network system is shown. The ONOS controller access each sensor cluster through the CH. Each CH implements controller updates to the underlying sensor nodes. Active and abstracted top layer applications, access the controller through the northbound Representational State Transfer (REST) Applications Programming Interfaces (APIs). Both our controller VM and network VM instances run on VMware Workstation Player (version 12.5.7). On each VM, a Linux distribution (16.04) Operating System (OS) is installed. ONOS supported REST APIs can be used facilitate application and service programmability of most high-level languages.

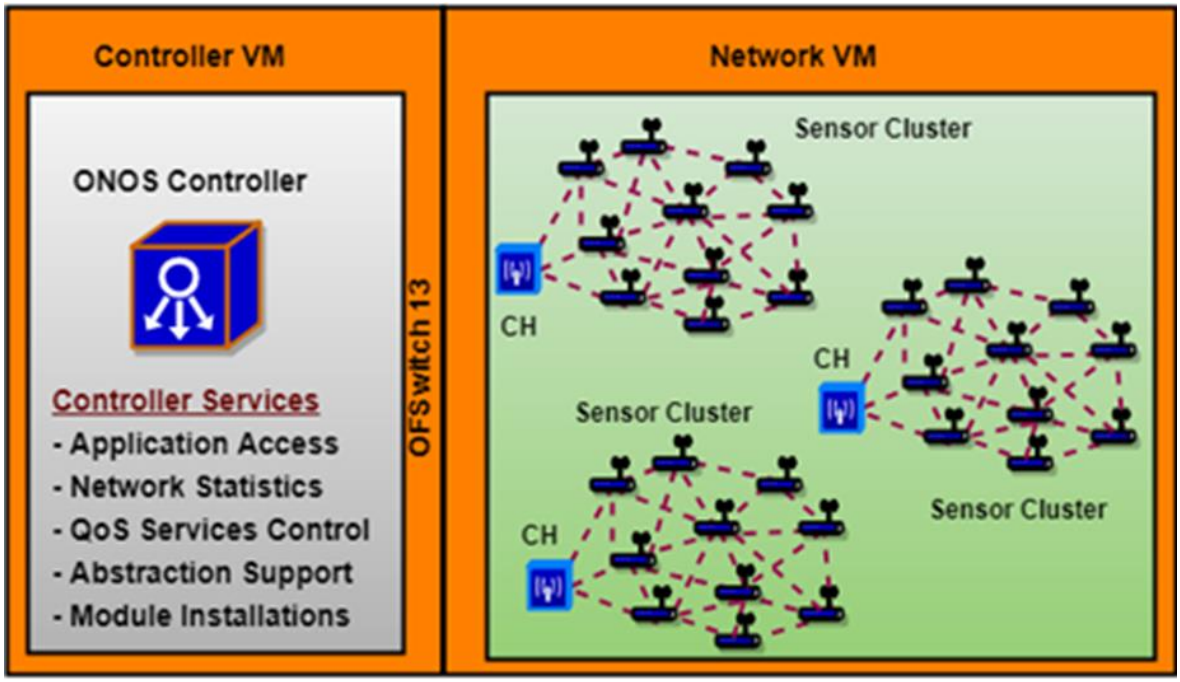

Figure 4.4: Modelled Network System.

The controller implements different network operations to network nodes using flow-rules. It applies role and rules on sensor nodes using the CH. The CH collects sensor information and aggregate it to the ONOS controller. The hierarchical approach of implementing a CH, serves the purpose of conserving compute energy amongst sensor nodes within a cluster. Table 4.1 describes the system parameters implemented in our network model.

Table 4.1: System parameters

Parameters	Value / Description
Network Size	600*900
Aggregate Efficiency	Sensor data / Total no. of nodes
Node Mac Protocol	IEEE 802.15.4
Buffer Size	512 (KB)
Max. Data Rate	256 kb/s
Simulation Time	Set depending on requirements

Figure 4.3 provides an illustration of controller-abstracted services. Of particular emphasis, an abstraction of Controller-QoS services, which are aimed at enhancing the performance of the network. The controller implements these QoS abstraction services through the RESTCONF using YANG specifications. Abstracted services promotes network flexibility and innovation.

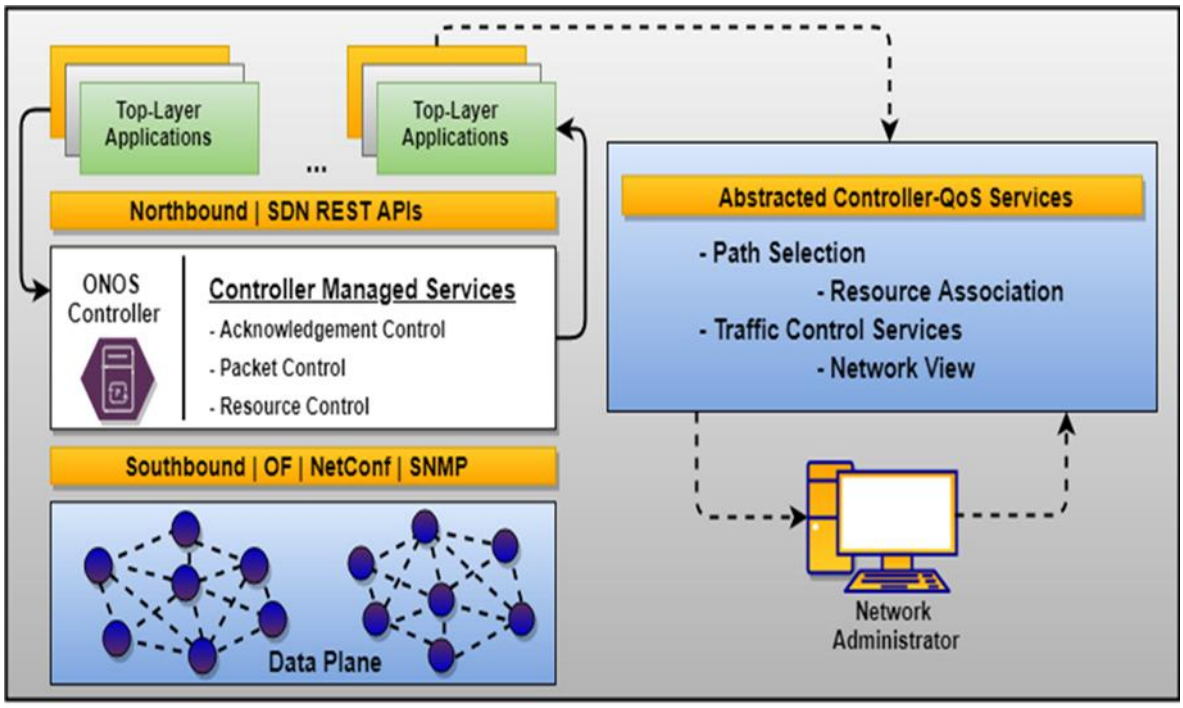

Figure 4.5: An Illustration of Abstracted SDN Implemented Controller-QoS Services.

The SDN network programmability aspect provides opportunities to operate on network activities using high-level programming languages. This facilitates the controller's capacity to implement target operations on different network resources. Hence, this advances the SDN's capability to effectively manage network resources. Figure 4.5 provides this illustration by means of indicating control processes on different layers of the architecture. The implemented system performs abstractions using a java abstraction method, which is a class that extends the abstraction method. This is performed by defining a specific global class for Controller-QoS operations. Abstraction are implemented to manipulate constructed object services, which are extended when creating the actual class. Abstracted services are constricted by the controller, so that, their operations do not affect other normal services.

A pseudo code of the proposed algorithm is given in Algorithm 4.1. The implementation of our proposed scheme focuses on data aggregation to provide efficient path selection, and facilitates the sharing of network resources to achieve network reliability. Based on the

implementation of our system, the following assumptions must be true to meet the operational requirements of the proposed scheme:

- Each resource is well defined on the controller.
- Data elements pertaining to sensor nodes, are known to the controller and can be pooled when required.
- The controller implements different path computations to support smooth networking.
- Network services requiring committed resources, are facilitated using resource abstractions.

Algorithm 4.1: The Proposed Q-PSR Scheme

```
1.  perform: net_check get.net_state()
2.  if net_state is normal
3.    compute: net_state info map.to CtrlNode
4.  end_if
5.  else if
6.    net_state is congested
7.    flag: netFlag ((CtrlNode (get.Extract(&flag))))
8.  end_ifelse
9.  if net_state == netTraffic(linkState(link.full))
10.  compute: new qpsrSelect (newPath)
11.  set: new newPath → linkState (new.link)
12.  map.netTraffic → linkState (new.link)
13.  end_if
14.  check: → this
15.    for all network events
16.      if net_state = unbalanced net.Resource()
17.        perform: net.Resource scout
18.        associate net.load → net.Resources()
19.        perform net.load → new net.Resources()
20.        end_if
21.    end_for
22. net_state new.info → map.CtrlNode
23.perform netView → CtrlNode(net_state(info))
```

Algorithm 4.1: The Proposed Q-PSR Scheme.

The proposed scheme performs traffic check functionality, to establish abnormal networking events. It uses the SDN flexibility to perform load balancing and associating network load to less committed resources. Every network event is reported to the ONOS controller for further network operations and command through the proposed Status Check Function. This improves the knowledge-base of the controller. The controller then implements QoS requirements services based on network events. The ability to abstract specific network resources promotes absolute flexibility and advances the SDN programmability. When abnormal traffic network events occur, the proposed scheme generate a flagged alarm to notify the controller of this event. The controller uses a reference pointer to the flag, then gets the actual index of where in the network, the flag is raised.

4.3 RESULTS

Based on the implemented approach and the proposed strategies, this section provides detailed analysis on achieved results. Since our work is based on SDN strategies, analysis of achieved experimental results will be exhausted in to verify the impact of SDN in WSNs. Our analysis will be largely focused on wireless network traffic control as well as resource management. The efficiency of the proposed scheme will be tested against other competing schemes or algorithms for this particular research focus.

Based on the proposed scheme, the ONOS controller can get sensor locations and use this information to compute alternative unexhausted paths. As a result, the controller builds flow tables for different sensor nodes by gathering path hop-counts and path traffic-states. Sensor nodes generates flow-based packets which they transmit to the ONOS controller via the CH. With this information, the controller is able to build a network knowledge base. Thus, QoS requirements can be implemented by the controller to facilitate the association of network resources, given the transmission quality and traffic status.

To verify the efficiency of our proposed scheme, we compare it with Topology Aware Resource Adaptation (TARA) [160] schemes and Dynamic Alternative Path Selection (DAlPaS) [161] scheme in terms of hop-by-hop delay, the success of received packet rate and associated packet loss percentage. Results are analysed in terms of how effective each scheme is in terms of managing network resources. Figure 4.4 presents performance comparisons of these three schemes in terms of average hop-by-hop delay.

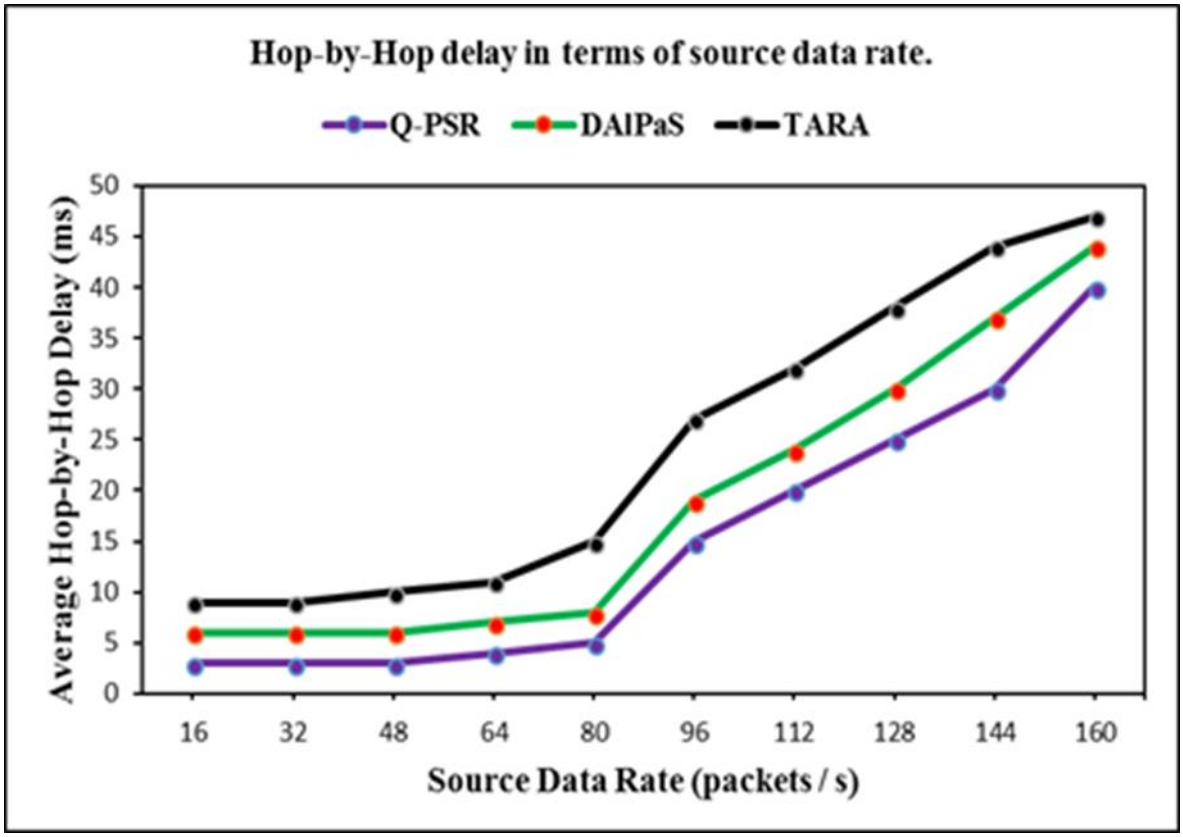

Figure 4.6: Average hop-by-hop delay based on source data rate.

Performance results in Figure 4.6 indicates that the proposed scheme produces the best hop-by-hop delay. It can also be observed that DAlPaS performs better than TARA. On average Q-PSR produces a hop-by-hop delay of about 18 milliseconds, whereas DAlPaS and TARA produces 21 milliseconds and 24 milliseconds delay respectively.

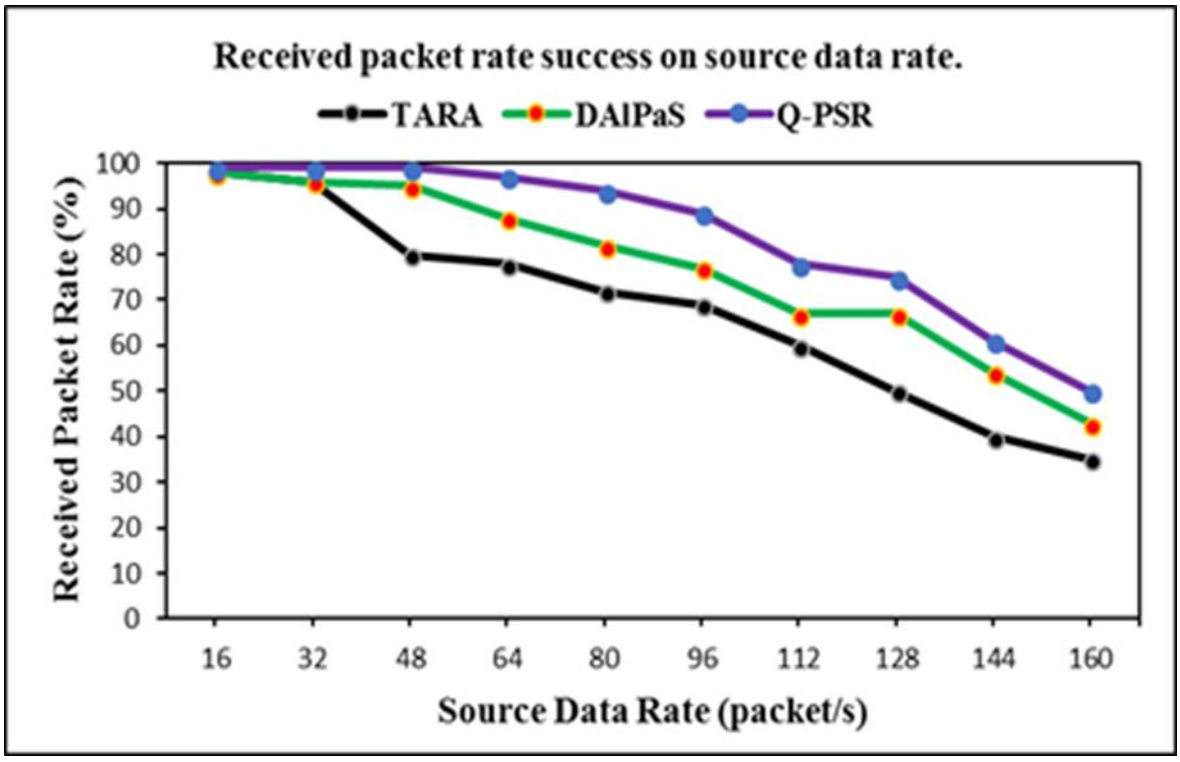

Figure 4.7: Received Packet Rate Success on Source Data Rate.

Figure 4.7 presents an efficiency plot for resource utilization in terms of received packet rate success, given different source data rate. From this plot, it can be observed that Q-PSR presents the best packet rate delivery compared to both DAlPaS and TARA. On average Q-PSR produce about 75%, as DAlPaS and TARA produce 70% and 65% respectively. This performance by Q-PSR is significant as it indicates how efficient our scheme restricts network congestion. This also indicate that the proposed scheme has the best strategy to select best transmission paths.

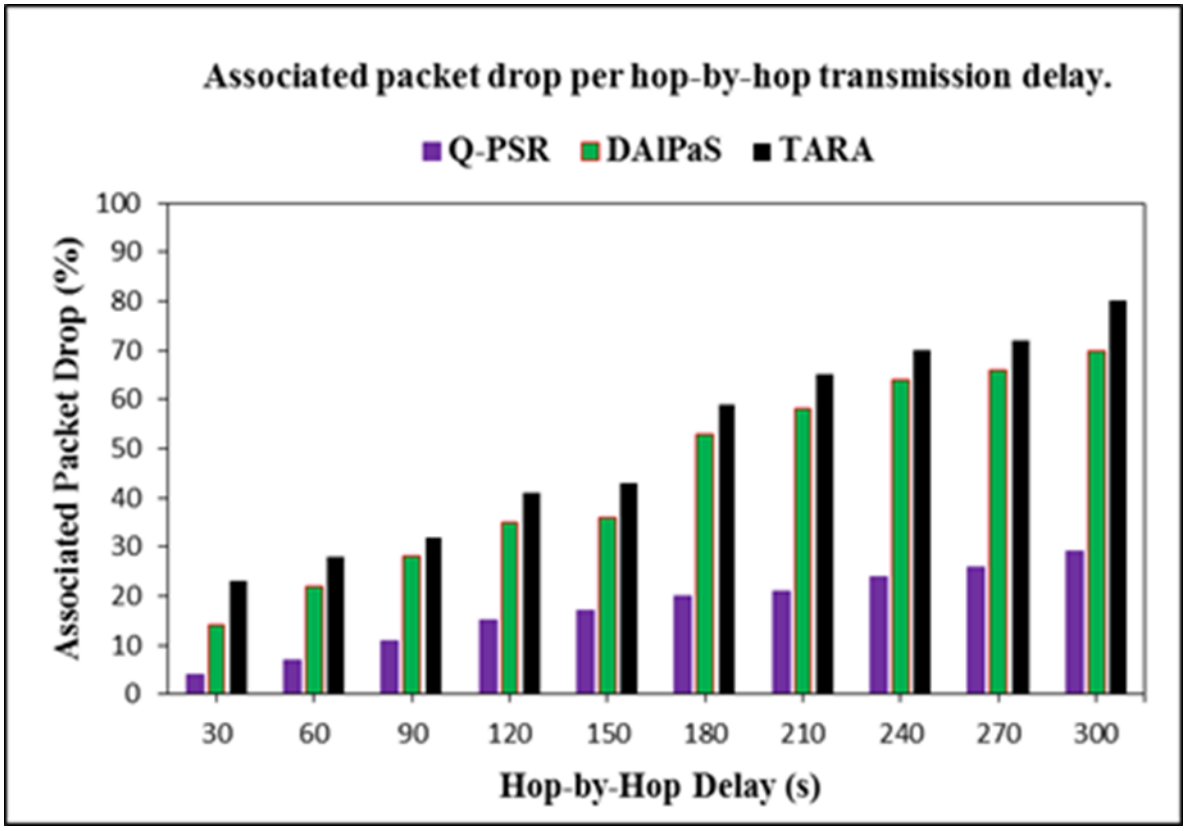

Figure 4.8: Associated Packet Drop Percentage per Hop-by-Hop Transmission Delay

Figure 4.8 presents an associated packet drop percentage per hop-by-hop transmission delay. This plot describes the ability of each scheme in terms of resource association per packet transmission time. It can be observed that, Q-PRS has the least associated packet drop percentage compared to TARA and DAlPaS. It can also be observed that DAlPaS performs better than TARA. On average Q-PSR has about 17% total associated packet loss, whereas DAlPaS and TARA has 32% and 45% respectively. This is a very significant performance by the proposed scheme, since this performance ensures efficient energy utilization through path selection. This also indicates its ability to associate and use network resources efficiently.

4.4 CHAPTER CONCLUSION

This work implemented a QoS Path Selection and Resource-associating (Q-PSR) scheme for adaptive load balancing and intelligent resource control for optimal network performance. With experimental results, we have verified the efficiency of this scheme against Topology Aware Resource Adaptation (TARA) schemes and the Dynamic Alternative Path Selection (DAlPaS) scheme. The proposed scheme has on all scenarios, produced the best results. However, DAlPaS performed better than TARA in all conducted experiments. Remarkable results have been realized based on the proposed scheme. Q-PSR uses SDN strategies as its core functionality, with very light computational methods.

CHAPTER 5 DISCUSSION

Informed by the extensive study performed in this work, estimable results were achieved hence this work will open opportunities for other researchers to venture into the field of SDWSN. This work exploited the flexibility and programmability of SDN by proposing QoS strategies that efficiently manage network resources. Applying QoS requirements is essential to every network computing application. However, QoS requirements have to be applied on well-formulated network policies. SDN can improve the application of QoS requirements by implementing service abstractions to associate different resources with various network activities. WSNs can also benefit from SDN programmability strategies to enhance traffic computations through performing flexible path computation checks. This can be useful continuously monitoring sensor transmission paths to balance abnormal traffic. With careful planning and functional implementations, SDN strategies can be used to flexibly balance traffic loads as well as manage network resources through programmability and network services abstraction.

It is worth noting that Congestion in WSNs leads to increased packet loss ratio, increased latency, increased delay, low throughput, wastage of node energy, increased retransmissions which lead to more energy wastage and hence deteriorates network performance and affect the reliability of the network. In that regard, this proves the importance of controlling congestion since it results in the degradation of QoS support. This work implemented an SDN programmable strategies that ensure QoS provisioning by efficiently managing network resources and network congestion control in WSNs. Firstly a status check function

was developed to fetch specific data on the global network state. To take advantage of the status check function capabilities an OpenFlow-based strategy called OF-ANM was proposed developed and implemented to manage network congestion in WSNs. WSNs play a greater role in short range communication networks. Their application adoption to mission critical systems has increased tremendously in the past decades, due to their deployment simplicity and reasonable implementation costs. However, these technologies have a number of limitations subsequent to their computing capacities. Hence a lot of past research approaches have tried to improve their capabilities in terms of processing, communication, resilience, etc.

In every computing network, efficient network traffic and resource control, are key to realizing supreme network performances. Traffic control techniques focuses of applying QoS metrics to balance network traffic load. This aspect of networking is actually performed on network data transmission links either through hop-by-hop or end-to-end computations. However, this technical aspect alone is not sufficient to produce supreme network performance, since load –balancing alone does not enhance the overall performance. Resource-aware QoS techniques are also important on the enhancement of network capacity. Resource optimization focus on the efficient management of network resources to support network demands and handling of traffic operations. Furthermore, this work developed and implemented controller-based QoS strategies (Q-PSR scheme) for adaptive load balancing and intelligent resource control for optimal network performance. These strategies implemented flexible SDN strategies to perform resource alignment based on different networking tasks are formulated and implemented, applied programmability strategies to perform path computation activities to control sensor path traffic and implemented service-oriented abstraction functionality to perform scheduling activities for various QoS requirements.

To evaluate the performance of the implemented strategies the experimental results of OF-ANM were compared with those of Openflow and modified SNMP in terms of transmission delay, throughput, end-to-end-delay and packet loss. These results can be seen in Figures 3.7

to 3.10. The proposed OF-ANM with the help of the status check function outperformed openflow and SNMP in all the performance metrics. However, modified SNMP produced better results than OpenFlow with regards to transmission delay and end-to-end delay. On the other hand Openflow yielded a higher throughput at a lower packet loss ratio as compared to modified SNMP. The proposed QoS strategies proved to provide improvements in data acquisition time and network information collection and hence increased the reliability of the network.

This work further proposed QPSR scheme- an artificial intelligence motivated controller-based strategy. This strategy exploits the flexibility of SDN by exploiting the capabilities of the status check function, the network programmability and the ability of OpenFlow to permit global visibility of network resources. Figures 4.6 to 4.8 display the results of the experimental performance of Q-PSR compared with DAlPaS and TARA. The experiments were conducted in terms of how each scheme manages the network resources and the performance metrics used are in terms of hop-by-hop delay, the success of received packet rate and associated packet loss percentage. In all the performance metrics Q-PSR yielded the best performance of all three. DAlPaS on the other hand outperformed TARA in all performance metrics. Figure 4.6 Displayed that even when the source data rate is high our proposed strategy still managed to experience lower hop-to-hop delay. This is a commendable achievement by the proposed scheme. Figure 4.7 Showed how received packet rate is affected by source data rate, the proposed scheme performed near optimal as most of the packets were received. Figure 4.8 displayed how hop-to-hop delay affected packet loss and Q-PSR yielded the lowest data loss since it achieved low delay value. This significant achievement by Q-PSR proved that the path selection promotes efficient energy utilisation.

CHAPTER 6 CONCLUSION

Extensive preference for using WSNs has grown over the years and has brought about new approaches to enhance their applications. WSN application systems are deployed in different fields of monitoring and control. Their application adopting comes as a result improving their potential for high-risk monitoring and control systems. However, these technologies are not applied in full-scale due to their capacity limitations. The SDN paradigm is evolving modern computer networks by relieving network nodes from performing control functions through programmability. Due to the sensitivity of aggregated data in WSNs, especially for monitoring applications, network performance is key. Network reliability and data integrity is essential for life concerning systems. However, to realize WSN systems that are highly responsive, accurate and resilient, a combination of efficient QoS measurements and networking services must be in place.

Efficient QoS measurement is achieved through service level management (SLM) and service level agreement (SLA) tools applied to an information system that has a particular goal in terms of its operation such as maintaining optimum performance in networking. QoS as a network measurement looks at availability of network resources as well as effective network performance. In WSNs, network load balancing requires a well-characterized network architecture that is cautious of sensor nodes energy, by minimizing high sensor network traffic. This can be achieved by implementing effective sensor data aggregation, traffic routing measurements and end-to-end communication methods. Hence, network load balancing is critical for all sensor nodes within the network since their energy consumption will not be the same given their region of transmission from different source nodes.

The Primary target of SDWSN is to mitigate the traditional WSN challenges by decoupling the data and control functionalities of these networks. In an SDWSN framework one or more centralized SDN controllers manage the network by assigning tasks gathering data, manage the network topology, make routing decisions and many more thereby offloading all the compute expensive tasks such as energy management, QoS control and path computations from the sensor nodes which will only have a forwarding and sensing roles. This type of architecture maps well with the WSN architectures since in WSNs sensor nodes are managed by one or more sinks which give tasks to sensor nodes and also collect the sensed data. Equipped with the global view of the network, the controller can offer efficient resource allocation and management through centrally controlled topology control, scheduling and routing.

This work developed and implemented intelligent SDN strategies for ensuring efficient resource management and ensuring QoS provisioning in WSNs. This was greatly motivated the wide adoption of WSN that proved their potential for integration in the future networks. However this adoption also prompted the importance of applying QoS guarantees into these resource constraint networks since it is difficult to provide the desired QoS results on them due to their dynamic architectural nature and efficient resource management being key to achieve the desired QoS in WSNs.

In Chapter 3, SDN programmability was used to developed strategies as a means to apply QoS requirements for efficient network management in WSNs with regards to response time, efficient resource management and congestion control. Equipped with the large knowledge pool of already existing literature, the candidate was able to formulate possible solutions for dealing with WSN inherent limitations. With OpenFlow as the main protocol used in this work, an Openflow-based strategy (OF-ANM) was developed through the use of the ONOS controller to mitigate the network congestion challenges. To validate the efficiency of the proposed OF-ANM strategy, its performance was compared to Modified SNMP and the OpenFlow protocol. The proposed OF-ANM strategy produced commendable performance. The performance results can be seen in Figures 3.7 to 3.10. The results indicates the highest throughput achievement from the proposed OpenFlow-based strategy. However, OpenFlow

also recorded a higher throughput achievement and lower packet loss compared to modified SNMP even when the number of sensor nodes increased which shows the potential of OpenFlow, though the performance of modified SNMP is also noteworthy since it was specifically designed for 6LoWPANs. The good thing about the ONOS Controller is that it already supports a number of protocols and it also provides a plug a play support for applications that are not part of the ONOS core which also endorses why most companies have adopted ONOS as their controller of choice.

Furthermore, an artificial intelligence-motivated Q-PSR strategy which works collectively with Openflow resource visibility and status check function uses adaptive SDN strategies to implement QoS requirement from the SDN controller is implemented in Chapter 4. To apply SDN strategies to WSNs, thorough investigations and careful planning must be undertaken. This is mainly because; WSNs are particularly limited in terms of capacity whereas SDN strategies might require optimal resources to perform efficiently. SDN has the potential to advance computing capacity in WSNs by evolving a manner as to how these systems are operated and managed It can also promote service abstraction and resource management flexibility using object-oriented programming languages. This will then; improve their application deployments especially for systems that require effective response, high security and operate on sensitive data. Q-PSR produced the best performance that ensures energy efficiency through path selection. This is also an indication of its ability to efficiently associate and utilize the network resources. Figures 4.6 To 4.8 Show the achievements of Q-PSR. The overall performance of Q-PSR indicates how the flexibility and programmability of SDN can transform the operations of resource constrained networks such as WSNs.

This work has also contributed to SDWSN literature through the publications produced in the form of review and result papers. Finally, the proposed strategies proved to associate and use network resources, ensure efficient energy use, achieve commendable throughput performances, significantly improve data acquisition rate and information integrity of the network and provide optimal communication speed. Therefore, it is safe to say the proposed strategies can be adopted in the mitigation of these constrained networks inherent challenges.

As part of future work, the work done in this research can be implemented in real life situations as a proof of concept to support the experimentations done through simulations. However, in the future more can be done to improve the feasibility of the proposed SDWSN schemes in an actual (physical) network such as exploring the use of parallel programming coupled with powerful computing elements such as Graphics processing units (GPUs) to transform how WSNs are designed, configured and maintained. Moreover, the use of GPUs may possibly bring forth the ability of these networks to operate and respond in real-time.

REFERENCES

[1]. I.F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci. “Wireless sensor networks: a survey.” Elsevier Science B.V.: Computer Networks, Vol. 38, pp. 393-422, Jan. 2002.

[2]. H. Ghasemzadeh, M. Rezaeian, F. D. Touranposhti and M. M. Ghasemian, “BN-LEACH: An Improvement on LEACH Protocol using Bayesian Networks for Energy Consumption Reduction in Wireless Sensor Networks,” In Proc. IEEE 7th International Symposium on Telecommunications (IST’2014). Tehran, Iran, pp. 1138-1143, Sept., 2014.

[3]. Karthik, S. and Ashok Kumar, A. (2015) Challenges of Wireless Sensor Networks and Issues Associated with Time Synchronization. International Journal of Advanced Networking and Applications, 19-23.

[4]. R. S. Alonso, I. T. Dante and M. C. Juan (2011)"SYLPH: A Platform for Integrating Heterogeneous Wireless Sensor Networks in Ambient Intelligence Systems" In: K. Curran, (eds) Pervasive and Ubiq.uitous Technology Innovations for Ambient Intelligence Environments, 2013, IGI Global, United States of America.

[5]. M. T. J, Portocarrero, F. C. Delicato, P. F. Pires, N. Gámez, L. Fuentes, D. Ludovino and P. Ferreira, “Autonomic Wireless Sensor Networks: A Systematic Literature Review,” *Journal of Sensors*, Vol. 2014, Article ID 782789, 13 pages, 2014. https://doi.org/10.1155/2014/782789.

[6]. R. S. Deshpande, S. M. Walke and P.D. Vyavahare, "Quality of Service in Wireless Sensor Networks: Issues and Challenges," *IOSR Journal of Computer Engineering (IOSR-JCE)*, Vol. 3, No. 33, pp. 45-52, 2007.

[7]. M. A. Hassan, Q. Vien, M. Aiash, "Software Defined Networking for Wireless Sensor Networks: A Survey," *Advances in Wireless Communications and Networks*, Vol. 3, No. 2, 2017, pp. 10-22. doi: 10.11648/j.awcn.20170302.11

[8]. K. Shah and M. Kumar, "Resource management in wireless sensor networks using collective intelligence," *2008 International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, Sydney, NSW, 2008, pp. 423-428. doi: 10.1109/ISSNIP.2008.4762025

[9]. K. Shah, M. Di Francesco and M. Kumar, "Distributed resource management in wireless sensor networks using reinforcement learning," *Wireless Networks*, Vol. 19. No. 705, 2013. https://doi.org/10.1007/s11276-012-0496-2

[10]. A.Perrig, J. Stankovic and D. Wagner, "Security in wireless sensor networks," Communications of the ACM - *Wireless sensor networks CACM*, Vol. 47, No. 6, pp. 53-57, June 2004.

[11]. G. Padmavathi and D. Shanmugapriya, "A Survey of Attacks, Security Mechanisms and Challenges in Wireless Sensor Networks," *International Journal of Computer Science and Information Security, IJCSIS*, Vol. 4, No. 1 & 2, August 2009.

[12]. G. Han, J. Jiang, L. Shu, J. Niu and H. Chao, "Management and applications of trust in Wireless Sensor Networks: A survey," *Journal of Computer and System Sciences*, Vol. 80, No. 3, 2014, pp. 602-617, https://doi.org/10.1016/j.jcss.2013.06.014.

[13]. I. Butun and R. Sankar, "A brief survey of access control in Wireless Sensor Networks," *2011 IEEE Consumer Communications and Networking Conference (CCNC)*, Las Vegas, NV, 2011, pp. 1118-1119. doi: 10.1109/CCNC.2011.5766345

[14]. A. de la Piedra, F. Benitez-Capistros, F. Dominguez and A. Touhafi, "Wireless sensor networks for environmental research: A survey on limitations and challenges," *Eurocon 2013*, Zagreb, 2013, pp. 267-274. doi: 10.1109/EUROCON.2013.6624996

[15]. I. E. Korbi, Y. Ghamri-Doudane, R. Jazi and L. A. Saidane, "Coverage-Connectivity based Fault Tolerance Procedure in Wireless Sensor Networks," In Proc. IEEE The

9th International Wireless Communications and Mobile Computing (IEEE IWCMC). Cagliari, Sardinia – Italy, pp. 1540-1545, Jul., 2013.

[16]. G. Gîrban and M. Popa, (2013) "A Brief Outline of Computational Limitations on Actual Wireless Sensor Networks." In: V. Balas, J. Fodor, A. Várkonyi-Kóczy, J. Dombi and L. Jain, (eds) *Soft Computing Applications. Advances in Intelligent Systems and Computing*, Vol. 195. Springer, Berlin, Heidelberg

[17]. V. Sharma, "Limitation Associated with Wireless Sensor Network," *International Journal of Computer Science and Technology (IJCST)*, Vol. 5, No.1, pp. 110- 112, First-quater 2014.

[18]. S. M. Nam and T. H. Cho, "Improvement of Energy Consumption and Detection Power for PVFS in Wireless Sensor Networks," In Proc. IEEE, Seventh International Conference on Mobile Computing and Ubiquitous Networking (ICMU), Singapore, pp.129-134, Jan. 2014.

[19]. P. Gyorke and B. Pataki, "Application of energy-harvesting in wireless sensor networks using predictive scheduling", In Proc. Instrumentation and Measurement Technology Conference (I2MTC), 2012 IEEE International, Graz, pp. 582 – 587,2012.

[20]. Q. Zao and Y. Nakamoto, "Routing Algorithms for Preventive Energy Holes and Improving Fault Tolerance in Wireless Sensor Networks," In Proc. IEEE Second International Symposium on Computing and Networking (CANDAR), Shizuoka, Japan, pp. 278-283, Dec. 2014.

[21]. A. Zeb, A.K.M. Muzahidul Islam, N. Mansoor, S. Baharun and S. Komaki, "Fault Tolerance in Dynamic Cluster-Based Wireless Sensor Networks," In Proc. IEEE 12th International Bhurban Conference on Applied Sciences & Technology (IBCAST). Islamabad, Pakistan, pp. 646-649, Jan. 2015.

[22]. M.Z. A. Bhuiyan, G. Wang, J. Cao and J. Wu, "Deploying Wireless Sensor Networks with Fault-Tolerance for Structural Health Monitoring," IEEE Transactions On Computers, Vol. 64, No. 2, pp. 382-395, Feb. 2015.

[23]. R. N. Duche and N. P. Sarwade. “Sensor Node Failure or Malfunctioning Detection in Wireless Sensor Networks.” ACEEE Int. J. on Communications, Vol. 03, No. 01, pp. 57-61, Mar. 2012.

[24]. M. Jammal, T. Singh, A. Shami, R. Asal and Y. Li. “Software-Defined Networking: State of the Art and Research Challenges”. *Journal of Computer Networks*, Vol. 72, pp. 74-98, Oct. 2014.

[25]. C. J. Bernados, A. De La Oliva, P. Serrano, A. Banchs, L. M. Contreras, J. Hao and J. C. Zuniga “An Architecture for Software Defined Wireless networking”, *IEEE Wireless Communications*, Vol. 21, No. 3, pg. 52-61, June. 2014.

[26]. M. A. Yitigel, O.D. Incel and C. Ersoy, "QoS-aware MAC protocols for wireless sensor networks: A survey", *Computer Networks*, Vol. 55, pg. 1982–2004, 2011.

[27]. K. Sukhkirandeep and N.M. Roohie, “Quality of Service in WSN-A Review”, *International Journal of Computer Applications*, Vol. 113, No. 18, pg. 42-46, March. 2015.

[28]. R. Jain, “introduction to Software Defined Networking (SDN)”, [online] available at: http://www.cse.wustl.edu/~jain/cse570-13/, (2013).

[29]. A.L. Valdivieso Caraguay, A.B. Peral, L.I. Barona López and L.J. García Villalba, " SDN: Evolution and Opportunities in the Development IoT Applications," *International Journal of Distributed Sensor Networks*, Vol. 2014, Article ID 735142, 10 pages, 2014, doi: 10.1155/2014/735142.

[30]. Babedi Betty Letswamotse, Reza Malekian, Chi-Yuan Chen, Kgotlaetsile Mathews Modieginyane, "Software Defined Wireless Sensor Networks: A Review on Efficient Resources, applications and technologies",Journal of Internet Technology, 2018.

[31]. Valdo Henriques, R. Malekian, "Mine Safety System Using Wireless Sensor Network", IEEE Access, Vol.4, pp. 3511-3521, 2016.

[32]. Johan Wannenburg, R. Malekian, “Body Sensor Network for Mobile Health Monitoring, a Diagnosis and Anticipating System”, IEEE Sensors Journal, Vol.15, No. 12, pp. 6839 - 6852 , 2015.

[33]. Arun Cyril Jose, R. Malekian, "Improving Smart Home Security; Integrating Logical Sensing into Smart Home", IEEE Sensors Journal, Vol. 17, Issue 3, pp. 4269-4286, 2017.

[34]. Zhongqin Wang, Ning Ye, R. Malekian, Fu Xiao, Ruchuan Wang, "TrackT: Accurate Tracking of RFID Tags with mm-level Accuracy Using first-order Taylor series approximation", AD Hoc Networks, Elsevier, Vol.53, pp.132-144, 2016.

[35]. Johan Wannenburg, R. Malekian, "Physical Activity Recognition from Smartphone Accelerometer Data for User Context Awareness Sensing", IEEE Transactions on Systems, Man and Cybernetics: Systems, Vol.47, no.12, pp. , 3142-3149, 2017.

[36]. Arnav Thakur, R. Malekian, Lakshmi Nair, Christian Fischer Pedersen, "A Sensor Based Peer to Peer Vehicle Data Sharing System, An Internet of Vehicles Approach, Journal of Internet Technology, In press.

[37]. Ning Ye, R. Malekian, Qiaomin Lin, and Ru-chuan Wang, "A Method for Driving Route Predictions Based on Hidden Markov Model", Mathematical Problems in Engineering, Vol.2015, pp.1-12, 2015.

[38]. Wan-Hee. Cho, J. Kim, and O. Song, "An Efficient Resource Management Protocol for Handling Small Resource in Wireless Sensor Networks," *International Journal of Distributed Sensor Networks*, Vol. 2013, Article ID 324632, 9 pages, 2013. doi:10.1155/2013/324632.

[39]. P. J. Del Cid, D. Hughes, J. Ueyama, S. Michiels, and W. Joosen, "DARMA: Adaptable Service and Resource Management for Wireless Sensor Networks", *MidSens'09*, pg. 1-6, 2009.

[40]. Open Networking Foundation, "Software-Defined Networking: The New Norm for Networks," ONF White Paper, ONF., Open Networking Foundation /2275 E. Bayshore Road, Suite 103 / Palo Alto, CA 94303, April 13, 2012. [Online]. Available: https://www.opennetworking.org/sdn-resources/openflow .

[41]. K.K. Yap, M. Kobayashi and D. Underhill," The Stanford OpenRoads deployment", *In ACM workshop on Wireless Network Testbeds, Experimentational Evaluation and Characterization* (WinTech), 2009.

[42]. W. Zhou, L. Li, M. Luo and W. Chou, "REST API Design Patterns for SDN Northbound API," *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, Victoria, BC, 2014, pp. 358-365. doi: 10.1109/WAINA.2014.153

[43]. RESTCONF Protocol. [Online]. Available: https://tools.ietf.org/html/rfc8040

[44]. S. G. Du, J. W. Lee and K. Kim, "Proposal of GRPC as a New Northbound API for Application Layer Communication Efficiency in SDN," Proceedings of the 12th International Conference on Ubiquitous Information Management and Communication, Langkawi, Malaysia, pp. 1- 6, January 2018.

[45]. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker and J. Turner, "OpenFlow: enabling innovation in campus networks," ACM SIGCOMM Computer Communication Review, Vol. 38, No. 2, pp. 69-74, April 2008. doi: 10.1145/1355734.1355746

[46]. Simple Network Management Protocol-IETF. [Online]. Available: https://www.ietf.org/rfc/rfc1157.txt.

[47]. Network Configuration Protocol (NETCONF). [Online]. Available: https://tools.ietf.org/html/rfc6241

[48]. The Open vSwitch Database Management Protocol. [Online]. Available: https://tools.ietf.org/html/rfc7047

[49]. A. Doria, J. H. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern, "Forwarding and Control Element Separation (ForCES)," March 2010.

[50]. A Border Gateway Protocol 4 (BGP-4). [Online]. Available: https://tools.ietf.org/html/rfc4271

[51]. OF-CONFIG 1.2- OpenFlow Management and Configuration Protocol. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config-1.2.pdf

[52]. PCEP Extension for Flow Specification draft-ietf-pce-pcep-flowspec-01. [Online]. Available: https://tools.ietf.org/html/draft-ietf-pce-pcep-flowspec-01

[53]. Extensible Messaging and Presence Protocol (XMPP): Core. [Online]. Available: https://xmpp.org/rfcs/rfc6120.html

[54]. T. M. C. Nguyen, D. B. Hoang and Z. Chaczko, "Can SDN Technology Be Transported to Software-Defined WSN/IoT?," 2016 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), Chengdu, 2016, pp. 234-239. doi:10.1109/iThings-GreenCom-CPSCom-SmartData.2016.63

[55]. Open Networking Foundation, "Software-Defined Networking: The New Norm for Networks," ONF White Paper, ONF., Open Networking Foundation /2275 E. Bayshore Road, Suite 103 / Palo Alto, CA 94303, 13 Apr., 2012. [Online]. Available: https://www.opennetworking.org/sdn-resources/openflow.

[56]. R. Jain, "introduction to Software Defined Networking (SDN)," available at: http://www.cse.wustl.edu/~jain/cse570-13/, (2013).

[57]. OpenFlow Switch Specification, Version 1.0.0 (Wire Protocol 0x01). [Online]. Available:https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2013/04/openflow-spec-v1.0.0.pdf

[58]. OpenFlow Switch Specification, Version 1.1.0 (Wire Protocol 0x02). [Online]. Available:https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-spec-v1.1.0.pdf

[59]. OpenFlow Switch Specification, Version 1.2 (Wire Protocol 0x03). [Online]. Available:https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-spec-v1.2.pdf

[60]. OpenFlow Switch Specification, Version 1.3.0 (Wire Protocol 0x04). [Online]. Availlable:https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf

[61]. OpenFlow Switch Specification, Version 1.3.1 (Wire Protocol 0x04). [Online]. Available:https://www.opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.3.1.pdf

[62]. OpenFlow Switch Specification, Version 1.3.2 (Wire Protocol 0x04). [Online]. Available:https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.2.pdf

[63]. OpenFlow Switch Specification, Version 1.3.3 (Wire Protocol 0x04). [Online]. Available:https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.3.pdf

[64]. OpenFlow Switch Specification, Version 1.4.0 (Wire Protocol 0x05). [Online]. Available:https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.4.0.pdf

[65]. OpenFlow Switch Specification, Version 1.4.1 (Wire Protocol 0x05). [Online]. Available:https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-switch-v1.4.1.pdf

[66]. OpenFlow Switch Specification, Version 1.5.0 (Wire Protocol 0x06). [Online]. Available:http://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.0.noipr.pdf

[67]. OpenFlow Switch Specification, Version 1.5.1 (Wire Protocol 0x06). [Online]. Available:https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

[68]. T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: a distributed control platform for large-scale production networks," *in Proceedings of the 9th USENIX conference on Operating systems design and implementation, ser. OSDI'10*. Berkeley, CA, USA: USENIX Association, pp. 1–6, 2010.

[69]. K. Ogawa, W. M. Wang, E. Haleplidis, and J. H. Salim, "ForCES Intra-NE High Availability," Internet Draft, Internet Engineering Task Force, October 2013. [Online]. Available: http://tools.ietf.org/id/draftietf-forces-ceha-08.txt

[70]. H. Yin, H. Xie, T. Tsou, D. Lopez, P. Aranda, and R. Sidi, "SDNi: A Message Exchange Protocol for Software Defined Networks (SDNS) across Multiple Domains," Internet Draft, Internet Engineering Task Force, June 2012. [Online]. Available: http://tools.ietf.org/id/draft-yinsdn-sdni-00.txt

[71]. F. A. Botelho, F. M. V. Ramos, D. Kreutz, and A. N. Bessani, "On the feasibility of a consistent and fault-tolerant data store for SDNs," *in Proceedings of the 2013 Second*

European Workshop on Software Defined Networks, ser. EWSDN '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 38–43.

[72]. A. Doria, J. H. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern, "Forwarding and Control Element Separation (ForCES) Protocol Specification," Internet Engineering Task Force, March 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5810.txt

[73]. B. Sonkoly, A. Gulyás, F. Németh, J. Czentye, K. Kurucs, B. Novak and G. Vaszkun, "On QoS Support To Ofelia and OpenFlow," *European workshop on Software Defined Networking (EWSDN)*, 2012, pp. 109-113.

[74]. H. E. Egilmez and A. M. Tekalp, "Distributed QoS Architectures For Multimedia Streaming Over Software Defined Networks, *IEEE Transactions on Multimedia*, Vol. 16, No. 6, October. 2014.

[75]. P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow and G. Parulkar, "ONOS: Towards an open, distributed SDN OS," in Proc. 3rd Workshop Hot Topics in Software Defined Networking, 2014, pp. 1–6.

[76]. OpenDayLight, "OpenDayLight: Open Source SDN Platform," 2017. Available at: https://www.opendaylight.org/

[77]. FloodLight, [Online] available at: http://www.projectfloodlight.org/floodlight/.

[78]. D. Zhao, M. Zhu and M. Xu, "Leveraging SDN and OpenFlow to Mitigate Interference in Enterprise WLAN", *Journal of Networks*, Vol. 9, No. 6, pp.1526-1533, June. 2014.

[79]. J. Lee, M. Uddin, J. Tourrilhes, S. Sen, S. Banerjee, M. Arndt, K.H Kim and T. Nadeem, "meSDN: Mobile Extension of SDN ", *MCS'14*, 2014, pp. 7-14.

[80]. Yan Jinyao, Zhang Hailong, Shuai Qianjun, Liu Bo and Guo Xiao. "HiQoS: An SDN-Based Multipath QoS Solution." *China Communications*, pp. 123-133, May 2015.

[81]. Y. Zhang, Y. Tang, D. Tang and W. Wang. "QOF: QoS Framework Based On OpenFlow." *2015 2nd International Conference on Information Science and Control Engineering*, pp. 380-387, 2015.

[82]. H. E. Egilmez, S. T. Dane, K. T. Bagci and A. M. Tekalp, "OpenQoS: An OpenFlow Controller Design for Multimedia Delivery with End-to-End Quality of Service over Software-Defined Networks", *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*, pp.1-8, Hollywood, California, December 2012.

[83]. N.S. Ko, H. Heo, J.D. Park and H.S. Park, "OpenQFlow: Scalable OpenFlow with Flow-Based QoS, *IEICE Transactions on communications*, Vol. E96-B, No. 2, pp. 497-488, February. 2013.

[84]. P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the Art of Virtualization," *In SOSP '03: Proceedings of the 19th ACM Symposium on Operating Systems Principles*, 2003, pp. 164–177.

[85]. A.L. Valdivieso Caraguay, J.A. Puuente Fernández and L.J. García Villalba, Framework for Optimized Multimedia Routing Over Software Defined Networks, *Computer Networks* (2015), Available at http://dx.doi.org/10.1016/j.comnet.2015.09.013.

[86]. R. Sherwood, G. Gibb, K.K. Yap, G. Appenzeller, M. Casado, N. McKeown, G. Parulkar, "FlowVisor: A Network Virtualization Layer," Deutsche Telekom Inc. R&D Lab, Stanford, CA OPENFLOW-TR-2009-1, October. 2009, [Online] Available at http://OpenFlowSwitch.org/downloads/technicalreports/openflow-tr-2009-1-flowvisor.pdf.

[87]. M. Yang, Y. Li, D. Jin, L. Su, S. Ma, and L. Zeng, "OpenRAN: A Software-defined RAN Architecture Via Virtualization," *in ACM SIGCOMM computer communication review*, Vol. 43, No. 4, pp. 549–550 , 2013.

[88]. B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "Elastic Tree: Saving Energy in Data Center Networks," *in NSDI*, Vol. 10, pp. 249–264, 2010.

[89]. K. M. Modieginyane, B. B. Letswamotse, R. Malekian and A. M. Abu-Mahfouz, "Software defined wireless sensor networks application opportunities for efficient network management: A survey", *Elsevier: Computers & Electrical Engineering*, Vol. 66, pp. 274-287, February 2017.

[90]. De Gante, M. Aslan and A. Matrawy, 2014, "Smart Wireless Sensor Network Management Based on Software-Defined Networking", *27th Biennal symposium on Communications (QBSC)*, pg. 71-75.

[91]. T. Luo, H. Tan and T.Q.S Quek, "Sensor OpenFlow: Enabling Software-Defined Wireless Sensor Networks", *IEEE Communication Letters*, Vol. 16, No. 11, pp. 1896-1899, Nov. 2012.

[92]. R. C. A. Alves, D. A. G. Oliveira, G. A. Nunes, and C. B. Margi, "IT-SDN: Improved architecture for SDWSN," *in XXXIV Simposio ´ Brasileiro de Redes de Computadores*, 2017, http://www.larc.usp.br/users/cbmargi/www//it-sdn/it-sdn-sbrc2017sf.pdf

[93]. A.S. Yuan, H. Fang and Q. Wu, 2014, "OpenFlow based hybrid routing in Wireless Sensor Networks", *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP) Symposium on Sensing, Propagation, and Wireless Networks for Healthcare Applications*, pp. 1-5.

[94]. Y. Li and H. Huang, "the design and experiment of a software-defined acoustic modem for underwater sensor network", *2010 IEEE Oceans*, 2010, pg. 1-4.

[95]. P. Levis S. Madden J. Polastre R. Szewczyk K. Whitehouse A. Woo D. Gay J. Hill M. Welsh E. Brewer and D. Culler, "TinyOS: An Operating System for Sensor Networks," *In: Weber W., Rabaey J.M., Aarts E. (eds) Ambient Intelligence. Springer*, Berlin, Heidelberg, 2005.

[96]. B. T. de Oliveira and C. B. Margi, "TinySDN: Enabling TinyOS to Software-Defined Wireless Sensor Networks," [Online] available: http://sbrc2016.ufba.br/downloads/Salao_Ferramentas/154318.pdf

[97]. B. T. de Oliveira and C. B. Margi, "Distributed control plane architecture for software-defined Wireless Sensor Networks," *2016 IEEE International Symposium on Consumer Electronics (ISCE)*, Sao Paulo, 2016, pp. 85-86. doi: 10.1109/ISCE.2016.7797384

[98]. C. Cao, L. Luo, Y. Gao, W. Dong and C. Chen, "TinySDM: Software Defined Measurement in Wireless Sensor Networks," *2016 15th ACM/IEEE International*

Conference on Information Processing in Sensor Networks (IPSN), Vienna, 2016, pp. 1-12. doi: 10.1109/IPSN.2016.7460723

[99]. Z. Lan, W. Ma, W. Xia, L. Shen, F. Yan and L. Ren, "Design and implementation of flow-based programmable nodes in software-defined sensor networks," *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, Chengdu, 2017, pp. 734-738. doi: 10.1109/CompComm.2017.8322640

[100]. W. Wang, Y. Chen, Q. Zhang and T. Jiang, 2016, "A software-defined wireless networking enabled spectrum management architecture," *in IEEE Communications Magazine*, Vol. 54, No. 1, pp. 33-39. doi: 10.1109/MCOM.2016.7378423.

[101]. H. Akram and A. Gokhale, "Rethinking the Design of LR-WPAN IoT Systems with Software-Defined Networking," *2016 International Conference on Distributed Computing in Sensor Systems (DCOSS)*, Washington, DC, 2016, pp. 238-243. doi: 10.1109/DCOSS.2016.16

[102]. M. A. Abdala, S.B. Younis, and S. A. Jaseem, "Software-Defined Networking for Wireless Sensor Networks," *Proceedings of Al-Salam University College First Scientific Conference SFSC'17*, Baghdad, Iraq, pp. 513-524, 16-17 April 2017.

[103]. P. Jayashreel and F. Infant Princy, 26 - 28 Mar 2015,"Leveraging SDN to Conserve Energy in WSN-An Analysis." *In: 2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*, Chennai, India, pp.1-6.

[104]. W. Xiang, N. Wang, and Y. Zhou, 2016, "An Energy-efficient Routing Algorithm for Software-defined Wireless Sensor Networks." *IEEE Sensors Journal*, pp. 1-8.

[105]. Y. Wang, H. Chen, X. Wu and L. Shu, 2016, "An Energy-Efficient SDN Based Sleep Scheduling Algorithm for WSNs." *Elsevier: Journal of Network and Computer Applications*, Vol. 59, pp. 39-45.

[106]. D. Zeng, P. Li, S. Guo and T. Miyazaki, "Minimum-energy reprogramming with guaranteed quality-of-sensing in software-defined sensor networks," *2014 IEEE International Conference on Communications (ICC)*, Sydney, NSW, 2014, pp. 288-293. doi: 10.1109/ICC.2014.6883333

[107]. W. Ejaz, M. Naeem, M. Basharat, A. Anpalagan, and S. Kandeepan, 2016, “Efficient Wireless Power Transfer in Software-Defined Wireless Sensor Networks.” *IEEE Sensors Journal*, pp. 1-12.

[108]. F. Olivier, G. Carlos and N. Florent, “SDN Based Architecture for Clustered WSN,” *2015 9th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, Blumenau, 2015, pp. 342-347. doi: 10.1109/IMIS.2015.52

[109]. D. Zeng, P. Li, S. Guo, T. Miyazaki, J. Hu and Y. Xiang, "Energy Minimization in Multi-Task Software-Defined Sensor Networks," *in IEEE Transactions on Computers*, Vol. 64, No. 11, pp. 3128-3139, Nov. 1 2015. Doi: 10.1109/TC.2015.2389802

[110]. Junli F, Yawen W and Haibin S, “An improved energy-efficient routing algorithm in software define wireless sensor network,” *In: 2017 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, 2017, pp 1-5. doi: 10.1109/ICSPCC.2017.8242610

[111]. D. Jian, X. Chunxiu, W. Muqing and L. Wenxing, “Design and implementation of a novel software-defined wireless sensor network,” *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, Chengdu, 2017, pp. 729-733. doi: 10.1109/CompComm.2017.8322639

[112]. S. Luo, H. Wang, J. Wu, J. Li, L. Guo and B. Pei, 15-18 May 2016, "Improving Energy Efficiency in Industrial Wireless Sensor Networks Using SDN and NFV." *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, pp. 1-5.

[113]. Q. Xu and J. Zhao, "A WSN Architecture Based on SDN," 4th International Conference on Information Systems and Computing Technology (ISCT 2016), Advances in Computer Science Research, vol. 64, pp. 159-163, 2016.

[114]. M. Mukherjee, L. Shu, T. Zhao, K. Li and H. Wang, "Low Control Overhead-Based Sleep Scheduling in Software-Defined Wireless Sensor Networks," *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, Sydney, NSW, 2016, pp. 1236-1237. doi: 10.1109/HPCC-SmartCity-DSS.2016.0174

[115]. L. Shu, M. Hauswirth, Han-Chieh Chao, M. Chen and Y. Zhang, "NetTopo: A framework of simulation and visualization for wireless sensor networks," *Ad Hoc Networks*, Vol. 9, No. 5, 2011, pp. 799-820, ISSN 1570-8705, https://doi.org/10.1016/j.adhoc.2010.09.003

[116]. P. Jayashreel and F. Infant Princy, 26 - 28 March 2015,"Leveraging SDN to Conserve Energy in WSN-An Analysis." *In: 2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*, Chennai, India, pp.1-6.

[117]. R. Huang, X. Chu, J. Zhang, and Y. Hen Hu, 2015, "Energy-Efficient Monitoring in Software Defined Wireless Sensor Networks Using Reinforcement Learning: A Prototype," *International Journal of Distributed Sensor Networks*, Vol. 2015, Article ID 360428, 12 pages. Doi:10.1155/2015/360428.

[118]. J. Wang, Y. Miao, P. Zhou, M. S. Hossain, and S. M. M. Rahman. 2017. A Software Defined Network Routing In Wireless Multihop Network. *Elsevier: Journal of Network and Computer Applications* 85, C (May 2017), 76-83. Doi: https://doi.org/10.1016/j.jnca.2016.12.007

[119]. K.M. Modieginyane, R. Malekian and B.B. Letswamotse, "Flexible Network Management and Application Service Adaptability in Software Defined Wireless Sensor Networks", *Journal of Ambient Intelligence and Humanized Computing*, (2018), Vol. 9, No. 44, March 2018. https://doi.org/10.1007/s12652-018-0766-7

[120]. G. Li et al., "Traffic Load Minimization in Software Defined Wireless Sensor Networks," *in IEEE Internet of Things Journal*. doi: 10.1109/JIOT.2018.2797906

[121]. Y. Wei, W. Muqing, L. Wenxing and Z. Min, "The design of load-balance based routing algorithm in software defined wireless sensor networks," *2017 IEEE/CIC International Conference on Communications in China (ICCC)*, Qingdao, 2017, pp. 1-6. doi: 10.1109/ICCChina.2017.8330522

[122]. I. T. Haque and N. Abu-Ghazaleh, "Wireless Software Defined Networking: A Survey and Taxonomy," *in IEEE Communications Surveys & Tutorials*, Vol. 18, No. 4, pp. 2713-2737, Fourthquarter 2016. Doi: 10.1109/COMST.2016.2571118

[123]. F. Junior and F. Matos, "SDN-Based Approach to Select Allocation Strategies in Heterogeneous Wireless Sensor Networks," *2015 Brazilian Symposium on Computing Systems Engineering (SBESC)*, Foz do Iguacu, 2015, pp. 25-29. Doi: 10.1109/SBESC.2015.12

[124]. S. Tomovic and I. Radusinovic, "Allocation algorithm for handling multiple applications in software-defined WSN," *2016 24th Telecommunications Forum (TELFOR)*, Belgrade, 2016, pp. 1-4. Doi: 10.1109/TELFOR.2016.7818740

[125]. Y. Zhang, Y. Zhu, F. Yan, Z. Li and L. Shen, "Semidefinite Programming Based Resource Allocation For Energy Consumption Minimization In Software Defined Wireless Sensor Networks," *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Valencia, 2016, pp. 1-6. doi: 10.1109/PIMRC.2016.7794902

[126]. M. M. Hassan; A. Alsanad, "Resource Provisioning for Cloud-Assisted Software Defined Wireless Sensor Network", *IEEE Sensors Journal*, Vol: PP, No. 99, pp. 1-8, June. 2016.

[127]. R. Sayyed, S. Kundu, C. Warty and S. Nema, 2014, "Resource optimization using Software Defined Networking for Smart Grid Wireless Sensor Network." *In: 2014 3rd International Conference on Eco-friendly Computing and Communication Systems (ICECCS)*, Mangalore pp. 200-205.

[128]. S. Namal, I. Ahmad, A. Gurtov and M. Ylianttila, "SDN Based Inter-Technology Load Balancing Leveraged by Flow Admission Control," *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, Trento, 2013, pp. 1-5. doi: 10.1109/SDN4FNS.2013.670255.

[129]. Mukherjee M, Shu L, Zhao T, Wang D, Wang H (2017) Lightweight flow management for software-defined wireless sensor networks with link fault in data plane. *In: 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp 998-999. doi: 10.1109/INFCOMW.2017.8116529

[130]. Miguel MLF, Penna MC, Jamhour E, Pellenz E (2017) A CoAP based control plane for Software Defined Wireless Sensor Networks. *Journal of Communications and Networks*, Vol. 19, No. 6, pp. 555-562. doi: 10.1109/JCN.2017.000095

[131]. Misra S, Bera S, Achuthananda MP, Pal SK, Obaidat MS (2017) Situation-aware protocol switching in Software-Defined Wireless Sensor Network systems. *IEEE Systems Journal*, pp (99):1-8. doi: 10.1109/JSYST.2017.2774284

[132]. Y. Wei, W. Muqing, L. Wenxing and Z. Min, "The design of load-balance based routing algorithm in software defined wireless sensor networks," *2017 IEEE/CIC International Conference on Communications in China (ICCC)*, Qingdao, 2017, pp. 1-6. doi: 10.1109/ICCChina.2017.8330522

[133]. A. G. Anadiotis, L. Galluccio, S. Milardo, G. Morabito and S. Palazzo, "SD-WISE: A Software-Defined WIreless SEnsor network," *eprint arXiv: 1710.09147*, 2017.

[134]. O. Flauzac, C. Gonzalez, F. Nolot "Developing a Distributed Software Defined Networking Testbed for IoT," *Procedia: Computer Science*, 83 (2016), pp. 680-684.

[135]. L. F. d. S. Santos, F. F. d. Mendonça and K. L. Dias, "µSDN: An SDN-Based Routing Architecture for Wireless Sensor Networks," *2017 VII Brazilian Symposium on Computing Systems Engineering (SBESC)*, Curitiba, 2017, pp. 63-70. doi: 10.1109/SBESC.2017.15

[136]. Y. Zhu, Y. Zhang, W. Xia and L. Shen, "A Software-Defined Network Based Node Selection Algorithm in WSN Localization." *In 2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, Nanjing, pp. 1-5, 15-18 May 2016.

[137]. A.S. Yuan, H. Fang and Q. Wu, 2014, "OpenFlow based hybrid routing in Wireless Sensor Networks", *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP) Symposium on Sensing, Propagation, and Wireless Networks for Healthcare Applications*, pp. 1-5.

[138]. Q. Xu and J. Zhao, "A WSN Architecture Based on SDN," 4*th International Conference on Information Systems and Computing Technology (ISCT 2016), Advances in Computer Science Research*, Vol. 64, pp. 159-163, 2016.

[139]. L. Han, S. Sun, B. Joo, X. Jin, S. Han, "QoS-aware routing mechanism in OpenFlow-enabled wireless multimedia sensor networks", *International Journal of Distributed Sensor Networks*, Vol. 8, No. 11, pp. 1-18, 2012.

[140]. S. Tomovic and I. Radusinovic, 2016, "Extending the lifetime of wireless sensor network with partial SDN deployment." *Telfor Journal*, Vol. 8, No. 1, pp. 8-13.

[141]. P. Di Dio, S. Faraci, L. Galluccioz, S. Milardo, G. Morabito, S. Palazzo, and P. Livreriy, 2016, "Exploiting State Information to Support QoS in Software-Defined WSNs." *MedHocNet*, 2016.

[142]. L. Galluccioz, S. Milardo, G. Morabito and S. Palazzo, 26 April-1 May 2015, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks." *2015 IEEE Conference on Computer Communications (INFOCOM)*, Hong Kong, pp. 513-521.

[143]. M. A. Jan, S. R. U. Jan, M. Alam, A. Akhunzada and I. Ur Rahman, "A Comprehensive Analysis of Congestion Control Protocols in Wireless Sensor Networks," *Springer: Mobile Networks and Applications*. Vol. 23, No. 456, doi: https://doi.org/10.1007/s11036-018-1018-y

[144]. . Ghanavati, J. Abawajy and D. Izadi, "A fuzzy technique to control congestion in WSN," *The 2013 International Joint Conference on Neural Networks (IJCNN)*, Dallas, TX, 2013, pp. 1-5. doi: 10.1109/IJCNN.2013.6706958

[145]. Ghaffari, "Congestion control mechanisms in wireless sensor networks: A survey," *Elsevier: Journal of Network and Computer Applications*, Vol. 52, pp. 101-115, 2015. doi: https://doi.org/10.1016/j.jnca.2015.03.002

[146]. H. Yuan, N. Yugang and G. Fenghao, "Congestion control for wireless sensor networks: A survey," *The 26th Chinese Control and Decision Conference (2014 CCDC)*, Changsha, 2014, pp. 4853-4858. doi: 10.1109/CCDC.2014.6853042

[147]. Philip Levis and David E. Gay. TinyOS, page 55. *Cambridge University Press*, 2009.

[148]. MEMSIC, TelosB Mote, TPR2420 Platform.

[149]. H. Fotouhi, M. Vahabi, A. Ray and M. Björkman, "SDN-TAP: An SDN-based traffic aware protocol for wireless sensor networks," *2016 IEEE 18th International*

Conference on e-Health Networking, Applications and Services (Healthcom), Munich, 2016, pp. 1-6. doi: 10.1109/HealthCom.2016.7749527

[150]. H. Silva, A. Hahn Pereira, Y. Solano, B. T. de Oliveira, and C. B. Margi, "WARM: WSN Application Development and Resource Management," *in XXXIV Simposio Brasileiro de Telecomunicacoes e Processamento de Sinais (SBrT 2016) (SBrT 2016)*, Santar´em, Brazil, August 2016.

[151]. N. McKeown, T. Anderson, H. Balakrishnan, et al., "OpenFlow: enabling innovation in campus networks," ACM SIGCOMM Computer Communication Review,2008, 38, (2), pp. 69-74.

[152]. P. Berde, M. Gerola, J. Hart, et al., "ONOS: Towards an Open, Distributed SDN OS," *Proceedings of the 3rd Workshop Hot Topics in Software Defined Networking*, 2014, pp. 1–6.

[153]. "Ns3 official website." [Online]. Available: www.nsnam.org, Accessed: 12 Jan 2017.

[154]. L. J. Chaves, I. C. Garcia, and E. R. M. Madeira, "OFSwitch13: Enhancing ns-3 with OpenFlow 1.3support," *Proceedings of the 8th Workshop on ns-3 (WNS3)*, 2016, pp. 33-40.

[155]. H. Choi, N. Kim and H. Cha, "6LoWPAN-SNMP: Simple Network Management Protocol for 6LoWPAN," *Proceedings of 2009 11th IEEE International Conference on High Performance Computing and Communications*, Seoul, 2009, pp. 305- 313.

[156]. Deze Zeng, Toshiaki Miyazaki, Song Guo, Tsuneo Tsukahara, Junji Kitamichi and Takafumi Hayashi, "Evolution of Software-Defined Sensor Networks", *2013 IEEE 9th Intern[160ational Conference on Mobile Ad-hoc and Sensor Networks*, 2013, pp.410-413.

[157]. IETF Recommendations Regarding Active Queue Management, [Online] available at: https://tools.ietf.org/html/rfc7567.

[158]. RESTCONF Protocol, [Online] available at: https://tools.ietf.org/html/rfc8040.

[159]. YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF), [Online] available at: https://tools.ietf.org/html/rfc6020.

[160]. J. Kang, Y. Zhang and B. Nath, "TARA: Topology-Aware Resource Adaptation to Alleviate Congestion in Sensor Networks," *in IEEE Transactions on Parallel and Distributed Systems*, Vol. 18, No. 7, pp. 919-931, July 2007. doi: 10.1109/TPDS.2007.1030

[161]. C. Sergiou and V. Vassiliou, "DAlPaS: A performance aware congestion control algorithm in Wireless Sensor Networks," *2011 18th International Conference on Telecommunications*, Ayia Napa, 2011, pp. 167-173. doi: 10.1109/CTS.2011.5898912

ADDENDUM A CODE SNIPETTS

#include <fstream> #include "ns3/core-module.h" #include "ns3/internet-module.h" #include "ns3/sixlowpan-module.h" #include "ns3/lr-wpan-module.h" #include "ns3/applications-module.h" #include "ns3/mobility-module.h" #include "ns3/netanim-module.h" #include "ns3/core-module.h" #include "ns3/network-module.h" #include "ns3/ipv6-static-routing-helper.h" #include "ns3/ipv6-routing-table-entry.h" #include "ns3/csma-module.h" #include "ns3/applications-module.h" #include "ns3/config-store.h" #include <cmath> #include "ns3/gnuplot.h" #include <string> #include <cassert> #include <stdio.h> #include <stdlib.h> #include "ns3/basic-energy-source.h" #include "ns3/simple-device-energy-model.h" #include <ns3/ofswitch13-module.h> #include <ns3/internet-apps-module.h> #include <ns3/tap-bridge-module.h>	**Header Files**
using namespace ns3; NS_LOG_COMPONENT_DEFINE("Remotely_Controlled_SDWSN");	**Declaring variables**

int
main (int argc, char *argv[])
{
uint16_t NumberOfNodes;
uint16_t NumberOfBaseStationNodes;
double SimTime = 60;
double distance = 150.0;
double InterPacketInterval = 150;
bool verbose = false;	

CommandLine cmd; cmd.AddValue ("simTime", "Simulation time (seconds)", simTime); cmd.AddValue ("verbose", "Enable verbose output", verbose); cmd.AddValue ("trace", "Enable datapath stats and pcap traces", trace); cmd.Addvalue ("Sensor_Nodes", "Number_Of_Sensor_Nodes", Sensor_Nodes); cmd.Addvalue ("SimTime", "Total duration of the simulation [s]", SimTime); cmd.Addvalue ("distance", "distance between ends[m]", distance); cmd.AddValue ("nPackets", "Number of packets to echo", nPackets); cmd.Addvalue ("InterPacketInterval", "Inter packet interval[ns]",InterPacketInterval); cmd.Parse (argc, argv);	**Command Line**

if (verbose)
{
 	
LogComponentEnable ("Remotely_Controlled_SDWSN", LOG_LEVEL_INFO);
LogComponentEnable ("Ipv6EndPointDemux", LOG_LEVEL_ALL);
LogComponentEnable ("Ipv6L3Protocol", LOG_LEVEL_ALL);
LogComponentEnable ("Ipv6StaticRouting", LOG_LEVEL_ALL);
LogComponentEnable ("Ipv6ListRouting", LOG_LEVEL_ALL);
LogComponentEnable ("Ipv6Interface", LOG_LEVEL_ALL);
LogComponentEnable ("Icmpv6L4Protocol", LOG_LEVEL_ALL);
LogComponentEnable ("Ping6Application", LOG_LEVEL_ALL);
LogComponentEnable ("NdiscCache", LOG_LEVEL_ALL);
LogComponentEnable ("SixLowPanNetDevice", LOG_LEVEL_ALL);
 	
OFSwitch13Helper::EnableDatapathLogs ();
LogComponentEnable ("OFSwitch13Interface", LOG_LEVEL_ALL);
LogComponentEnable ("OFSwitch13Device", LOG_LEVEL_ALL);
LogComponentEnable ("OFSwitch13Port", LOG_LEVEL_ALL);
LogComponentEnable ("OFSwitch13Queue", LOG_LEVEL_ALL);
LogComponentEnable ("OFSwitch13SocketHandler", LOG_LEVEL_ALL);
LogComponentEnable ("OFSwitch13Controller", LOG_LEVEL_ALL);
LogComponentEnable ("OFSwitch13LearningController", LOG_LEVEL_ALL);
LogComponentEnable ("OFSwitch13Helper", LOG_LEVEL_ALL);
LogComponentEnable ("OFSwitch13ExternalHelper", LOG_LEVEL_ALL);
}
	Enabling Log Components

NS_LOG_INFO ("Create Sensor nodes"); NodeContainer Sensor_Nodes; Sensor_Nodes.Create (100);	**Creating the Wireless**

NS_LOG_INFO ("Create Base Stations"); NodeContainer Base_Stations; Base_Stations.Create (4);	**Sensor Network**
CsmaHelper csmaHelper; csmaHelper.SetChannelAttribute ("DataRate", DataRateValue (DataRate ("100Mbps"))); csmaHelper.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));	**Connect Sensor Nodes and Base stations**
Mobility Helper mobility; mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel"); Ptr<ListPositionAllocator> Sensor_NodesPositionAlloc = CreateObject<ListPositionAllocator> (); Sensor_NodesPositionAlloc->Add (Vector (0.0, 0.0, 0.0)); mobility.SetPositionAllocator (Sensor_NodesPositionAlloc); mobility.Install (Sensor_Nodes); Ptr<ListPositionAllocator> Base_StationsPositionAlloc = CreateObject<ListPositionAllocator> (); Base_StationsPositionAlloc->Add (Vector (50.0, 0.0, 0.0)); mobility.SetPositionAllocator (Base_StationsPositionAlloc); mobility.Install (Base_Stations);	**Installing Mobility**
LrWpanHelper lrWpanHelper; NetDeviceContainer devContainer = lrWpanHelper.Install(Sensor_Nodes); lrWpanHelper.AssociateToPan (devContainer, 100); LrWpanHelper lrWpanHelper; NetDeviceContainer devContainer = lrWpanHelper.Install(Base_Stations); lrWpanHelper.AssociateToPan (devContainer, 4);	**Creating Channels**
InternetStackHelper internetv6; internetv6.SetIpv6StackInstall (false); internetv6.Install (Sensor_Nodes); internetv6.Install (Base_Stations);	**Installing IPV6 Stack**
NS_LOG_INFO ("Install 6LoWPAN."); SixLowPanHelper sixlowpan; NetDeviceContainer six1 = sixlowpan.Install (devContainer);	**Installing 6LoWPAN Layer**
NS_LOG_INFO ("Assign addresses."); Ipv6AddressHelper ipv6; ipv6.SetBase (Ipv6Address ("2001:1::"), Ipv6Prefix (64)); Ipv6InterfaceContainer i = ipv6.Assign (six1);	**Assigning Addresses**
Ptr<Node> controllerNode = CreateObject<Node> ();	**Creating the Controller Node**
Ptr<OFSwitch13ExternalHelper> of13Helper = CreateObject<OFSwitch13ExternalHelper> (); Ptr<NetDevice> ctrlDev = of13Helper->InstallExternalController (controllerNode);	**Configuring the OpenFlow**

of13Helper->InstallBase_Stations (Base_Stations.Get (0)); of13Helper->InstallBase_Stations (Base_Stations.Get (1)); of13Helper->InstallBase_Stations (Base_Stations.Get (2)); of13Helper->InstallBase_Stations (Base_Stations.Get (3)); of13Helper->CreateOpenFlowChannels ();	**network domain using a remote controller**
TapBridgeHelper tapBridge; tapBridge.SetAttribute ("Mode", StringValue ("ConfigureLocal")); tapBridge.SetAttribute ("DeviceName", StringValue ("ctrl")); tapBridge.Install (ControllerNode, ctrlDev);	**TapBridge the controller node to the local machine using port 6653 for connection**

for (uint32_t i = 0; i < sensorNodes.GetN (); ++i)
{
anim.UpdateNodeDescription (Sensor_Nodes.Get (i), "Sensor_Nodes");
anim.UpdateNodeColor (Sensor_Nodes.Get (i),0, 255, 0);
}
 	
for (uint32_t i = 0; i < Base_Stations.GetN (); ++i)
{
anim.UpdateNodeDescription (Base_Stations.Get(i), "Base_Stations");
anim.UpdateNodeColor (Base_Stations.Get (i),0, 0,255);
}
	Setup the animation

NS_LOG_INFO ("Run Simulation."); Simulator::Run (); Simulator::Destroy (); NS_LOG_INFO ("Done."); }	**Termination of simulation**

ADDENDUM B ONOS FRAMEWORK

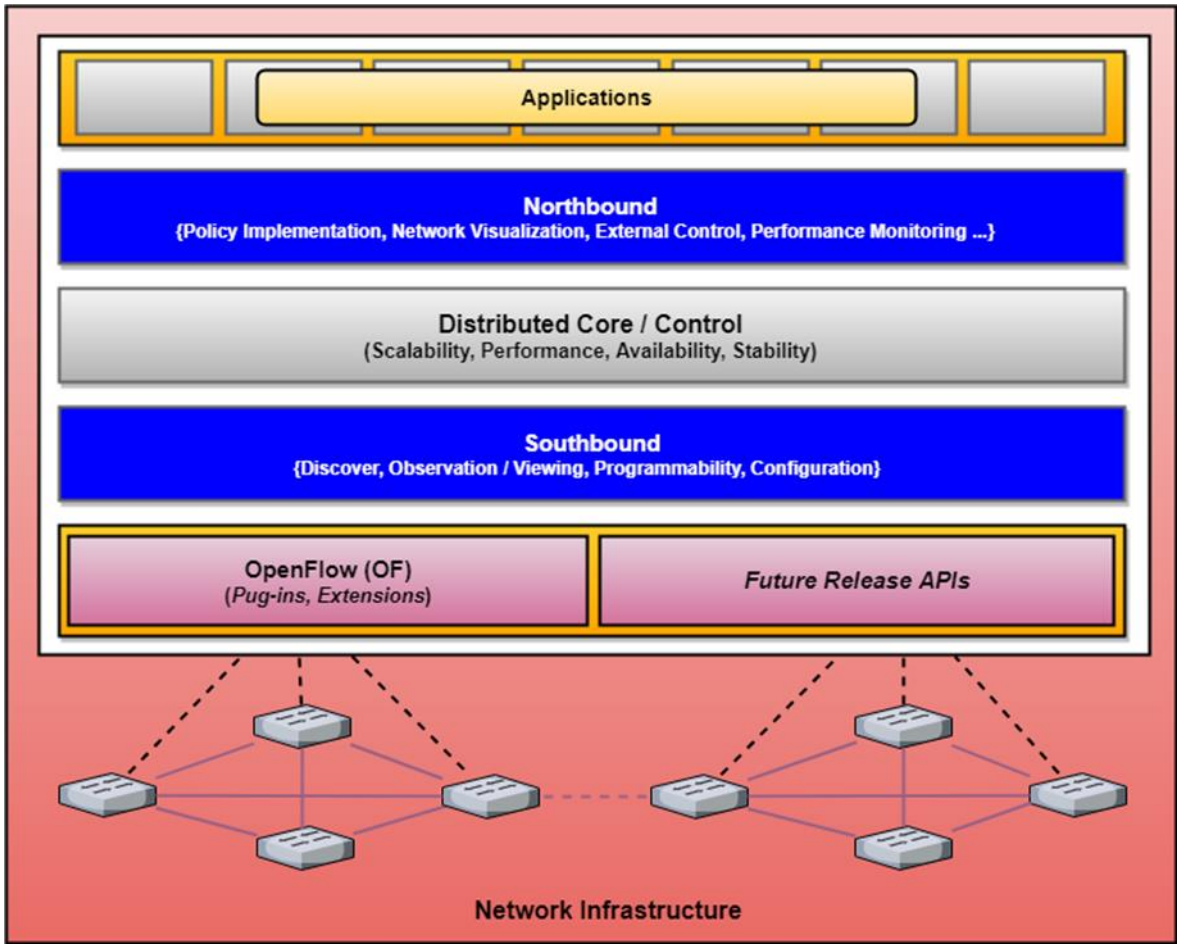

Figure B.1: ONOS Framework.