

**FLEXIBLE NETWORK MANAGEMENT IN SOFTWARE DEFINED  
WIRELESS SENSOR NETWORKS FOR MONITORING  
APPLICATION SYSTEMS**

By

**Kgotlaetsile Mathews Modieginwane**

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

In the

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

UNIVERSITY OF PRETORIA

June 2018

## SUMMARY

---

### **FLEXIBLE NETWORK MANAGEMENT IN SOFTWARE DEFINED WIRELESS SENSOR NETWORKS FOR MONITORING APPLICATION SYSTEMS**

By

**Kgotlaetsile Mathews Modieginyane**

Supervisor: Prof. R. Malekian

Department: Electrical, Electronic and Computer Engineering

University: University of Pretoria

Degree: Philosophiae Doctor in Engineering (Electronics)

**Keywords:** Wireless Sensor Networks, Software Defined Networking, Software Defined Wireless Sensor Networks, Internet of Things, Quality of Service, Artificial Intelligence, Machine Learning, Discrete Event Simulation, Resource Management, Network Function Virtualisation.

Wireless Sensor Networks (WSNs) are the commonly applied information technologies of modern networking and computing platforms for application-specific systems. Today's network computing applications are faced with high demand of reliable and powerful network functionalities. Hence, efficient network performance is central to the entire ecosystem, more especially where human life is a concern. However, effective management of WSNs remains a challenge due to problems supplemental to them. As a result, WSNs application systems such as in monitored environments, surveillance, aeronautics, medicine, processing and control, tend to suffer in terms of capacity to support compute intensive services due to limitations experienced on them. A recent technology shift proposes Software Defined Networking (SDN) for improving computing networks as well as enhancing network resource management, especially for life guarding systems. As an optimization strategy, a software-oriented approach for WSNs, known as Software Defined Wireless

Sensor Network (SDWSN) is implemented to evolve, enhance and provide computing capacity to these resource constrained technologies.

Software developmental strategies are applied with the focus to ensure efficient network management, introduce network flexibility and advance network innovation towards the maximum operation potential for WSNs application systems. The need to develop WSNs application systems which are powerful and scalable has grown tremendously due to their simplicity in implementation and application. Their nature of design serves as a potential direction for the much anticipated and resource abundant IoT networks. Information systems such as data analytics, shared computing resources, control systems, big data support, visualizations, system audits, artificial intelligence (AI), etc. are a necessity to everyday life of consumers. Such systems can greatly benefit from the SDN programmability strategy, in terms of improving how data is mined, analysed and committed to other parts of the system for greater functionality. This work proposes and implements SDN strategies for enhancing WSNs application systems especially for life critical systems. It also highlights implementation considerations for designing powerful WSNs application systems by focusing on system critical aspects that should not be disregarded when planning to improve core network functionalities.

Due to their inherent challenges, WSN application systems lack robustness, reliability and scalability to support high computing demands. Anticipated systems must have greater capabilities to ubiquitously support many applications with flexible resources that can be easily accessed. To achieve this, such systems must incorporate powerful strategies for efficient data aggregation, query computations, communication and information presentation. The notion of applying machine learning methods to WSN systems is fairly new, though carries the potential to enhance WSN application technologies. This technological direction seeks to bring intelligent functionalities to WSN systems given the characteristics of wireless sensor nodes in terms of cooperative data transmission. With these technological aspects, a technical study is therefore conducted with a focus on WSN application systems as to how SDN strategies coupled with machine learning methods, can contribute with viable solutions on monitoring application systems to support and provide various applications and services with greater performance. To realize this, this work further proposes and implements machine learning (ML) methods coupled with SDN strategies to; enhance sensor data aggregation, introduce network flexibility, improve resource

management, query processing and sensor information presentation. Hence, this work directly contributes to SDWSN strategies for monitoring application systems.

---

## **DEDICATION**

*I, dedicate this work to the rest of my family, including those who have passed on.*

*I duly dedicate this work especially to my*

*Father- Tshukudu William Modieginyane and Mother- Seirwang Jane Modieginyane.*

## ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor Prof. Reza Malekian for his utmost support throughout this work. His patience and technical advice has played a critical role in my work. Thank you Prof. Reza Malekian for encouraging me to work hard. Your efforts towards the greatness of my work are priceless.

I would also like to thank the University of Pretoria and the Department of Electrical, Electronic and Computer Engineering for their research support and provision of technical equipment to achieve my 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 permit my dearest appreciation to Babedi Betty Letswamotse (my spouse and research partner) for her substantial encouragement and support throughout my work and my life. Thank you for being there for me throughout. I cannot leave out my friends who have always been there for me throughout the challenges of life. You are all closer to my heart, and I thank you.

It would not have been possible for me to achieve this without the blessings from GOD and all the support from the kindest people He has placed around me, to only some of whom it is possible to give particular mention here. Above all, I would like to thank my father- *Tshukudu William Modieginyane* and mother- *Seirwang Jane Modieginyane*, brothers and sisters, who have given me their unequivocal support throughout the years of my academic life, as always, for which my mere expression of thanks likewise does not suffice.

Indeed;

“It is, Good to Wait Quietly for The Salvation of The LORD”. [ Lamentations 3:26 ]

## **LIST OF ABBREVIATIONS**

<b>AC</b>	Ant Colony
<b>AI</b>	Artificial Intelligence
<b>AODV</b>	Ad hoc On-Demand Vector
<b>API</b>	Application Programming Interface
<b>BC</b>	Black Circles
<b>BGP</b>	Border Gateway Protocol
<b>BIER</b>	Bit Index Explicit Replication
<b>BS</b>	Base Stations
<b>CDF</b>	Cumulative Distribution Function
<b>CEOG</b>	Cooperative Energy-aware Opportunistic Game
<b>CoAP</b>	Constrained Application Protocol
<b>COPS</b>	Common Open Policy Service Protocol
<b>COSPA</b>	Controller-based Open Status Present Algorithm
<b>CRLB</b>	Crameo-Rao Lower Bound
<b>CSDWSN</b>	Cloud-assisted Software-Defined Wireless Sensor Networking
<b>DA</b>	Detection Accuracy
<b>DB</b>	Distribution Board

<b>ECCKN</b>	Energy-Consumption-based Connected K-Neighbourhood
<b>FEC</b>	Forward Error Correction
<b>FPGA</b>	Field Programmable Gate Array
<b>FPR</b>	False Positive Rate
<b>FTP</b>	File Transfer Protocol
<b>GD</b>	Green Diamond
<b>GEAR</b>	Geographic and Energy Aware Routing
<b>GMM</b>	Gaussian Mixture Model
<b>HTTP</b>	Hypertext Transfer Protocol
<b>IoT</b>	Internet of Things
<b>ISO</b>	International Organisation for Standardisation
<b>ISP</b>	Internet Service Provider
<b>JRRC</b>	Joint Routing and Resource Control
<b>KNN</b>	K-Nearest Neighbours
<b>KNN CLIQ</b>	K-Nearest Neighbours Cluster Level Information Query
<b>LEACH</b>	Low Energy Adaptive Clustering Hierarchy
<b>LISP</b>	Locator/ID Separation Protocol
<b>LR-WPAN</b>	Low-Rate Wireless Personal Area Networks
<b>M2M</b>	Machine-to-Machine
<b>MAC</b>	Medium Access Control
<b>MCU</b>	Microcontroller Unit
<b>MD-SAL</b>	Model-driven Service Abstraction Layer



<b>MQTT</b>	Message Queuing Telemetry Transport
<b>NFV</b>	Network Function Virtualization
<b>NP</b>	Non-Probabilistic
<b>NETCONF</b>	Network Configuration Protocol
<b>NS</b>	Network Simulator
<b>ODL</b>	OpenDayLight
<b>OFConfig</b>	OpenFlow Configuration
<b>OLSR</b>	Optimized Link State Routing
<b>ONF</b>	Open Networking Foundation
<b>OOP</b>	Object-Oriented Programming
<b>OPNET</b>	Optimized Network Engineering Tool
<b>OS</b>	Operating System
<b>OSI</b>	Open Systems Interconnection
<b>OVSDB</b>	Open Virtual Switch Database Management Protocol
<b>PCEP</b>	Path Computation Element Communication Protocol
<b>PCIKNN</b>	Parallel Concentric-circle Itinerary-based KNN
<b>PCMM</b>	Packet Cable Multi Media
<b>PDR</b>	Packet Delivery Ratio
<b>PNN</b>	Probabilistic Neural Network
<b>QoS</b>	Quality of Service
<b>REST</b>	REpresentational State Transfer
<b>RL</b>	Reinforced Learning

<b>RT</b>	Red Triangles
<b>SAPS</b>	Situation Aware Protocol Switching Scheme
<b>SDCSN</b>	Software Defined Cluster Sensor Network
<b>SDM</b>	Software Defined Measurement
<b>SDN</b>	Software Defined Networking
<b>SDR</b>	Software Defined Radio
<b>SDSN</b>	Software Defined Sensor Network
<b>SDWSN</b>	Software Defined Wireless Sensor Network
<b>SFC</b>	Service Function Chaining
<b>SGT</b>	Security Group Tags
<b>SNMP</b>	Simple Network Management Protocol
<b>SOF</b>	Sensor OpenFlow
<b>SPIN</b>	Sensor Protocols for Information via Negotiation
<b>SSH</b>	Secure Shell
<b>SSL</b>	Secure Socket Layer
<b>SVM</b>	Support Vector Machine
<b>SXP</b>	SGT Exchange Protocol
<b>TCP</b>	Transmission Control Protocol
<b>TELNET</b>	Teletype Network
<b>UAV</b>	Unmanned Aerial Vehicle
<b>UDP</b>	User Datagram Protocol
<b>VM</b>	Virtual Machine

<b>WSAN</b>	Wireless Sensor and Actuator Network
<b>WSN</b>	Wireless Sensor Network
<b>XMPP</b>	Extensible Messaging and Presence Protocol
<b>YANG</b>	Yet-Another Network Generation

## LIST OF SYMBOLS

$\delta$	A determining function for $c(v_i)$ , i.e. $\delta(q, r) = 1$ if $q = r$ .
$c(u)$	A determined class of $u$ .
$c(v_i)$	A class, $c$ of a particular neighbour $v_i$ .
$D_{cr}$	Detection capacity of a particular sensor node in terms of range.
$d_{Ec}(a, b)$	The Euclidian distance between two points $(a, b)$ , in a dimensional space $D$ .
$D_{w_j}^k$	Classified $k$ nearest objects to $w_j$ , from the KNN operation.
$k$	A number of query points or samples in a specified “nearest neighbours” region.
$\hat{p}(c u)$	Probability function for the class $c$ , given $u$
$Q = \{(x_1, c_1), \dots, (x_N, c_N)\}$	The actual query process, performed in various points $\{(x_1, c_1), \dots, (x_N, c_N)\}$ .
$R = (w_1, \dots, w_n)$	KNN query results $(w_1, \dots, w_n)$ , in a region $R$ .
$u$	A test point in a query space or region.
$\{v_1, \dots, v_k\}$	Nearest neighbours of $u$ , in a classified query region.
$(w_j, c_i)$ and $(w_j, c_i)$	Different class instances based on the object $w_j$ .
$(x_i, c_j)$	Results instances from the KNN operation.
$(x_i, w_j)$	Similarity instances matched against the query point during the KNN process.

# TABLE OF CONTENTS

<b>CHAPTER 1</b>	<b>INTRODUCTION .....</b>	<b>1</b>
1.1	PROBLEM STATEMENT .....	5
1.1.1	Context of the problem .....	6
1.1.2	Research gap .....	7
1.2	RESEARCH QUESTIONS AND OBJECTIVE.....	7
1.2.1	Research Questions .....	7
1.2.2	Research Objectives.....	8
1.3	APPROACH.....	8
1.4	RESEARCH GOAL.....	10
1.5	RESEARCH CONTRIBUTION .....	10
1.6	RESEARCH OUTPUTS .....	11
1.7	OVERVIEW OF STUDY .....	12
<b>CHAPTER 2</b>	<b>LITERATURE STUDY .....</b>	<b>14</b>
2.1	CHAPTER OBJECTIVES .....	14
2.2	WSN ROUTING TECHNIQUES .....	15
2.2.1	Design Constraints for WSN Routing Protocols .....	20
2.2.2	WSNs Communication and Routing Protocols.....	21
2.3	WSN: APPLICATIONS AND SYSTEM CHALLENGES.....	24
2.3.1	WSN APPLICATION SYSTEMS .....	24
2.3.2	WSN SYSTEM CHALLENGES.....	26
2.4	SDN PARADIGM.....	27
2.4.1	SDN Controllers.....	32
2.5	SDN FOR FLEXIBLE RESOURCE MANAGEMENT IN IOT NETWORKS ..	33
2.5.1	Application Challenges in IoT Networks .....	38
2.5.2	SDN Benefits for IoT Networks .....	41

2.5.3	In-Network Cache Computing for IoT Objects .....	43
2.5.4	IoT Objects Communication and Data Integrity.....	44
2.5.5	SDN Enabling Secure WSN Based IoT Systems .....	45
2.6	SDWSN: APPLICATION IMPROVEMENTS FOR WSN SYSTEMS .....	46
2.6.1	Network Resource Management.....	59
2.6.2	Network Flexibility.....	61
2.6.3	SDN for Scalable WSN Networks.....	62
2.6.4	Machine Learning Approaches in WSNs and SDWSNs.....	64
2.7	CHAPTER SUMMARY.....	68
<b>CHAPTER 3</b>	<b>SDWSN: A FLEXIBLE WSN NETWORK .....</b>	<b>70</b>
3.1	CHAPTER OBJECTIVES .....	70
3.2	SIMULATION ENVIRONMENT.....	70
3.2.1	NS-3 Network Modules and Model Library .....	70
3.2.2	ODL: Enabling the ODL Controller .....	74
3.2.3	OpenFlow Communication Standard.....	76
3.2.4	BGP AND NETCONF.....	80
3.3	EXPERIMENTAL SETUP .....	81
3.4	EXPERIMENTAL RESULTS .....	84
3.5	CHAPTER CONCLUSION .....	93
<b>CHAPTER 4</b>	<b>APPLYING MACHINE LEARNING TECHNIQUES IN SDWSN .....</b>	<b>95</b>
4.1	CHAPTER OBJECTIVES .....	95
4.2	THE KNN ALGORITHM .....	95
4.2.1	KNN CLIQ.....	97
4.3	EXPERIMENTAL SETUP .....	98
4.4	EXPERIMENTAL RESULTS .....	109
4.5	CHAPTER CONCLUSSION.....	117
<b>CHAPTER 5</b>	<b>DISCUSSION.....</b>	<b>119</b>
<b>CHAPTER 6</b>	<b>CONCLUSION .....</b>	<b>123</b>
<b>REFERENCES</b>	<b>126</b>	

<b>ADDENDUM A</b>	<b>SOFTWARE AND HARDWARE COMPONENTS.....</b>	<b>143</b>
<b>ADDENDUM B</b>	<b>OPENFLOW SOFTWARE SPECIFICATION .....</b>	<b>144</b>
<b>ADDENDUM C</b>	<b>ODL OXYGEN RELEASE FRAMEWORK.....</b>	<b>146</b>
<b>ADDENDUM D</b>	<b>NS-3 STATISTICAL FRAMEWORK .....</b>	<b>147</b>
<b>ADDENDUM E</b>	<b>SDN BASED WSN SIMULATION CODE SNIPPETS .....</b>	<b>148</b>

## LIST OF FIGURES

<b>Figure 1.1:</b> An Overview of a Wireless Sensor Network in a Monitored Field.....	2
<b>Figure 1.2:</b> WSN applications fields and sensor technology segments.....	3
<b>Figure 1.3 :</b> Communication Mechanism for sensor devices in a simulation environment.	4
<b>Figure 2.1:</b> WSN protocol stack with critical layers and management planes.....	16
<b>Figure 2.2:</b> WSN routing protocol classification based on network structures.....	20
<b>Figure 2.3:</b> An overview of SDN architecture.....	29
<b>Figure 2.4:</b> IoT architecture with SDN as its core feature.....	37
<b>Figure 2.5:</b> An overview of the SDWSN Framework .....	47
<b>Figure 2.6:</b> Formulated SDN strategy in terms of different network tasks execution, service association and resource allocation with QoS requirements. ....	60
<b>Figure 2.7:</b> A distributed SDN controller network infrastructure. ....	64
<b>Figure 3.1:</b> A collaboration diagram for certain applications with the object base and the parent instance. ....	73
<b>Figure 3.2:</b> A description of an object class and its related class implementation in NS-3. ....	74
<b>Figure 3.3:</b> OpenFlow based SDN infrastructure with data plane communication support. ....	77
<b>Figure 3.4:</b> OpenFlow's flow entry operations on different flow tables.....	78
<b>Figure 3.5:</b> An OpenFlow's process flow operation on flow entries. ....	79
<b>Figure 3.6:</b> NETCONF Protocol Layers.....	80
<b>Figure 3.7:</b> BGP data flow diagram.....	81
<b>Figure 3.8:</b> Presentation of the Network Model. ....	83
<b>Figure 3.9:</b> ODL OpenFlow communication.....	84
<b>Figure 3.10:</b> SDWSN packets flow illustration.....	85
<b>Figure 3.11:</b> Random sensor traffic transmission.....	86
<b>Figure 3.12:</b> Data Aggregation Plot for Three Different SDN Controllers.....	87
<b>Figure 3.13:</b> Packet Error Rate per Number of Sensor Nodes. ....	88
<b>Figure 3.14:</b> Energy Consumption for over Data Packet Size for Considered SDN Protocols. ....	89



<b>Figure 3.15:</b> Packet Loss Estimation over Throughput. ....	90
<b>Figure 3.16:</b> End To End Delay Over Sensor Nodes.....	91
<b>Figure 3.17:</b> Throughput Plot over Transmitted Data. ....	92
<b>Figure 4.1:</b> Classification operation of KNN algorithm using optimal “k” values. ....	97
<b>Figure 4.2:</b> Processes Flows of both KNN CLIQ and the COSPA Algorithms Relative to the ODL Controller.....	101
<b>Figure 4.3:</b> Process flowchart of the KNN-CLIQ algorithm.....	106
<b>Figure 4.4:</b> A flow diagram of the COSPA algorithm. ....	108
<b>Figure 4.5:</b> CDF percentage for different values of k .....	110
<b>Figure 4.6:</b> Classification times for different values of k.....	111
<b>Figure 4.7:</b> Association probability plot for various query attempts on specific regions. ....	112
<b>Figure 4.8:</b> Data aggregation efficiency plot.....	113
<b>Figure 4.9:</b> A data aggregation plot for all the compared algorithms. ....	114
<b>Figure 4.10:</b> Average query classification times for all compared algorithms.....	115
<b>Figure 4.11:</b> A CDF function plot for all compared algorithms.....	116
<b>Figure 4.12:</b> Association probability of all compared algorithm. ....	117

## LIST OF TABLES

<b>Table 2.1:</b> A description of WSN protocol stack layers. ....	17
<b>Table 2.2:</b> A Protocol Stack for WSNs.....	21
<b>Table 2.3:</b> Classification of WSNs Routing Protocols. ....	22
<b>Table 2.4:</b> Advantages and disadvantages of WSN protocols.....	24
<b>Table 2.5:</b> Common WSN applications and their technology systems. ....	25
<b>Table 2.6:</b> Comparison between an SDN Based Network and a Traditional Network. ....	31
<b>Table 2.7:</b> Popular SDN Controllers and Their Descriptions. ....	33
<b>Table 2.8:</b> Immediate IoT systems challenges with technological descriptions.....	40
<b>Table 3.1:</b> Critical modules of the NS-3 simulator with their descriptions.....	71
<b>Table 3.2:</b> Core platform services and applications for the ODL framework. ....	75
<b>Table 3.3:</b> Environment and System Parameters .....	82
<b>Table 4.1:</b> Notations and parameters used, with their brief descriptions. ....	99
<b>Table 4.2:</b> System parameters and values.....	100

## LIST OF ALGORITHMS

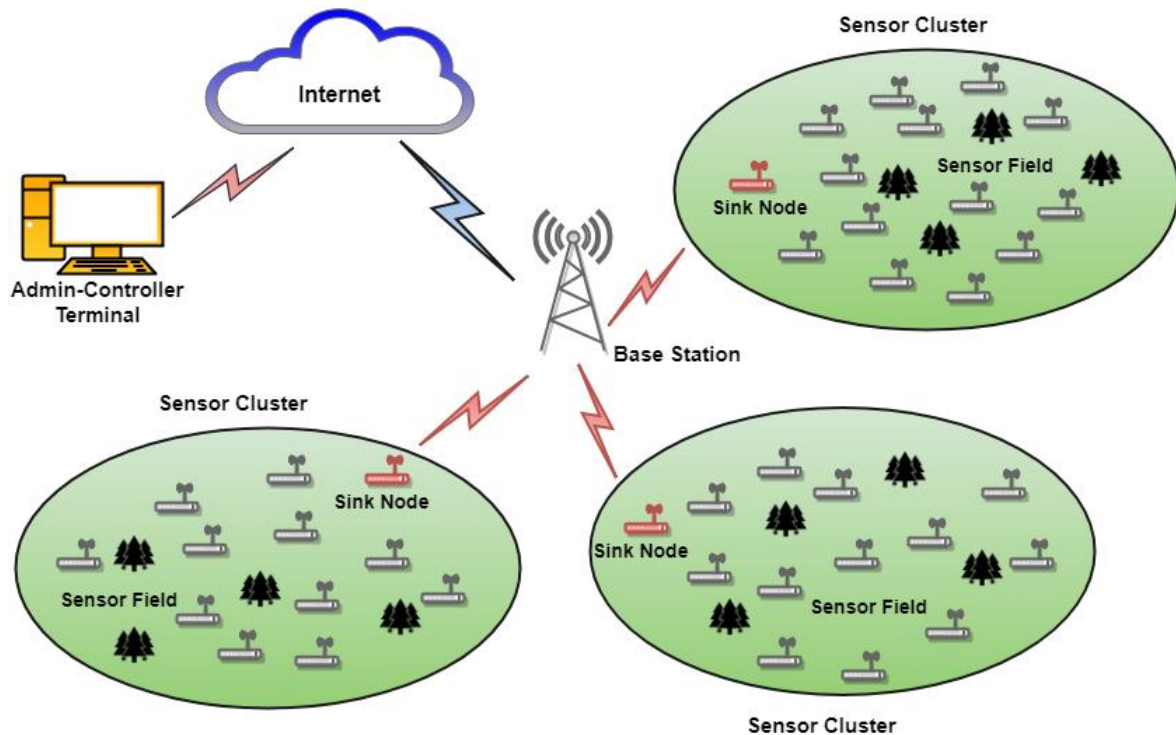
<b>Algorithm 4.1:</b> The adopted original KNN algorithm .....	104
<b>Algorithm 4.2:</b> The proposed KNN-CLIQ algorithm .....	105
<b>Algorithm 4.3:</b> The proposed COSPA algorithm.....	107

# CHAPTER 1            INTRODUCTION

WSNs are application-specific sensor network technologies which are widely deployed in applications such as in; environmental monitoring, atmospheric monitoring, process monitoring, material sensing, security or surveillance systems, medical systems, etc. These networks operate on collective computing capabilities of individual sensors based on their physical sensing properties and processing capabilities. Sensors nodes, cooperatively communicate and relay aggregated data to the main network control system for further processing and acting. In this regard, these sensors, must have some kind of ability to conform to the collective networking functionalities as governed by their respective network policies. In WSNs, sensor nodes can be randomly deployed, in essence allowing opportunities for applications even in inaccessible areas [1]. This feature about sensor networks, allows the possibility of deploying a large number of sensors over intuited areas for as long as communications can be established and sustained among these sensor nodes.

A WSN consists of but not limited to; a wireless sensor network server (attached to a power source), routers, switches, sensor nodes, etc. depending on the design setup as required for its purpose. Depending on the purpose of the WSN, sensor nodes communicate amongst themselves as a means to forward sensed data to the network sink, which will then be passed by the network sink to the WSN server for further processing. Even though there are some network stability reflections to be noted about node failures in WSNs such as network partitioning [2], [3] (which lead to poor Quality of Service (QoS)) and more energy depletion due to neighbouring node rerouting as an effort to amend faults [4], [5] a failure in one of the sensor nodes does not collapse the network [6], [7]; rather network traffic is routed [8] through adjacent sensor nodes thereby sustaining the traffic flow [9], [10]. In cases when

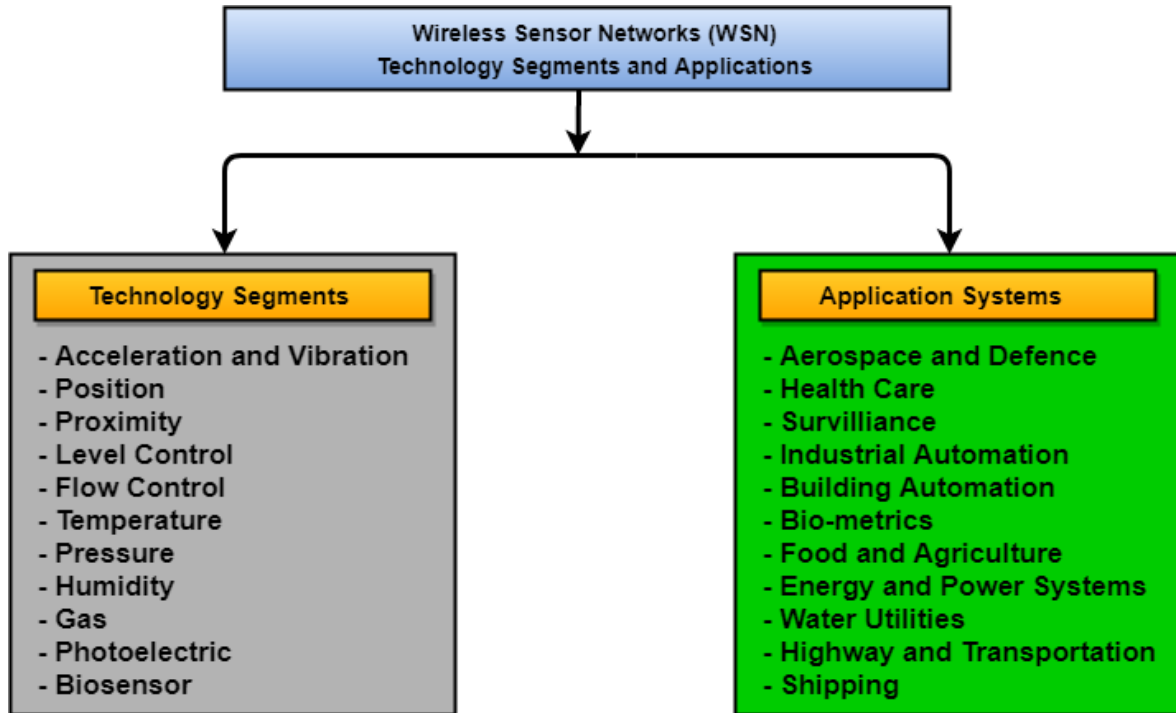
accessing a WSN remotely, a network sink may also be used as gateway to connect to the internet [11]. This phenomenon about WSNs can be regarded as an opportunity for network computing innovation. An overview of a wireless sensor network for a monitored field is shown in FIGURE 1.1.



**Figure 1.1:** An Overview of a Wireless Sensor Network in a Monitored Field.

WSNs are envisioned to be deployed on a larger scale, as millions of wireless sensor nodes will be working cooperatively to transmit critical data and at the same time being connected to the internet as an effort towards the realization of IoT [12]. Commonly reported challenges include; sensor node energy limitations, low memory and processing capacity, low channel bandwidth and being application specific [13], [14]. A lot of work has been done particularly on energy limitations such as in [15], [16], as an effort to improve the node energy utilization on WSNs. In addition to that, other methods of energy harvesting have been proposed as a means to leverage this limitation on sensor nodes [17], [18]. However, only to a certain extent, this particular energy issue has been moderated, as it is still one of the serious challenges in WSNs [19], since it affects the lifetime of a WSN directly. Other reported

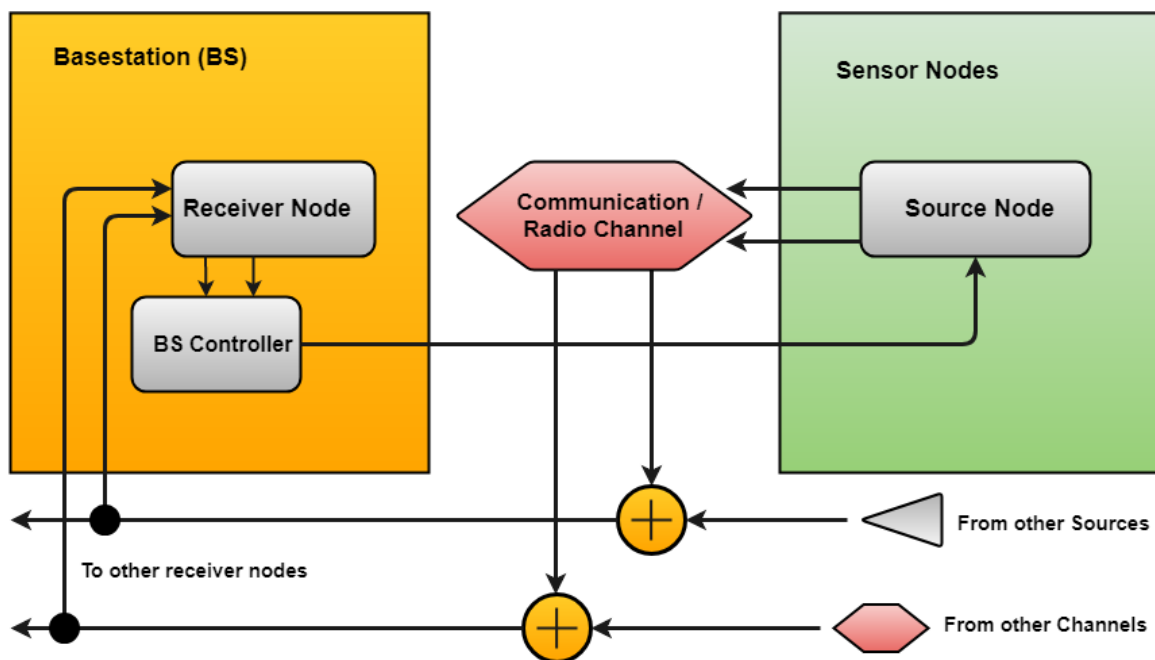
challenges such as in [20], [21] are, high latency in communication (which occurs as a result of multi-hop routing), traffic congestion, processing delays on intermediate nodes due to the packet-based routing nature of WSNs. Figure 1.2 presents some commonly known WSN application systems and sensor technological segments thereof.



**Figure 1.2:** WSN applications fields and sensor technology segments.

These application technologies benefit from the use of sensing devices. However due to limitations in WSN systems, there are very limited opportunities to bring innovation into the network. Computing aspects such as memory and processing power are the mostly critical resources of WSN systems. Therefore, to improve these technologies, it is necessary to develop efficient resource-aware strategies that will rather enhance their operations. The cooperative nature of sensors in terms of data aggregation and routing exhibits some level of system intelligence that can be used to improve their applications. Sensor technologies allows smart applications such as in acquisition, control and monitoring [22], [23]. These technologies are commonly deployed for sophisticated operations that requires lightweight but efficient sensing capabilities. Hence, the use of sensor applications for modern control technologies is rapidly growing.

To optimize sensor communication or traffic routing in a simulation environment, an understanding of how data packets are transmitted is critical. Figure 1.3 below, illustrates this process given different sensors numbers acting as both data sources and receivers. The figure describes collaborative data transmission to the base station from different sources. This data transmission mechanism provides the foundation for developing efficient data transmission techniques, which will result in commendable performances. It is also important to consider transmission or channel distortions as data (in the form of electrical signals) is propagated within the system. Different network devices share all the network communication or radio channels to transmit traffic to various destinations



**Figure 1.3 :** Communication Mechanism for sensor devices in a simulation environment.

On the basis that it is not yet possible to realize WSNs that have a prolonged network lifetime (as sensor nodes are battery-power reliant), important considerations must be made before the deployment phase of the network. These considerations includes (but not limited to); understanding phenomena/event requirements and the deployment environment (as this will assist in deciding on relevant equipment and their capability, and also whether there are no radio frequency disturbances within the area (in case of industrial or city/infrastructure deployments )), sensor network connectivity (i.e. critical measure must be taken to maintain

connectivity), the ability of the network to self-configure in adverse or intrusion situation [24], [25], choice of wireless protocol depending on the sensor type to be used (as protocols/standards differ in power utilization, throughput and communication range) [26], [27]. We emphasize on this so that if there are counter measure (either software-based or enhanced equipment/devices) on these consideration, be it that they are accounted for on the network planning phase.

Network programmability, such as that introduced by SDN, is seen as a potential direction to fully evolve computing networks especially those that used for human survival. SDN proposes a system whose controlling functionality is central but well-resourced to facilitate the underlying network infrastructure from its point. In terms of WSNs, this means, the centralized SDN controller will be responsible for executing network intelligence, so that the underlying network devices, perform data transmission. This framework is presented as software defined wireless sensor network (SDWSN). However, it is yet to be realized how SDN can improve computing capabilities of WSNs application technologies. It is also noteworthy to mention that, because of its early state of activity, current SDN systems are still under a lot of developmental work and testing. Therefore, it must be appreciated that, a lot of work spanning different proposals and approaches is still to be done. However, this does not disregard the already achieved work under the area. This work aims to develop intelligent software-oriented strategies that will be implemented in resource limited WSNs, with application focus on monitoring systems. Machine learning approaches will be used to develop network intelligent algorithms which will form the basis for improved network knowledge presentation and efficient data aggregation.

## **1.1 PROBLEM STATEMENT**

This section describes the challenges and networking demands that informs the efforts of developing network-computing systems that are highly efficient. Through experimentation, the performance potential of proposed approach in terms of providing realistic strategies that can be implemented to improve WSN application systems is also justified, especially those deployed for life-concerning missions.



### 1.1.1 Context of the problem

The modern era of network computing is confronted with a high demand of powerful systems that not only are reliable and fast but also are, adaptable to requirement changes. This is predominantly due to the radical advancement in wide computing platforms operating at highly descriptive and abstracted mediums such as; reconfigurable computing systems, smart automation systems, parallel programming and cognitive systems which communicate using highly complex resources or methods. Hence, such systems must integrate the best forms of technologies to accommodate the rapidly growing and heterogeneously connected platforms which uses different means of interconnecting diverse machines or devices. In addition, these systems must be able to reach unusual environmental spaces, be cost effective, easy to deploy and manage.

Due to their deployment simplicity and cost advantages, WSNs have emerged as favourable platforms that can be used to advance application technologies for mission critical systems. However, these technologies have not yet been deployed as fully trusted systems due to their resource limitations, spanning from; memory and computing limitations, limited power sustainability, communication and delay constraints. Therefore, there is a need to enhance these network computing technologies for them to be realized as fully capable systems for mission critical systems. To put into perspective, these technologies must be optimized to fully support today's highly sophisticated life-oriented systems such as medical, surveillance, aeronautical and agricultural systems. However, to effectively manage such multifaceted high-end computing resources, requires well-organized and carefully implemented systems. These systems must be able to cater for any change that counts to the benefit of the users and customers at large. They must be designed with a focus to be efficient in terms of performance and be scalable throughout.

---

### 1.1.2 Research gap

Due to the technical constraints as well as limited network performance of WSNs, a lot of research approaches has been applied and tested in this field as a means to improve their application capabilities. Even though some work has resulted into great achievements, WSN application systems are still experiencing issues especially in their network capability and resource management. Some approaches work to a certain extent or for specific conditions only, which still reflects considerable limitations in their applications.

This work applies SDN strategies to these application technologies as an effort to enhance their network performance as well as their application capacity using a high-level programming language. Software driven policies and communication protocols are implemented to the actual network infrastructure to bring innovation, allow efficient resource management, introduce network flexible as well as to improve their overall network performance.

## 1.2 RESEARCH QUESTIONS AND OBJECTIVE

The following subsection provide a list of formulated research questions, as well as the developed research objectives, which then provides the direction of the study.

### 1.2.1 Research Questions

Subsequent to the technical considerations in this study, the following questions where formulated:

1. How will SDN solve these reported constraints about WSNs since SDN itself has challenges?
2. What other options exist that can augment the fast realization of SDN?
3. If SDN technology happens to be realized, what will be involved in the task of transiting from legacy networks to SDN enhanced network environments?
4. Will the SDN architecture be considered for production networks or not, given its form of implementation and infrastructure?
5. How does the current state of SDN affect network hardware design?

### 1.2.2 Research Objectives

To technically respond to the formulated research questions, the objectives of this research work are:

1. To implement SDN programmability strategies in WSNs to achieve network flexibility as well to improve their application potential.
2. To develop a SDWSN strategy for sensor traffic load balancing in monitoring application systems.
3. To formulate and implement SDN strategies to perform network services abstraction for improving network resource management.
4. To formulate a process interfacing strategy to integrate machine learning techniques to SDWSN systems.
5. To develop a Controller-based Open Status Present Algorithm (COSPA) that uses Machine Learning techniques for increased data aggregation and network knowledge.

### 1.3 APPROACH

In this work, software-oriented network functionalities, programmable networking strategies and the network control will be implemented and utilized for efficient network resource management in SDWSN systems, with the focus for monitoring network applications. Implementation options for the proposed approach will be provided for both small and large WSN application systems. Based on the actual plan of this work, the proposed research methodology was to carry out the following:

1. Literature study:
  - a. Carried out a detailed literature study on WSNs in terms of their application and system challenges.
  - b. Conducted a thorough study on SDWSN systems and its applicable strategies for monitoring applications.
  - c. The conducted literature on these areas were thoroughly analysed, wherein a study plan was developed.

- d. Various approaches in SDN based WSNs were identified.
  - e. Different approaches were critically analysed and compared.
2. Develop SDWSN and Machine Learning strategies for improved network operation and performance:
- a. A novel approach for a flexible and scalable SDWSN was designed.
  - b. A software-oriented strategy for efficient resource management for SDWSN systems was developed.
  - c. A cluster based, and query efficient machine learning technique was developed for improving SDWSN data aggregation.
  - d. An SDN controller-based status presentation algorithm was developed to enable accurate knowledge presentation to the system controller.
  - e. Both the developed machine learning and SDN controller-based algorithm were coupled to form an intelligent system for efficient query processing, data transmission and reliable information presentation to the implemented controller.
3. Implementation:
- a. The formulated strategies for improved network performance were implemented using a Discrete Event Simulation (DES) tool coupled with programmable capabilities of an SDN controller.
  - b. Different system design approaches that use the same methods were also simulated and compared with the formulated strategies.
  - c. Relevant data from all simulations was recorded for statistical, evaluation and comparison purposes.
4. System Analysis and Result Discussions
- a. Formulated strategies based on the proposed approach were implemented for real world monitoring application systems scenarios. Realized improvements were carefully analysed.
  - b. Other variations of the implemented strategy were performed, as a purpose for different developmental options for optimal results.
  - c. Achieved results from the proposed method were carefully analysed and compared to that of the existing work.

- 
- d. These achievements were documented as validation and proof of efficiency by the proposed approach, in this thesis.

#### **1.4 RESEARCH GOAL**

The main goal of this research is to develop and implement efficient programming methods to improve WSNs monitoring applications using SDN strategies and Machine Learning techniques. These strategies are focused on improving network flexibility, enable efficient resource management, and improve sensor data aggregation as well as ensuring efficient communication between the SDN controller and the underlying network infrastructure.

#### **1.5 RESEARCH CONTRIBUTION**

This study directly contributes in the area of WSNs by exhibiting innovative strategies to the management and operational functionality of the said application technologies with emphasis to develop scalable and flexible systems. Furthermore, SDN programmable strategies for network computing will be developed in essence to mitigate sensor node limitations such as; memory and computing capacity, bandwidth utilization and load balancing. Lastly, in this work, Machine Learning techniques were developed and successfully coupled with SDWSN strategies to improve networking intelligence for monitoring application systems. It is further highlighted that, though the systematic view of an SDN architecture seems to be complex, its capability to managing and customizing WSNs will make a huge impact towards the evolution of computing networks. This work also contribute immensely towards the manufacturing industry for future customizable wired and wireless networking devices.

Moreover, this research work open up research opportunities for programmable WSNs applications, especially for large scale developments. This work further open up opportunities for programmable network prototyping and also the integration of numerous network operations or applications development using different high-level programming languages in WSN networks. From this work, developmental analysis and directions towards effective applications of WSNs will be given as for future work. It is emphasized that this

proposed approach will enormously contribute towards the realization of IoT in future computing networks.

## 1.6 RESEARCH OUTPUTS

To this point, there has been a number of research outputs that have been achieved from this research study, which are listed below:

- a. K. M. Modieginnyane, 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, September 2016. {*SATNAC, WIP, Published*}.
- b. K. M. Modieginnyane, 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, {*Elsevier, Survey Paper, Published*}.
- c. K. M. Modieginnyane, 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*. {*Springer, Technical Paper, Published*}.
- d. K. M. Modieginnyane, R. Malekian and B.B. Letswamotse, “Increased Network Knowledge Using a Status Update Algorithm for Monitoring Applications in Software Defined Wireless Sensor Networks”. {*Technical Paper, Submitted, to Springer*}
- e. B. B. Letswamotse, K. M. Modieginnyane, and R. Malekian, “SDN based QoS provision in WSN technologies,” in *Southern Africa Telecommunication Networks and Applications Conference (SATNAC)*, George, South Africa, Sep. 2016. {*SATNAC, WIP, Published*}.
- f. B. B. Letswamotse, R. Malekian, and K. M. Modieginnyane, “Resource-aware QoS Support for Efficient Congestion Control in Software Defined Wireless Sensor Networks.” {*Technical Paper, Submitted, to Springer*}

## 1.7 OVERVIEW OF STUDY

This study provides a comprehensive account of achieving efficient network management with resource flexibility in SDWSN. The study is focused on wireless monitoring application systems. Background on associated aspects and techniques applied in this research area is provided with the aim to formulate strategies to develop and implement intelligent methods to achieve the objectives of the study. A review of related achievements in this area is given, with critical analysis that contributes to the development of efficient SDN methods for improved network management. A detailed understanding of the SDWSN framework is provided with developed strategies for improving network adaptability and the overall performance. Machine learning techniques are adapted to this study wherein intelligent strategies are developed to improve resource management and increase network visibility. A thorough discussion based on performances of formulated strategies and achieved experimental results is given with critical analysis. The rest of the thesis is structured as follows:

- *Chapter 2:* Provides a review of related work in the area of SDWSN in terms of resource management, efficient performance and introducing innovation to the network. An in-depth analysis of these achievements is made to contribute towards developing new techniques to improve WSN application networks.
- *Chapter 3:* Based on the gathered information, in this chapter, SDN programmable strategies will be formulated, developed and implemented to realize network flexibility, efficient resource management and improve network performance. Developed strategies are compared to other approaches in terms of experimental performances.
- *Chapter 4:* In this chapter, machine learning techniques are developed and implemented to SDWNS application systems with the purpose to enhance their information acquisition, data aggregation and knowledge presentation. Experiments are conducted, developed methods are compared to other competing strategies and results are critically analysed in terms of various performances.

- 
- *Chapter 5:* This chapter provides thorough discussions in terms of achieved performance results. Performance impacts of all developed and implemented strategies are clearly explained and benefits brought about by these strategies are discussed.
  - *Chapter 6:* In this chapter, a general research conclusion is provide based on what has been achieved in this study, especially in terms of the contributions realized through the developed and implemented strategies.



## **CHAPTER 2      LITERATURE STUDY**

This chapter deals with theoretical background of all the technical aspects that forms part of this study, with critical analysis on related work that forms the basis of this research. Existing work, in terms of proposals and technical achievements in the field will be critically analysed to provide comparative reasons, which will build to the importance of this research work.

### **2.1    CHAPTER OBJECTIVES**

The objectives of this chapter are to:

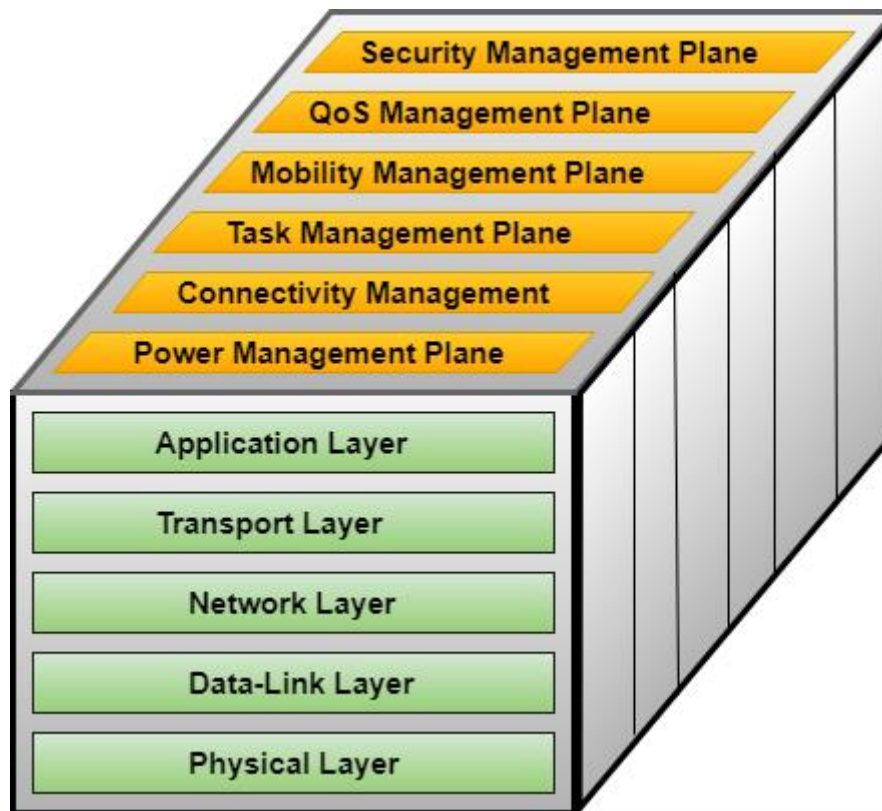
- Conduct a thorough literature study around the area of research. This includes, acquiring knowledge of research challenges experienced in the area as well as solutions that have emerged in an effort to solve these challenges.
- Acquaint myself with approaches, methods and technologies used in the research area. This involves carefully providing potential strategies that could be applied to solve experienced technological challenges in the area.
- Provide relevant and up to date research references which forms the research focus of my work. This builds towards the actual methods and strategies that will be implemented in my work.
- Critically analyse surveyed research references and draw remarks regarding the actual achievements on the area. This will build towards strong knowledge around the technological application.
- Provide sound methodological options that could mitigate challenges experienced in the area. This will provide support towards the methodology chosen in this work.

From these objectives, a sound strategy for SDWSN will be drawn towards the implementation of a flexible, scalable and sustainable WSN application system. This strategy will be thoroughly interpreted as methods for this research work.

## **2.2 WSN ROUTING TECHNIQUES**

Cooperative communication of WSN devices is driven by electronics communication techniques/technologies, which are enabled by the electromagnetic radio waves. These techniques allow wireless communication operations amongst these devices as a means to provide connectivity for the actual exchange of data. Hence, a thorough understanding of how wireless networking devices acquire and share data is needed, especially for developmental purposes. In a sensor network, wireless communication occurs as a result of signals broadcasting (due to an event triggered or as a means of relaying network transmitted data) of a source sensor node to a dedicated node using employed communication or routing protocols. This phenomenon involves cooperative communication of adjacent sensor nodes to relay transmitted data to the destination node or targeted application services. It is thus, critical to have technical knowledge of these communication technologies looking at; communication channels and distance, speed and radio signal frequency, since this is most important for mission critical systems.

Figure 2.1 illustrates a protocol stack of WSNs with critical layers and management planes. These layers implement the operational functionalities of wireless transmitting sensor nodes within a WSN. Using these layer, network functions and roles can be applied to perform different networking tasks.



**Figure 2.1:** WSN protocol stack with critical layers and management planes.

The WSN protocol stack provides the processing capability property of wireless communication devices. This stack informs how activities and data are controlled by communicating devices. To enhance WSNs' application potential, the layers of the protocol stack must be fully understood. For example, various application-layer protocols of the application layer must be well studied when developing applications that will directly interact with this layer to fulfil certain application tasks. This applies to all the other layers of the stack together with the associated management planes. Table 2.1 provides a description of the stack layers appearing in Figure 2.1.

**Table 2.1:** A description of WSN protocol stack layers.

Stack Layer	Description
<b>Application Layer</b>	This layer is responsible for performing various sensor application activities such as; node localization, query propagation, process synchronizations, security activities, etc. It is critical in that it supports software-oriented applications for manipulating sensors nodes activities or orientations.
<b>Transport Layer</b>	Facilitates end-to-end sensor data aggregation. This layer ensures reliable communication and networking. Using this layer, lightweight transmission protocols are enabled for resource-constrained sensors to meet quality end-to-end communication without straining network resources.
<b>Network Layer</b>	Ensures routing capabilities between networking devices. Using this layer, sensor nodes employ relevant routing protocols, which are resource efficient, to transmit sensed data to various network destinations. In most case, a Multihop short-range routing is employed in WSNs.
<b>Data Link Layer</b>	Facilitates the medium access control (MAC) operations for efficient resource management. This layer ensures efficient network performance by managing communication throughput, packet delivery and sensor energy.
<b>Physical Layer</b>	Responsible for the adaptation of data-link layer bit streams into meaningful signal, which are then transmitted through various communication mediums. This layer is extremely critical, as it, interface with different networking devices and their operational capabilities.

Depending on their specific application, WSNs uses special sensor devices to detect process and transmit sensed data to controlling nodes or application information systems for further processing and analysis. This process requires accuracy and acceptable processing speed in terms of sensing since sensed data needs to be of integrity and time sensitiveness to avoid disastrous situations. Therefore, the type of sensing devices to be used in any sensor network system, must be informed by the type of application that the system is undertaking.

Furthermore, the setup must facilitate for the types of communication techniques to be used. Depending on the network architecture and system application, WSN routing techniques are categorised as in:

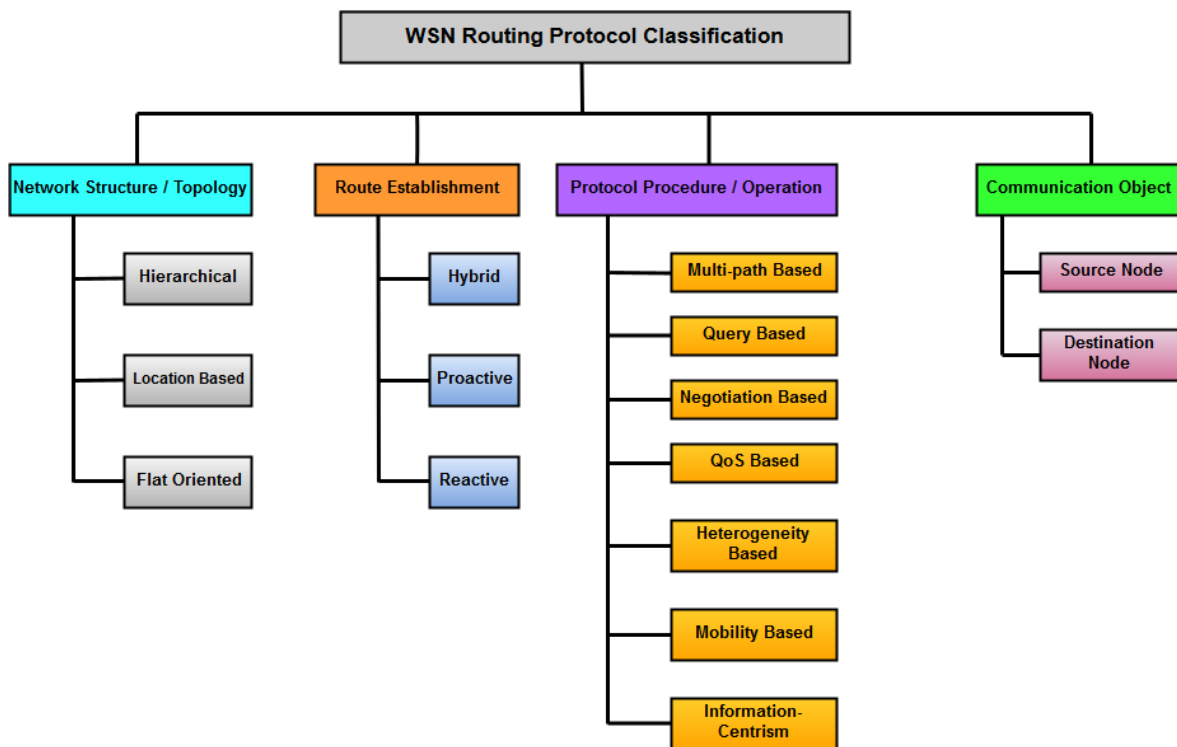
- *Information-centric Routing Protocols.* In this routing protocol, data aggregation is performed amongst the source and sink nodes in a way to communicate. The source node generates a query-based data and awaits destination response. After sensing a query event, the destination node then responds by reverse transmitting sensed data to the source node [28]. The source node may be a sink node or an authorised remote network application service. This type of routing has less energy consumption.
- *Location-based Routing Protocols.* Sensor nodes' position data is used for routing purposes, whereby routing signal strength is used to calculate routing distances [29]. Routing algorithms using this protocol are energy-efficient since sensor nodes operate using radio ON or OFF modes depending on whether there is data transmission detection or not. Sensor nodes listening or on idle modes.
- *Multipath-based Routing Protocols.* This type of routing techniques uses a multipath routing scheme to enhance network availability and reliability. This routing technique is fault-tolerant due to the fact that, any network fault such as path failure, the network uses alternate path to route traffic [30]. However, due to its great performance, this routing technique tends to use more energy and has high traffic.
- *Mobility-based Routing Protocols.* In this routing techniques a sink node is rather mobile circulating the sensor cluster either than being stationary or fixed. This model was proposed [31] in an effort to minimize high energy dissipation from cross routing cluster sensor nodes. However, this technique requires algorithms that are energy-efficient since mobile sink nodes utilize high energy.
- *Hierarchical Routing Protocols.* This type of routing protocol supports both heterogeneous and homogenous sensor nodes [32]. Sensor nodes are organized into clusters wherein a cluster-head performs the actual data transmission out of the cluster to other clusters or the far base station. This type of sensor topology is suitable

---

for large WSN since transmitted data can cover large areas or distance. Routing algorithms employed in this protocol are energy-efficient since every sensor node belonging to a particular cluster participate fairly in data aggregation operations thereby saving energy.

- *QoS-based Routing Protocols.* This type of routing techniques employs a mechanism to meet QoS routing requirements and decrease sensor energy consumption [33]. This technique is applied to maintain a reliable network that is fault-tolerant and has limited delays.
- *Heterogeneity-based Protocols.* In this wireless routing setup, the network is designed based on two different sensor nodes types being battery-powered and line-powered nodes [34]. Sensor nodes use energy as minimal as possible to prevent them from quickly running out of power. Sensor nodes, which are line-powered, are used so as prolong the network lifetime even when battery-powered sensor nodes have ran out of energy.

Figure 2.2 illustrates how these protocols are classified in terms of their network orientation and communication objects.



**Figure 2.2:** WSN routing protocol classification based on network structures.

### 2.2.1 Design Constraints for WSN Routing Protocols

WSN application systems requires a technical balance in terms of networking and computation for the best performance of the entire network. However, for a while, this has been a serious challenge to WSN application systems due to their limitations of meeting technical requirements that are necessary for optimal performance. These limitations span around their weak processing capability in terms of power and memory, short network life and unreliable data transmission capacity due to signal or electromagnetic radio wave disturbances. That being the case, a lot of research work has been done to at least deal with the extent of these challenges. Therefore, the use of routing protocols in WSNs is to meet the following systematic requirements:

- Energy efficiency
- Application adaptability
- Network resilience
- Network scalability

- Heterogeneity

### 2.2.2 WSNs Communication and Routing Protocols

The protocol stack as defined by the International Organisation for Standardisation (ISO) governs systematic communication in WSNs. This protocol stack comprises of policy layers that ensure connectivity and communication amongst sensor devices. Each policy layer is responsible for executing specific layer services to achieve networking and communication in WSN infrastructure. Based on the protocol stack design, these layers operate independent of one another in terms of achieving their designated tasks. TABLE 2.2 below, illustrate this protocol stack with a description of each layer depending on its duty.

**Table 2.2:** A Protocol Stack for WSNs.

Layers	Description
<b>Application Layer</b>	This layer serves purpose for user-oriented applications to the network. Here reside applications such as HTTP, FTP, TELNET, SNMP, etc.
<b>Transport Layer</b>	Serves purpose for transport layer protocols, which are TCP and UDP. These protocols ensure reliable communication between sensor devices.
<b>Network Layer</b>	Responsible for providing device network connection and ensures that packets are successfully transmitted.
<b>Data Link Layer</b>	Guarantees that sensor devices access and share a communication medium.
<b>Physical Layer</b>	Ensures connection between sensor devices to different communication mediums.

It is also noteworthy to mention that, unlike the Open Systems Interconnection (OSI) model, which has seven layers, the WSN ISO protocol stack model has only five layers, which makes it easy to implement. This stack also makes it possible for WSNs to be easily designed and implemented. The protocol stack for WSNs discussed here is based on the IEEE 802.15.4 standard, which is compatible for sensor devices that communicate in short or limited distances. This, however, is one of the reasons why WSN systems are mostly



applicable for short-range communication, since traffic is of low data throughput and due to their processing and computing limitations.

Depending on how routing or path determination, routing protocols for WSNs are classified as either being reactive or proactive. Reactive WSN routing protocols calls on routing discovery mechanisms based on requests, whereas routing protocols in the proactive category, establish the route before requests are made and then updates such routes as the network topology is changed. TABLE 2.3, illustrate classification of these routing protocols categorised as either oriented on the network structure of protocol operation.

**Table 2.3:** Classification of WSNs Routing Protocols.

<b><i>Protocol Operation Oriented</i></b>	<ol style="list-style-type: none"> <li>1. Query-Based Routing</li> <li>2. Negotiation-Based Routing</li> <li>3. Multipath-Based Routing</li> <li>4. QoS-Based Routing</li> <li>5. Coherent-Based Routing</li> </ol>
<b><i>Network Structure Oriented</i></b>	<ol style="list-style-type: none"> <li>1. Flat Network Based Routing</li> <li>2. Hierarchical Based Routing</li> <li>3. Location Based Routing</li> </ol>

To improve WSN system operations and applications, several access and routing protocols have been developed and applied in this area, which includes the following:

- *Medium Access Control (MAC)* protocol [35] – whose main strategy is to reduce high energy consumption, since sensor nodes in a WSN systems are battery powered thereby resulting in a limited network lifetime. MAC protocol attributes for WSNs include collision avoidance, network scalability, efficient bandwidth utilisation, improved throughput and latency, etc.
- *Low Energy Adaptive Clustering Hierarchy (LEACH)* protocol [36] – which is the fundamental protocol to propose some level of data fusion as well as a focus to implement a strategy for low power utilisation in hierarchical WSNs.

- *Ad hoc On-Demand Vector (AODV)* routing protocol [37] – its main objective is to reduce packets flooding which causes overhead within the network. One critical functionality of this protocol is to utilize routing tables to store routing information. This assist in situations where sensor nodes needs to send packets to targeted destinations.
- *Sensor Protocols for Information via Negotiation (SPIN)* [38] – These types of protocols are based on sensor nodes negotiations for allowing data transmission and resource adaptation mechanisms for energy saving.
- *Geographic and Energy Aware Routing (GEAR)* protocol [39] – is based on the energy and location of sensor nodes, which are on their transmission paths towards their targeted regions. Also, focused on WSNs energy optimization. Transmitted packets are targeted for a region. Its implementation facilitates the trade-off between energy and distance.

Table 2.4 discusses the advantages and disadvantages of the protocols discussed above. These protocol advantages and disadvantages are discussed in terms of how they perform concerning network resources and energy utilization. These protocols have different capabilities due to their design and perform differently for various network scenarios and structures. The purpose for developing resource-aware and energy efficient routing protocols is to ultimately improve sensor data transmission and network capacity without drawing excessive energy from sensor nodes and not strain network resources. Some routing protocols such as info-centric, location and multi-path based are cautious about the data being processed, the location of the sensor nodes and the transmission capacity of different nodes. The purpose of this is to optimize the sensor network's capacity in terms of scalability and reliability. The availability of network services and resources is extremely critical since is directly impacts the performance of the whole system.

**Table 2.4:** Advantages and disadvantages of WSN protocols

<b>Protocol</b>	<b>Advantages</b>	<b>Disadvantages</b>
MAC	High data rates, which as a result, has high throughput. Supports commonly used transmission protocols (TCP and IP), i.e. has advantages for internet connectivity.	Energy wastage due to packets collision, which have to be re-transmitted. Sensor nodes tend to receive packets destined to other nodes hence increase energy consumption. MAC protocols are mainly affected by packet-overhead, and thus energy utilization increases.
LEACH	Increases Network Lifetime Energy saving due to aggregation by CHs	Dynamic clustering brings extra overhead. Cannot ensure real load balancing in the case of sensor nodes with different amounts of initial energy. Nodes use single-hop communication.
AODV	Connection setup delay is less. Routes are established on demand	Intermediate nodes can lead to inconsistent routes. Multiple route request packets in response to a single route request packet can lead to heavy control overhead.
SPIN	Topological changes are localised	Not certain that the data will reach the destination or not. If the nodes that are interested in the data are far away from the source node and the nodes between source and destination are not interested in that data, such data will not be delivered to the destination at all. Idle nodes consume energy.
GEAR	Increases network lifetime. Helps in balancing energy consumption	Not cautious of QoS, and a result, incurs high overhead. Does not support network scalability. Also poor in terms of sensor mobility.

## 2.3 WSN: APPLICATIONS AND SYSTEM CHALLENGES

### 2.3.1 WSN APPLICATION SYSTEMS

Even though WSNs are popular due to their simplicity of deployment and cost effectiveness, managing them is a difficult task pointing to their resource-constrained nature. However,

applications of WSNs have continued to grow regardless of the challenges experienced in their use. WSNs are envisioned to be deployed on large scale, as millions of wireless sensor nodes will be working cooperatively to transmit critical data and as well being connected to the internet as an effort for the realization of internet of things (IoT).

WSN technologies have been successfully developed and implemented in a wide range of applications as in [40], [41], [42]. These technologies are now commercialized according to their range and specificity of application. Table 2.5 illustrates some of the commonly used systems or technologies developed for critical applications.

**Table 2.5:** Common WSN applications and their technology systems.

<b>Application</b>	<b>Condition</b>	<b>Parameters</b>	<b>Technologies</b>
Agriculture	Planting: Soil preparations	Soil: pH, Nutrients, Humidity, etc.	AgroSense, AgriServe, etc.
	Animals: Moving patterns and behavior.	Position, Animal Tracking, etc.	ZebraNet, WiSense
Military	Surveillance [Safety and Security]	Infrared, Motion, etc.	Ultra-Stable Tripods, Radar, EcoKit, etc.
Health	Operative or Intensive Care	Heartbeat, Pulse, Rate, BP, Temperature, etc.	iMONNIT, Wisense, Autonomous Systems and Biomechanics
Infrastructure	Surveillance, Control and Prediction, Measurements, Maintenance	Temperature, Motion, Infrared, Vibration, Strain, Stress, Air-Flow	Camera Systems, SensoNode, VIBCODE Transducer, etc.

Transportation, Automotive	Asset tracking, Presence	Radio Frequency, Accelerometer, etc.	Trackers, RFIDs, Smart Logistics, etc.
Water Quality	Observation and Control	pH, Iron, Nutrients, Color, etc.	Libelium Smart Water, etc.

Systems such as those used in health, military and environmental monitoring are some of the most commercialized technologies around the area WSNs. These technologies play a pivotal role in everyday living, as some of them are mission critical systems. The wide adoption of WSN technologies is since these systems are affordable and not difficult to implement. However, the cost of implementing such technologies depends on their range (in terms of size and purpose) of operation. Due to factors, such as their affordability, ease of expansion and not conforming to a particular network topology, WSN technologies could be adopted and optimized as the fundamental structure for the IoT network paradigm.

### 2.3.2 WSN SYSTEM CHALLENGES

Commonly reported challenges include; sensor node energy limitations, low memory and processing capacity, low channel bandwidth and being application specific. A lot of work has been done particularly on energy limitations as an effort to improve the node energy utilization on WSNs. In addition to that, other methods of energy harvesting have been proposed as a means to leverage this limitation on sensor nodes. However, the energy issue is still a serious challenge in WSNs since it affects the lifetime of a WSN directly. Other challenges include high latency in communication, traffic congestion and processing delays on intermediate nodes due to the packet-based routing nature of WSNs.

Based on the proposed strategy, another controller functionality could be to write container rules or applications that will operate on certain network devices such as routers and programmable sink nodes. This functionality could be used to apply software-focused strategies, which will be responsible for load balancing, ensuring resourceful bandwidth utilization as well as perform quick search operations for data transmission to complement efficient processing, thereby reducing high delays.

Since WSNs are energy and processing constrained, important considerations must be made before the deployment phase of the network. These considerations include, but not limited to; understanding phenomena/event requirements and the monitored environment, sensor network connectivity, the ability of the network to self-configure in adverse or intrusion situation, choice of wireless protocol depending on the sensor type to be used as protocols/standards differ in power usability, throughput and communication range. The study therefore, put emphasis towards these considerations, so that if necessary any software or resource-oriented limitation be accounted for during the network-planning phase. Understanding the monitored phenomena, will assist in deciding on relevant equipment to use, and also whether there are no radio frequency disturbances within the area.

## **2.4 SDN PARADIGM**

The SDN model stands out in today's systematic developments due to its architectural implementation, data and process-handling capabilities as this is envisioned to positively prepare a platform for network innovation [43]. This paradigm shift, implements a separate data control and traffic flow management to the underlying network components. This makes it possible for SDN strategies to be application compatible to many technologies such as for wired and wireless systems. SDN is a framework that decouples the control plane and the data plane to allow network administrators to automatically and dynamically manage and control numerous network devices, services, topology, traffic paths, QoS and packet handling policies using high level programming languages and Application Programming Interfaces (APIs) of their choice [44] .

The integration of SDN strategies to modern computing platforms serves as a potential direction to evolve them in that it allows capabilities for efficient control and understanding of all system processes and functionalities. Recent developments in the field of network computing are reporting tremendous achievements due to innovation capacities that are enabled by SDN. The effort of managing any system of networked components is a very difficult task. However, the application of SDN to such systems has advanced some level of hope towards the implementations of high-performance technologies that will be easy to

---

manage. Hence, with the right combination of tools and expertise, a lot can be achieved through SDN since it allows a platform for creativity.

Networking concepts and strategies such Network Function Virtualization (NFV), IoT and 5G have the same elements as that of the SDN architecture. Hence, these concepts can greatly benefit from the computing mechanisms proposed by the SDN project. It is possible to achieve some level of network resource orchestration, application automation and system function virtualization by correctly coupling and applying SDN strategies to modern and future network technologies. Previous network technologies relied primarily on circuit switching methods, whereas modern and future network technologies use both circuit and packet switching methods.

The use of OpenFlow communication standard and SDN strategies can greatly benefit from this tech technological perspective since traffic routing and network resource management are key in general networking systems. Effective network configuration and flexible network resource management can be realized since specific flow rules can be easily manipulated from the SDN controller, which directly communicate with OpenFlow capable devices. In [45] OpenFlow and SDN strategies were commended as potential solutions to deal with challenges experienced in wireless and switched networks towards the effort of realizing standard IoT platforms.

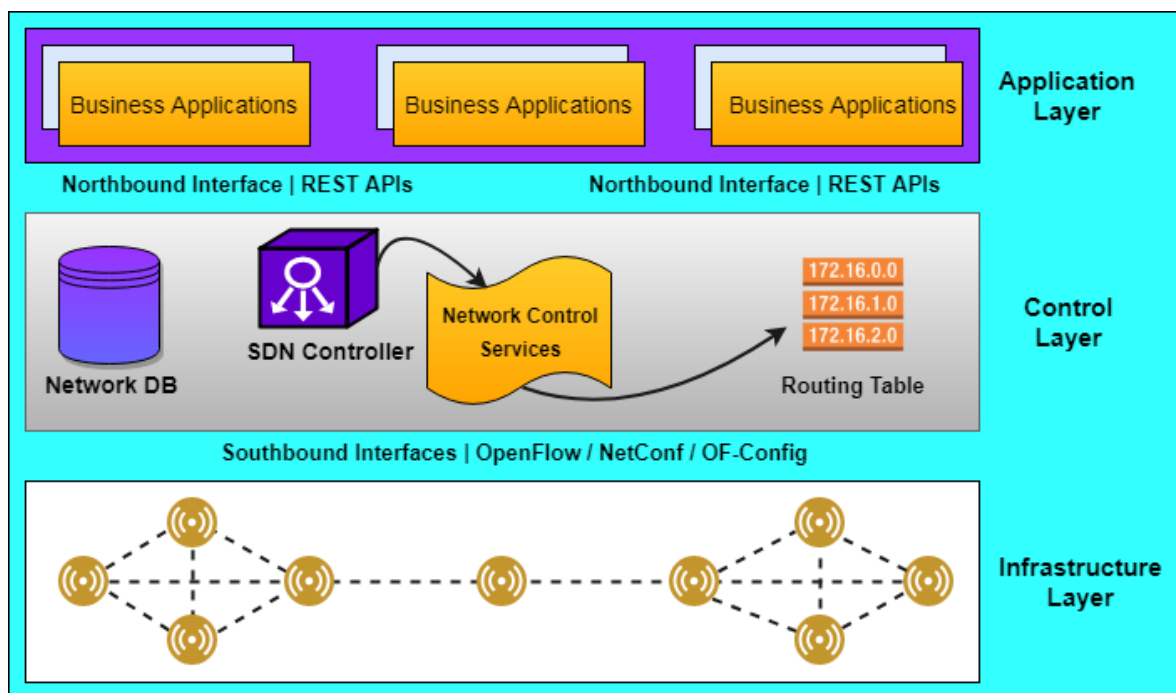
The candidate also describes an SDN approach using OpenFlow as shown in Figure 2.3. The OpenFlow, as developed and described by Open Networking Foundation (ONF); is a standard communications interface defined between the control and forwarding layers of an SDN architecture, which allows direct access to and manipulation of the forwarding plane of network devices such as routers and switches, both physical and virtual (hypervisor-based) [46]. OpenFlow extends the capability of SDN by providing a configuration platform to the SDN controller in implementing its control functionalities. The OpenFlow specification provides software procedures to operate on controller implemented packet flows, based on their respective flow tables. A header method for OpenFlow based packets is given on the code sample below.

```

struct ofp_header {
    big_uint8_t version; // OpenFlow protocol version.
    big_uint8_t type;    // An OF protocol constants.
    big_uint16_t length; // Length including the header.
    big_uint32_t xid;    // ID associated with this packet.
}; OFP_ASSERT(sizeof(struct ofp_header) == 8);

```

An SDN overview diagram with OpenFlow protocol is shown in Figure 2.3 below, as defined by ONF. The SDN controller implements flow entries according to network events and employ the OpenFlow protocol to manipulate the status and behaviours of these tables.



**Figure 2.3:** An overview of SDN architecture.

From Figure 2.3, the network controller is separated from the network data forwarding devices. The ONF based SDN architecture defines three distinctive layers, which are; the application layer, the control layer and the infrastructure layer. On these layers; end user



applications that uses the SDN communication service sits on the application layer through the northbound interface, the control layer is responsible for providing level functional control over the network forwarding behaviour through an open interface, and lastly the infrastructure layer, which consists of network components, that act on packet switching and forwarding [47].

Any network system that provides optimum performance and experience is implemented with critical considerations. From these considerations, sound network policies are developed. This is a very critical concern since any improvement in the network directly speaks to these considerations. The candidate consider that, any network design must support all its different entities; be it users, storage infrastructure, applications and services, system administrators, etc. Most importantly, it must be available and secured. It must also be able to adapt to all critical changes within the environment.

Traditional strategies of networking have proved to be capable and reliable for over the years. This points to the fact that large, medium and small size network architectures have been successfully implemented using these strategies to date. In today's technological demand for powerful systems, traditional ways of network computing face a tough time in terms of efficiency and management. In this regard network innovation becomes difficult to realize in traditional networks. This is mainly because, in addition to the network devices being proprietary, the network control is implemented on each forwarding device, thereby making it hard to access and operate on the functionalities of these hardware.

On the contrary, SDN approach implements the network controlling mechanism separate from the data forwarding plane. The network control is performed on the centralized SDN controller whereas data forwarding is performed by the adjacent network devices. A programmable switch such as an OpenFlow switch is directly connected to the SDN controller for programmable instructions and process manipulation. Therefore, the fundamental networking strategies as implemented by both these network architectures, directly reflects their operational nature.

Some arguments still exist in terms of management and performance of SDN based networks and traditional network. These arguments are mainly looked at as comparisons between these

two network technologies as an effort to highlight which technology brings benefits. These arguments are also further interpreted to the act of making decisions as to which technology could be used and for what intent that technology could be used. Table 2.6 gives these arguments in terms of comparisons.

**Table 2.6:** Comparison between an SDN Based Network and a Traditional Network.

<b>Potential</b>	<b>SDN</b>	<b>Traditional (Non-SDN) Network</b>
Operate Features	Separation of the control and data forwarding planes. Easily controlled and deployed.	Hard-structured and logically coupled operations. Very complex control.
Implementation	Fast and easily implemented. Adapt to the need of application environment.	Long implementation time. Operate on dedicated environments.
Stability	Currently unstable with few technological supports.	Stable with great network support.
Architecture	Software oriented with resource customizable features. Centralized network intelligence.	Hard-structured and operated on dedicated and proprietary devices.
Configuration	Configuration can be largely done remotely. Software monitoring and operation done centrally.	Network devices need to be configured directly and individually.
Management	Easily managed using APIs. Can be easily modified depending on the network demand.	Difficult to manage as network devices are solely proprietary.
Maintenance	Can be easily maintained as new services or network upgrades can be done easily without affecting the whole network.	Difficult to maintain as the whole network might be affected by a small change in the network.

Error Checking	Easy to check network errors as software module are dedicated to do this. Therefore, errors can be quickly isolated easily from the network.	Error checking is extremely time consuming and difficult to isolate as operators have to the check the whole network even for small errors.
Novelty	Network innovation becomes easy as new features of the network could be attached to the existing infrastructure. Does not need to revisit the whole network processing.	A simple change will require a serious study and understanding of the whole network structural design. Therefore, innovation is possible but difficult to implement.

### 2.4.1 SDN Controllers

As described in [46], OpenFlow is ONF's standardization communication protocol for SDN based control and data forwarding layers, which provides access to the data forwarding plane of the adjacent network resources (switches/routers) through software abstraction. The candidate takes this capability (of OpenFlow) to e of extreme importance pointing to network computing innovation, as this aspect will ever be exploited for future improvements in the overall networking sphere. Of late, as an effort towards network computing improvements, recent manufacturing of networking devices (routers, switches, etc.) are OpenFlow capable as it is still the recently exploited standard. On the other hand, SDN oriented controllers have been developed by different institution and companies according to their preferences and specifications.

Even though these controllers differ in design, operation and platform; they at least need to be OpenFlow capable for SDN based computing. Table 2.7 shows some popular SDN controllers as well as their specific system design. Other improvements on these controllers are still anticipated by various network computing developers as more and more technical network demands are experienced. This is of paramount importance as these systems needs to be equally capable of supporting the ever-challenging network architecture and support.

**Table 2.7:** Popular SDN Controllers and Their Descriptions.

Developer/Owner	SDN Controller	Built Language	Architecture
Stanford University	Beacon	Java	Centralized
Big Switch Networks	Floodlight	Java	Distributed, multi-level
HP	HP VAN SDN	Java	Distributed
ON.LAB/Stanford Univ.	FlowVisor	JavaScript/JSON	Distributed, multi-level
Nicira, NTT, Google	Onix	C, Python	Distributed, multi-level
Nicira/VMware	NOX	C++	Distributed, multi-level
USENIX / --	NOX-MT	C++	Distributed, multithreaded
Cisco Systems	NodeFlow	JavaScript	Distributed
Juniper Networks	OpenContrail	Python, C++, Java	Distributed, multi-level
OpenDayLight Project	OpenDayLight	Java	Centralized, multi-level
OpenMUL Foundation	OpenMUL	C	Distributed, multithreaded
ON.LAB	ONOS	Java	Distributed. multi-level
NTT Communications	Ryu	Python	Distributed, multi-level
NOX Repo	POX	Python	Distributed, multithreaded
NEC	ProgrammableFlow	C	Distributed, multithreaded
ECI Telecom	SMArtLight	Java	Distributed, multi-level
NEC	Trema	C, Ruby	Distributed, multi-level

## 2.5 SDN FOR FLEXIBLE RESOURCE MANAGEMENT IN IOT NETWORKS

The current technological space sees IoT as an enabler for efficient network availability, high performance and scalability with possibilities for network application innovation. This relates to networking aspects such as; rich resources availability, fast network connectivity and data access, support for heterogeneous devices and reliable applications. However, there are still a lot of issues such as; computing, connectivity, communications, caching and security for billions of heterogeneous objects that needs to be considered prior the realization IoT solutions that could be confirmed through technological applications. Numerous works that incorporates the IoT paradigm into everyday systems such as in [48], [49], [50], [51] already exist. This work also looks into possible opportunities for realizing applicable and

efficient IoT solutions using SDN, through network flexibility and enhanced resource management. It further looks into other network functionality enhancements using NFV for secure and robust IoT platforms. As a result, relevant SDN strategies are critically analysed and aligned to IoT design specifications in terms of its vast objectives of technological operation. Related achievements are surveyed to refer to IoT application systems and to build technical arguments, which will contribute to the development of such application systems. The impact of SDN as a possible driver for real time resource allocation, interoperability and network innovation for modern and future IoT applications is then provided.

Due to estimations around a number of devices (or things) that future IoT systems are predicted to facilitate, it is certain that IoT networks will become more complex, extremely difficult to manage and highly susceptible to security threats. Heterogeneity of these devices poses communication challenges to the IoT paradigm. However, some researchers proposed standardization to select the suitable communication standards, yet standardization also presents problems since there are many standards some of which sometimes evolve. Due to this standardization problem, users may deal with an increasing number of heterogeneous devices, which are incapable of communicating with one another [52]. In that regard, an SDN-based solution for managing the heterogeneity of IoT devices and networks was proposed in [53]. This solution made use of Docker techniques implemented on IoT devices for communication through SDN-enabled networks. Their experimental results suggested that the traffic flow remained stable for different connectivity scenarios. However, the delay for Docker-Docker connectivity was recorded as the highest. On the other hand, in [54] the results showed that Dockers consume low power and the overall impact of the Docker virtualization layer in terms of performance (delay) is almost negligible.

In IoT networks, numerous applications with various requirements will be sharing the same network resource and the heterogeneity of these applications and the connected networks make resource management extremely challenging. Some researchers in [55] introduced the Device cloud approach, which was intended to solve the resource management issue by illuminating the static application domain dependent bindings between users and devices by applying cloud computing concepts such as resource pooling to the IoT paradigm.

Today's computing epoch is challenged by an increasing demand for intelligent systems and applications due to considerations for the advancement and convenience of consumers. Information systems such as data analytics, shared computing resources, control systems, big data support, visualizations, system audits, artificial intelligence, etc. are a necessity to everyday life of consumers. However, to realize such information systems that can be supported by most system providers and be systemically allocated or shared for vast computing needs, requires the best platforms and technologies to cater for their design and implementation. WSN application technologies holds the potential to be implemented as blueprint architectures for IoT systems. This is due to how these wireless technologies are developed, implemented and deployed. Sensor nodes are placed ubiquitously to detect and act on phenomena or event data for various applications. Thus, sensor devices, promotes intelligent forms of computing capabilities to support IoT systems provided they are enhanced for IoT functionalities or operations.

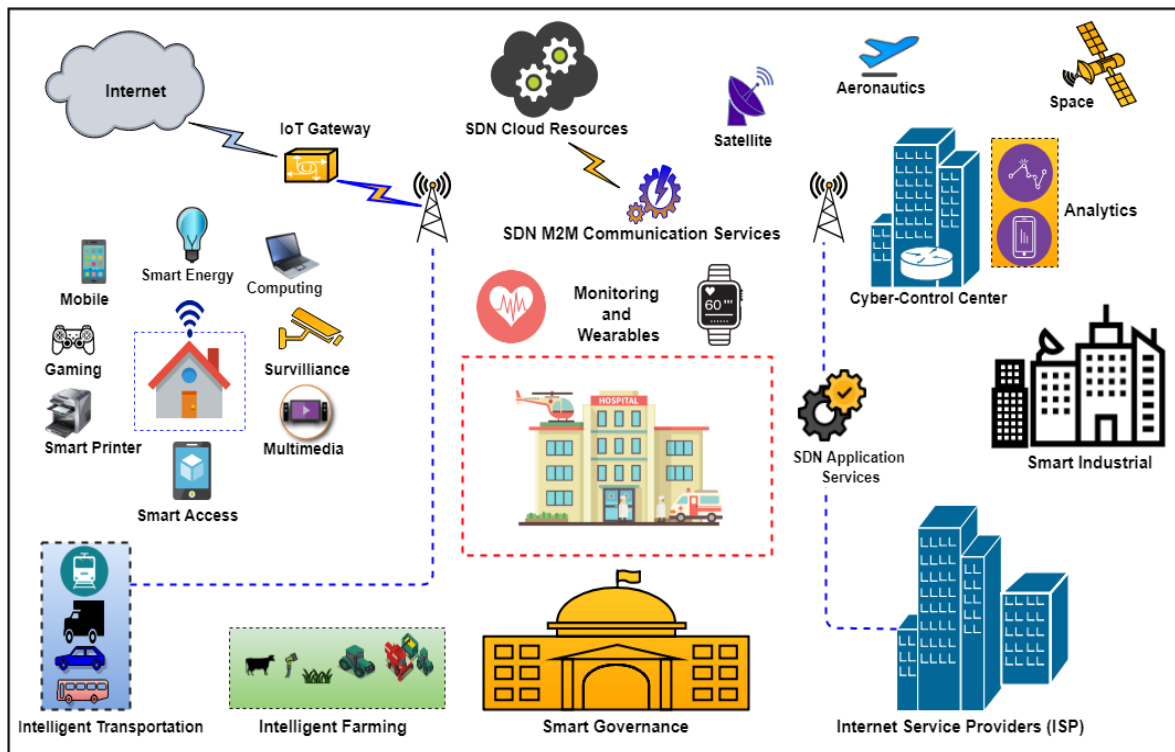
In the literature, SDN is categorized as a potential solution to a simple, flexible and manageable IoT. SDN introduces significant flexibility for resource management and adaptation of network functions. In [56], an SDN based programmable hybrid architecture, which can select whether to use centralized or semi-distributed control processes by considering the various heterogeneous parameters in an IoT system was proposed. They introduced three control levels: Principal, secondary and local controllers for adaptation into heterogeneous technologies, whereby local controllers were selected using connected domain set and fuzzy set algorithms. Their result suggested that their proposed selection strategy achieved satisfactory results concerning local controller selection. In [57] the authors introduced a novel layered SDN controller into the IoT Multi Network Information Architecture. The SDN controller's layered architecture enabled flexibility and efficient management of tasks, network and resources by incorporating and supporting commands to differentiate flow scheduling over task levels, exploited network calculus, and genetic algorithms to enhance usage of current IoT network opportunities.

Current computing technologies perform to some extent and somewhere lacks robustness, reliability and scalability to support high computing demands. Today's ideal computing or information system is one that can efficiently support and provide various applications and

---

services for as long as there is connectivity and efficient network resource management. Such systems must incorporate the best architecture and capabilities to support any device, application, services and system requests, service provider, etc. ubiquitously at any time with powerful resources that can be easily accessible. Such systems can mostly be realized through IoT platforms that incorporates different but rich applications and resources that can support any application system.

IoT refers to a network-computing paradigm where unrelated objects or systems are systematically connected, communicate and compute in accordance or response to request or operations commanded from them. These objects uses smart communication capabilities to transmit data, requests or commands to one another or other systems through Machine-to-Machine (M2M) communication technology. IoT is envisioned to facilitate a connection of 50 billion devices in 2020 [58], wherein different applications or systems will be interlinked through this connection to share IoT platform resources. IoT potential applications includes; home automation (with smart energy management techniques), smart process control for manufacturing and other industries, smart grids and cities, smart health systems, smart devices and automotive. However, IoT specifications or standards are still not clear in today's agenda because its stage and different views in terms of proposals. There are still a lot of discussions amongst engineers and professionals around IoT standards and implementation models. This work presents an IoT infrastructure that uses SDWSN strategies as its key features to simplify network resource management. Figure 2.4 illustrates an overview architecture that features SDN as its core component.



**Figure 2.4:** IoT architecture with SDN as its core feature.

In Figure 2.4, possible connected services and solutions that can be driven by IoT and other networking paradigms such as SDN and cloud are shown. SDN can play a very crucial role in enabling IoT related services such as; object communication, provide opportunities to create new network protocols, provide some level of abstraction to interface with different objects, enable software security features, create special high demand applications or services for emergency purposes, etc. High or special attention applications can be; emergency alerts, high bandwidth provisions for crucial big data, authoritative alerts for cyber or other forms of crime. IoT can also greatly benefit on SDN cloud-based solutions and services whereby, IoT objects can easily share rich cloud resources. SDN cloud resources could be abstracted services or applications that can be called by objects during intensive compute loads. Cloud services can also benefit big data in a manner that data becomes quickly reachable and available. Service providers can also benefit from SDN oriented operations that can be used to optimize data centre computations. This architecture can also profit government and transportation in delivering services.



### 2.5.1 Application Challenges in IoT Networks

Recent technological developments have taken a direction to develop or design IoT platforms that will facilitate a connection and resource support for substantial number of “things” which will eventually make living simple. The foundation for IoT systems is to have platforms that can facilitate inter-connections to many but different devices or applications regardless of their nature of design. This technological view is aimed at connecting every electrically powered objects to the internet to communicate and share relevant data in accordance to any prompted requirement or alert, systematic command or service request. However, to provide such connection-oriented resources and services to a large number (billions) of devices that are different in terms of design and mode of operation is a difficult task. A typical example could be an electrical distribution board (DB), communicating with an owner’s mobile phone, to provide an alert that there is an electrical fault somewhere in a house wiring. Another one can be, a retail IoT system sending alerts or messages to regular customers’ mobile devices about current product prices with recent stock updates and competing prices from various retailer on the same products.

Since IoT systems aims at connecting billions of heterogeneous devices which have different computing capabilities in terms of communication technologies, computation resources, energy source, security capabilities, manufacturer’s standards etc. it makes interoperability between connected heterogeneous devices an extremely challenging task to perform [38]. Without mutual communication standards or protocols that are suitable for all the devices participating in these networks the connected devices may end up operational but rarely or unable to communicate with each other. This, challenge the concept of IoT since IoT aims at connecting various devices that will be capable of communicating, working together to bring essential services to the network users.

Dealing with heterogeneous devices also introduces security and privacy challenges to IoT systems due to the following reasons:

- Communication between heterogeneous devices with different capabilities introduces complications in designing security mechanism.
- Any device connecting to the internet is vulnerable to security threats.

- 
- In IoT navigability, accessibility and global connectivity are a requirement this gives anyone even malicious users permission to access any IoT device any time.
  - Interactions between heterogeneous devices rely on lightweight wireless transmissions such as radio signals/waves, which are extremely vulnerable to attacks [42], [49] and [50].

The evolution of the web, as one of the elements of IoT has introduced a vast number of applications with different bandwidth, scalability, responsiveness, security, availability, reliability and effectiveness requirements. This brings forth the issue of Quality of Service (QoS) since these applications, the devices connected to them and users require a level of quality assurance from the network. At the moment, the implementation of QoS to IoT platforms is one of the most trending and yet challenging topics among researchers since QoS provisioning in IoT systems entails meeting those QoS requirements of every user and application that uses information systems connected in the IoT system [59] and [60].

The management of network resources also affects QoS provisioning. This simply means, the effective utilization of network resources impacts positively on QoS, which is necessary for every network. IoT systems are perceived to involve billions of devices with various capabilities, complexities and constraints. Therefore, this makes resource management in IoT extremely difficult [59]. The benefits of effective resource management in IoT is not only to reduce power consumption but also to prolong the lifetime of constraint networks and provide real-time monitoring and real-time fault detection and correction. Hence, resource management should be treated as a critical aspect of IoT systems. Table 2.8 provides a description of immediate IoT technological systems challenges that need to be well accounted for during the planning phase for such systems. These challenges relate to any application system but here the emphasis is directed to IoT platforms or networks. This work tries to contribute with an understanding on these challenges and make efforts to provide advices or considerations that could be applied towards successful implementations of IoT systems. In addition to these technological concerns, the study also provides accountability considerations that needs to be considered when designing dedicated IoT devices in terms of ethics and regulations to avoid endangering lives of consumers.

**Table 2.8:** Immediate IoT systems challenges with technological descriptions.

<b>Computing Challenges</b>	<b>Descriptions</b>	<b>Technological Options</b>	<b>System Accountability</b>
<b>Connectivity</b>	- An IoT systems aims at interconnecting different but many objects. These objects will collect and perform various operations on different data for as long as they are connected.	- Suitable and manageable data access and operational modes needs to be applied to these different objects to allow connectivity.	- Device and system level access techniques must be practiced in ensuring proper connectivity.
<b>Interoperability and Standards</b>	- A crucial part of the IoT system is to ensure that these different objects communicate to form a successful system. This technology aspect speaks directly to all communication standards that are applicable to ensure this compatibility.	- Traditional methods or techniques needs to be optimized to meet new operational requirements. - New dedicated procedures need to be developed to cater for new applications or operations.	- Efficient and smart communication protocols must be ensured.
<b>Architecture</b>	- Different information systems of the IoT platform will be responsible for supporting many objects of unique designs. The underlying network needs to have intelligent topologies and taxonomies that will ensure smooth computation amongst these objects.	- A network can be a combination of different hosted frameworks. - These frameworks will support various devices and information systems to ensure that each level or part of the network is afforded adequate resources.	- System design must allow flexibility for any future change.
<b>Security</b>	- IoT will certainly support massive networks with billions of objects collecting and sharing data to advance consumer convenience. Security is extremely crucial as these objects needs to be guarded with tough security policies to ensure that they are not vulnerable, which will then lead to a compromise of the entire system or exploitation of consumers.	- The network could be divide into dedicated parts to ensure that each part is supported with conforming security policies. To ensure enough security, virtual security applications can be implemented on various parts of the network using software paradigms such as SDN and Network Function Virtualization (NFV).	- Security must be treated with high consideration to ensure relevance in humanity.

---

<b>Ethics and Regulations</b>	- Any design must follow, conform and comply with certain design principles. This is to ensure that, human safety is not compromised. Device engineering needs to follow design regulations to ensure compliance.	- Comprehensive and appropriate principles could be converted to design strategies. These principles must be socially appropriate.	- Computer experts, political and social authorities need to work together to provide guidelines for design and accountability measures.
-------------------------------	---	--	--

---

### 2.5.2 SDN Benefits for IoT Networks

Advances in today's network computing has stimulated a lot of possibilities for available, highly responsive and robust technologies. These technologies span from mobile devices, in-house appliances and cloud-based services or applications. As a result, a lot of industrial and research proposals and developments have surfaced in an effort to meet high application and services demands at least in any computing field. The purpose of all this is to advance everyday life of consumers with affordable yet powerful systems or technologies. Efforts to develop high performance computing platforms is therefore key in today's era of living. These efforts are made along various applications such as in; medical practice, general surveillance, multimedia purposes, crime combating, manufacturing and control systems, transportation, etc. that requires powerful networking systems to achieve on time and best results. This has led to the advances to develop IoT platforms that will incorporate abundant resources and services to objects that they will inter-connect.

Given IoT platform challenges expressed in subsection 2.5.1, this work looks into benefits that could be provided or enabled by SDN. To make a comment, at this moment it will be biased to suggest that SDN can fully provide solutions to reflected IoT challenges. Therefore, it has to be emphasized that, this work gives advices regarding benefits that could be achieved using SDN strategies, by providing details as to how such technological aspects can be realized. In addition to the challenges as noted, it must also be recalled that, there are currently no universal IoT structures or standards that can be followed to design and implement all-inclusive IoT systems. This is due to the fact that, current developments are still in pilot stages or at proposal level. Critical systematic analysis and proper planning still

needs to be embarked on with mindful considerations around all aspects of the envisioned IoT platforms. In terms of what SDN can provide or enable in IoT systems, below are systematic opportunities that can be expressed as SDN driven benefits:

1. SDN strategies can be applied to IoT systems to improve sharing or allocation of resources by means of creating application specific service that could be accessed by IoT objects. This will promote adequate resource sharing and as a result improve the overall network performance and capability.
2. SDN strategies can be applied to IoT communication protocols to enhance efficient data access from various network objects and also promote smart device functionalities in terms of data wireless aggregation [61].
3. IoT platforms can greatly benefit from the SDN programmability to create software-oriented applications, policies or protocols that can form a layer of abstracted services that can be used by all authorised underlying IoT information systems and devices. This will allow network flexibility [57] and therefore promote network innovation as well as enable smooth interfacing with IoT heterogeneous objects.
4. SDN can also provide virtual security applications or services to the IoT network by implementing these services in dedicated parts of the network. This will enable distributed security solutions to distinct parts of the system and therefore, creates opportunities to develop, device level security applications.
5. SDN strategies can be used to facilitate QoS provisioning within the IoT network through the implementation of software-oriented priority mechanisms which are QoS aware. Compute intensive tasks that requires large resources can benefit through this by means of being afforded computing advantages.

With the right systematic planning and proper implementation of SDN strategies, IoT platforms can be massively enhanced with opportunities for rich network innovations. Correct and sound planning will also convert to efficient IoT resource management. Therefore, a well-balanced and carefully designed IoT network will support vast applications and be able to allow opportunities for catering to any technological application.

### 2.5.3 In-Network Cache Computing for IoT Objects

The core functionality of any computing network is to successfully acquire/sense activities, process or events and perform data transmission or queries processing internally or from external objects, that forms part of the network. This activity involves a lot of processes that entails network services such as; necessary routing protocols, identifying shortest paths for fast data transmission, efficient content delivery mechanisms, etc. that forms the core of the system. One of the essential services that is provided by both the network's hardware and software is caching. This network service is critical for fast data retrieval and making data sufficiently available for related queries. Likewise, participating IoT objects must have some capabilities to provide enough caching operations to support increased speeds of data access and retrieval. However, most IoT network objects are resource constrained, which creates challenges for in-network caching operations since these objects have limited networking lifetime, processing and memory capacities. As a result, such resource-constrained objects would not sufficiently perform in-network cache functions. This means, most IoT objects will perform poor cache functions, due to their capacity limitation. A critical question regarding this would therefore be how would these objects efficiently perform in-network caching, for efficient networking responses?

The strategic model of SDN is to exploit computing networks using high level programming languages to create specific network service applications that performs special operations. Moreover, IoT systems or platforms, provides the opportunity to develop network services and applications to meet its high resource computing demands. Therefore, SDN strategies can be aligned with IoT application opportunities to create powerful computing solutions that are efficient. In particular, SDN can be used to advance in-network caching through the implementation of software-based routines to facilitate efficient content caching for increased data availability. Container based software rules can be implemented on special IoT resource components/platforms for the provisions of efficient cache functionalities. These dedicated components could be installed or implemented on network branches or subnets. This will also advance network innovation opportunities as would be leveraged from the SDN strategy. Another opportunity is to implement secure lightweight autonomous

frameworks on the network to facilitate computing IoT objects with cache resources to improve access to historical content with maximum speed.

#### **2.5.4 IoT Objects Communication and Data Integrity**

On the basis that these billions of communicating or participating objects will mostly be part of public networks, data communication becomes extremely crucial. An observation is that security and data communication cannot be treated separately in computing networks. Therefore, any development of an IoT system or platform must simultaneously relate any communication protocol to a relevant security policy. It is also limiting to assume that public networks' security is enough, since security policies from such networks are not to be trusted with confidential data. System owners need to develop own security applications or policies to ensure authoritative device communication. Objects or connected "things" must use unique keys to advance strong authentication mechanisms in their communication. The network must use high encryption policies that must be applied throughout the network so as to enforce secure communication. These procedures must be employed to avoid unauthorized data access which will lead to extortion and system disaster. Data must be guarded with the most advanced and updated security policies to maintain data integrity as this is the fundamental core business of network computing.

Application protocols such as XMPP (Extensible Messaging and Presence Protocol), MQTT (Message Queuing Telemetry Transport) and CoAP (Constrained Application Protocol) are commonly used in IoT M2M communication. These protocols are used to connect networking objects to a dedicated control point or a server. IoT uses these protocols together with network resource platforms such as the cloud to facilitate M2M communication amongst its participating devices. SDN strategies can be used to enhance these application protocols by integrating their functionalities with software-oriented application services to allow rich service abstractions. SDN programmability injections can be applied to these application protocols' message passing as a means to advance network flexibility. Another option would be the use of SDN strategies to install subroutine functions within the protocol's message carrier suite to allow network innovation. However, connecting different IoT objects to a dedicated resource platform must be done carefully as this could cause

scalability issues due to heavy data load. To avoid this, resource platforms such as servers or the cloud, could be largely distributed to allow rich data access and enough redundancy.

### **2.5.5 SDN Enabling Secure WSN Based IoT Systems**

Sensor networks or devices are characterized with rich capabilities to form the foundation for ubiquitous IoT systems [62] as explained earlier. Thus, future WSN application technologies will be developed with IoT based directives. Therefore, for heterogeneous or ubiquitous IoT platforms, WSNs architectures and deployments will form part of the IoT developmental strategies. Any computing system needs some level of security as a means to advance its functional or operational purpose. However, for a system to be fully secured, efficient security methods need to be applied on the whole system architect to ensure a protected and reliable computing platform. IoT systems presents new security systems' challenges to modern computing networks, since it aims to facilitate a connection of technologically related and unrelated computing objects. To successfully secure IoT systems, intelligent security protocols or applications must be developed. This means, each communicating object must be associated with a security mechanism to prevent malicious actions on the system. In addition to this, it must be common that, security is not a one-off activity but an iterative process that needs to be maintained to ensure system integrity at all times.

WSNs as principal elements of the IoT paradigm also present flexible control, resource orchestration, sensor network resource-virtualization, sensor node configuration and management challenges to IoT [63]. As computing operations in sensor nodes are different from those in traditional switched network devices, [64] proposed the following components with suitable and specific functionalities for an SDWSN or SDIoT infrastructure:

- VSensor, which is a virtual sensor.
- SDIoT controller to suit the constrained nature of WSNs.
- A SFlow, which is a protocol designed in the same manner as OpenFlow yet catering specifically to WSN requirements.

To exploit the advantages of SDN in WSNs, authors in [65] proposed an Efficient Software Defined Wireless Sensor (ESD-WSN) architecture whereby the controller used dynamic



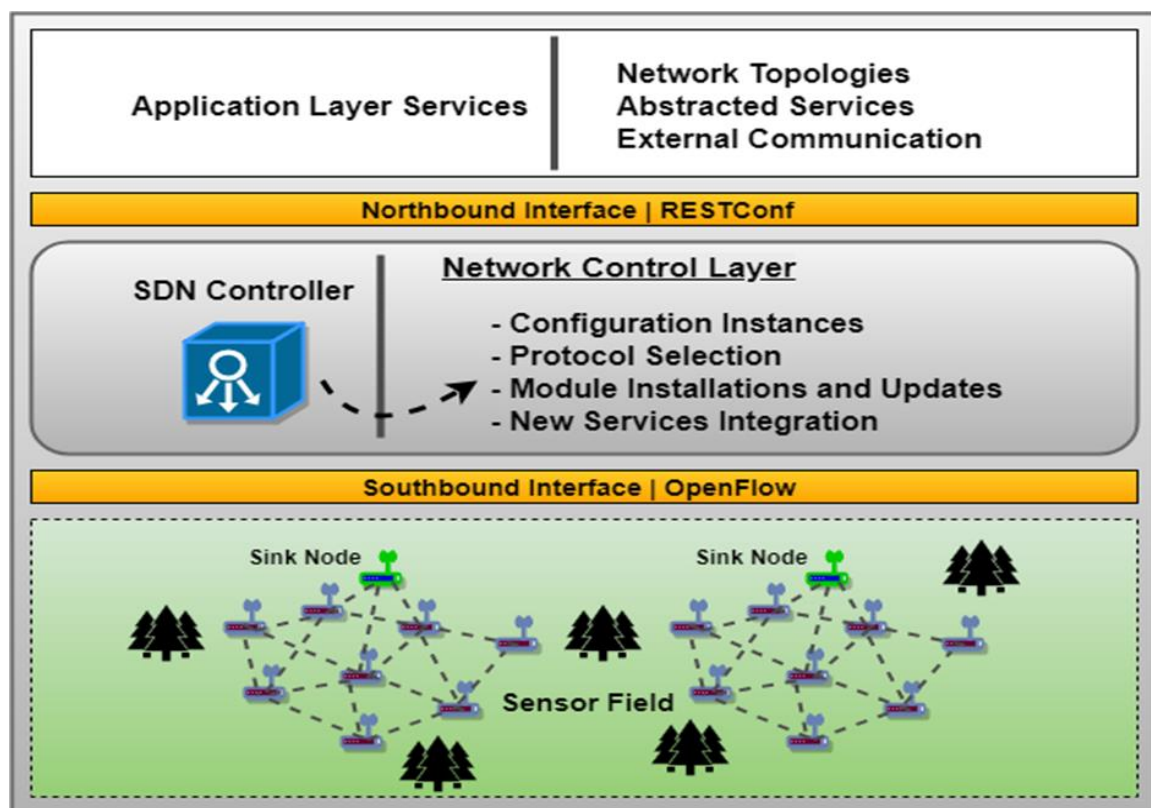
selection strategies to optimally select certain nodes as proxies for processing and aggregating the control traffic. The results of their experimentations suggested that the proposed architecture achieves significant performance improvements compared to SDN-WISE.

Upon appropriate system planning, SDN strategies can be applied to IoT systems to ensure some level of security. Leveraging from the SDN paradigm, central security application services can be implemented on the controller and be used to remotely operate on parts of the network that needs administrative attention. Operations such as remote security patches, updates, process abstractions and event logging inspections can be implemented on a highly secured and resourced central controller. There has to be a way to perform these operations without compromising critical network functions. The SDN controller could also have isolated software-oriented application services that are responsible for reverting to the last normal system state in case of any unauthorised event in the system. The SDN controller must have the capability to verify and prevent network events that can alter the state of the network system. Each service or application event could be linked to a controller abstracted services for control resources. However, remote control needs to be only applicable at the central controller point to avoid intrusion and data manipulation activities. In actuality, SDN strategies could be used to develop local security application since standard provided security services are prone to attacks or known to some people, thereby creating high security risks.

## **2.6 SDWSN: APPLICATION IMPROVEMENTS FOR WSN SYSTEMS**

The application of SDN strategies in WSNs seems to be evolving their functionality and hence their performance. The integration of these strategies forms the SDWSN framework. This framework uses SDN programmability techniques to enhance WSNs application systems performances in terms of processing and functionality [66]. Some level of flexibility and system adaptability have been achieved using SDN techniques to WSNs [56]. However, the effort to improve their capability is still an ongoing work. Robustness and accuracy in terms of system operation and data presentation is still a fundamental aspect to enhance them.

In Figure 2.5, a SDWSN framework is presented to provide an understanding of different network services that can be applied, depending on different system demands. This framework also shows various system functionalities that forms its architectural theme and those that can be performed in the underlying sensor network. The SDN controller is well resourced and all applications or services running on the controller have the advantage of this aspect of the system. The controller is directly connected to the network's database for improved networking resource access and information storage. The SDN controller access and implement top layer applications through the northbound interface.



**Figure 2.5:** An overview of the SDWSN Framework

Sensor aggregated data is relayed through OpenFlow capable switches to the central SDN controller, where all compute intensive tasks are operated. System administrator can accordingly operate on sensed data to support critical system requirements. This setup also promotes network application creativity through abstracted network services. Figure 2.5, shows a sensor field with sensor nodes deployed for environmental sensing. Sensed data is

---

relayed through OpenFlow channels to the controller for authoritative purposes. System administrator can remotely operate the network and install network functionalities that are relevant for every job that requires attention.

The main focus of SDN is to introduce application and functional programmability onto network computing systems. Hence, the application of SDN to WSNs has formulated the SDWSN approach. The principal idea of SDWSN is to enable WSN application technologies to be more efficient in terms of processing and network management. This approach aims at simplifying wireless network and application policies in a way that these technologies can allow network innovation.

SDWSN refers to a networking paradigm that entails separating the control mechanism of an operational wireless sensor network from the data forwarding plane of the underlying vertically integrated network system. By introducing SDN techniques to WSNs, sensor nodes will only be performing forwarding tasks, whereas compute intensive tasks will be performed by the controller without affecting the overall energy consumption of the entire network. The benefit would be that the controller will not be resource constrained since it runs on a machine with more resources compared to the sensor nodes. Moreover, the controller will have the global view of the network and therefore being able to make optimal routing decisions based on the network's state information.

Today's network computing applications are faced with a high demand of powerful network functionalities and performance. This high demand is mainly motivated by advancements in computing spanning provider networks, data centers, cloud computing environments, etc. However, resource management of WSNs is a serious challenge, due to problems supplemental to them. Conversely, advancements and improvements in the manufacturing of computing or networking devices has awarded researchers and developers opportunities to experiment on these devices as an effort to enhance network computing capabilities and experience. An improvement position is that, since resource management and process control are so difficult in WSNs due to their sensor properties and structural complexity, SDN aims at bringing convenient control mechanisms to WSNs [67], [68] by simplifying the whole network infrastructure. Current studies indicate that this capability will allow the

---

controlling unit (the controller) to easily manage network resources as well as preparing a platform for other networking applications [67], [68].

As part of an anticipation to advance WSN application systems that are secure by employing SDN computing strategies, the proposed architecture intends to develop and implement security algorithms as part of the controller functionality. These algorithms could be implemented on each level of the network through some form of function virtualization. Since the course of developing powerful security is an ongoing process, vigorous forms of system testing could also be used to exploit the system of any security weaknesses. This process will then inform future implementations of powerful security features. SDWSN is also aimed at introducing a simplified implementation of a software-oriented strategies for broad WSN computing systems by means of moderating sensor and other network resources workload for as far as data traffic and computing is concerned.

Since every system has its own challenges, one critical challenge in SDWSN is the efficient assignment of spectrum resources to the virtual network, which then contributes to the spectrum resource problem [69]. However, dynamic programming and graph theory- based spectrum sharing algorithm is proposed in [69], wherein a performance improvement in this regard was achieved. In addition, with the current vision about IoT through WSN systems, it must be considered as to whether a relevant infrastructure can be realized based on SDWSN strategies.

The first attempt to combine SDN and WSN to solve the inherent problems experienced in WSNs was presented in [70]. The authors proposed a SDWSN architecture that clearly separates the control plane and the data plane. It also features Sensor OpenFlow (SOF) with emphasis to improve the adaptability of WSN applications for environmental monitoring, which is the main component of SDWSN as a standard communication protocol between the two planes. Their proposed model was reported to have significantly enhanced traffic load balancing as well as having reduced the amount of communication overhead. This approach has also reported significant improvements in the energy consumption of the system.

Later, SDWSN was used in Smart Grid WSNs [71] where it was said to have minimized complexity and power optimization in sensor nodes. The work in [72] proposed the use of

SDN as a means to smartly manage WSNs. They suggested that the controller be placed at the base station and their paper argues that smart management using SDN envisions solving some of the inherent problem that come with WSNs. One of the limitations experienced in WSNs is high sensor energy consumption. As a strategy, the concept of SDWSN intends to reduce high energy consumption on sensor nodes through some means of shifting other processing tasks to the SDN controller. In [73] SDN concepts are introduced into WSNs for smart management and energy optimisation. A generic framework for SDWSN where the controller is located at the base station is proposed.

An SDN based architecture for WSNs is proposed in [74]. Their simulations were performed in two cases where in one case a WSN without a controller was simulated and in the other a WSN with an SDN controller was simulated. The results of their simulations demonstrated that the implementation of the WSN with an SDN controller achieved 53% and 53.2% increase in received packets and throughput respectively as compared to the WSN implementation. However, the WSN implementation proved to be more power efficient than the WSN implementation with an SDN controller implementation which experienced a 3% increase in power consumption.

[75] attempted to solve this energy consumption issue by proposing a general SDWSN framework where the controller is placed at the base station and the sensor nodes performs only packet switching. They used Open- Flow as the core communication protocol between the controller and the switching elements. The proposed architecture minimized the energy consumption as anticipated but only to a certain level.

An SDWSN prototype was proposed to improve the adaptability and the energy efficiency for WSN monitoring systems [76]. In this work, an energy-efficient cognitive SDWSN prototype based on Reinforced Learning (RL) was developed for monitoring systems, wherein complex data fusion was managed centrally on the control plane while low complexity computations were performed on the data plane. Their results suggested that the proposed prototype had the ability to enhance self-adaptability of WSN environment monitoring with QoS. Moreover, by effectively hindering the transmission of unnecessary

loads; energy efficiency is significantly improved, the amount of cross-plane communications was minimized and hence, load balancing in SDWSN was improved.

An SDN based sleep scheduling algorithm SDN-ECCKN that reduces the total time of transmission of a network was proposed in [77]. Control nodes were selected to dynamically allocate different tasks. The control node selection was formulated as an NP-hard problem, taking the residual energy of the nodes and the transmission distance into consideration. As a result, an efficient particle swarm optimization algorithm was proposed to solve the NP-hard problem. The proposed optimization algorithm created a cluster structure in order to minimize the transmission distance and to enhance the energy consumption of the network. Their results suggested that the proposed algorithm outperformed its counterparts with respect to the network lifetime, number of functional nodes and number of isolated nodes and therefore, has potential to prolong the network lifetime. However, SDN-ECCKN experiences a high control overhead due to the continuous beacon messages that every node is required to send to the controller every time.

Mukherjee et al. [78] proposed a sleep scheduling scheme for SDWSNs. Their scheme employed low control overhead based flow entries in the flow table to address the control traffic overhead problem faced in SDWSNs. Their scheme does not send control request message from each sensor node to the controller and hence reduces the number of control messages circulating over the network to update the node's state in ECCKN based sleep scheduling algorithm. Their proposed scheme is said to have outperformed SDN-ECCKN with regards to minimizing the number of control messages circulating over the network.

Ejaz et al [79] presented an efficient SDN-based placement and energy transmitter scheduling for WSNs with RF energy harvesting. The SDN controller facilitated all the operations and placements to ensure that the energy transmitters reached a maximal number of nodes. The optimal activation of energy transmitters was achieved by employing the branch and bound algorithm.

In the work [80] and [81] Zeng et al proposed a SDN-based architecture for multi-tasking WSNs. They investigated designing an energy efficient reprogramming mechanism with guaranteed quality-of-sensing for sensing tasks. They specified that any type of optimization

across various applications can be achieved through globally controlled scheduling and quality-of-sensing. Their architecture defined the multi-application intents as a Mixed Integer Linear Program (MILP) problem with coverage, storage, and scheduling constraints. Moreover, they developed an online algorithm to consider the operational dynamics (application and sensor operations) during runtime. Their results illustrated that the online algorithm achieved almost the same optimization of energy as a global optimization but with less control overhead and rescheduling time.

Luo et al. [82] presented an energy efficient scheme (Modified SEECH) industrial wireless sensor networks which exploited monitoring infrastructures, controlling network topologies, and saving energy using SDN and NFV. Armed with the global view of the network, controlling the topology of IWSNs and node modes was accomplished through SDN programmability and instant deployment capabilities of NFV. Their results indicated that proposed scheme was more energy efficient than LEACH and SEECH with regards to data aggregation compression ratio.

A semidefinite programming (SDP) [83] oriented algorithm was proposed to manage energy consumption of a WSN. They define an energy management problem as an optimization issue given WSN QoS constraints to achieve efficient resource allocation. 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 reported that their proposed CABPA provide better performances in terms of energy consumption and the utilization of bandwidth. They also reported some trade-offs with regard to power allocation and bandwidth utilization.

Tomovic and Radusinovic [84] presented a partial deployment solution where sensor nodes and SDN nodes coexisted. A novel controller-based routing algorithm for prolonging the WSN lifetime was proposed. Their simulation results demonstrated that if the controller

knows the locations and key features of the sensor nodes, the overall network performance can be improved and WSN lifetime can be prolonged regardless of the deployment size.

Wang et al [85] proposed an SDN-based multipath protocol for multihop networks which computes the shortest path by considering the hop count and the node's residual energy. Their experiment results demonstrated that the proposed protocol had lower end-to-end-delay and also prolongs the network lifetime for specific traffic load values as compared to AODV and OLSR. The authors also indicated that the greedy algorithm used to select which nodes to broadcast to, assisted in prolonging the network lifetime.

Due to the application-specific nature of WSNs, it remains a challenge to manipulate its sensor nodes after a full system deployment as these sensors are built for their specific applications. In this regard, a work by Miyazaki et al. [86] proposes an SDWSN approach to curb this challenge by means of reconfigurable sensor nodes composed of an ultra-low power Field Programmable Gate Array (FPGA) and a Microcontroller Unit (MCU) through role assignment. In their proposition, a wireless communication was used to inject roles on certain sensor nodes, thereby allowing the possibility to manipulate or even update the functionality of these sensor nodes depending on the system's application requirement. They reported that their system produced some extent of network flexibility and indicated some potential to serve as a standard structure for WSN deployments depending on different user application requirements.

In [87] OpenFlow based SDN is introduced into WSNs allowing the WSN networking elements to be reconfigurable. In this work programmable nodes are designed and implemented in the data plane. Moreover, the base station is configured as the local controller thereby reducing the processing latency of the central controller. Each node in the data plane is required to comply with the rules distributed in their flow tables in order to communicate with each other in the absence of the current state information of the neighbouring nodes. The experimental results suggested that this architecture achieved a low packet loss rate.

An SDWSN focused Situation Aware Protocol Switching Scheme (SAPS) for real-time support of application-specific requirements was proposed in [88]. 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.

Another SDWSN attempt to improve traffic routing and WSN sensor programmability is that from the work in [89] which introduced an SDN solution for WSN systems, intended for reducing the amount of information exchange between the SDN controller and the adjacent sensor nodes and for enabling these sensor nodes to be programmed as finite state machines. They reported that their system approach increased network elasticity and provided simplified network programmability since it allowed system developers freedom to use programming languages of their choice when implementing the SDN controller. Joint Routing and Resource control (JRRC) protocol for Software Defined Sensor Networks (SDSNs) which rearranges routes and allocate resources for new applications and network services in real-time to maximize the overall throughput was proposed in [90]. Their simulations reveal that the controller is capable of allocating resources reasonably to maximize the network throughput.

Wei et al. [91] proposed a centralised strategy for establishing the path of data upload called Load-balance Centralized Control Routing protocol (LCCR). LCCR is motivated by link quality-based routing protocols and is implemented on Raspberry Pi. The controller has the global view of the network which enables it to measure the link quality which is vital when calculating the routing policies using the load-balance algorithm. Once the routing policies are obtained the controller the runs the LCCR algorithm to distribute the routing policies to the sensor nodes. LCCR ensures that data upload is efficient and reliable by avoiding the selection of unstable links.

Remarkable results were also achieved in [92] where an SDN routing protocol was implemented using OPNET to a multi-hop wireless network. This approach was reported to have increased the network lifetime compared to both Optimized Link State Routing (OLSR) and AODV routing protocols, since it had the potential to provide the shortest path routing and disjoint multipath routing to wireless nodes.

$\mu$ SDN- an SDN based routing architecture for WSNs is proposed in [93]. This architecture extends AODV protocol and Link-Quality Routing Protocol (LQRP) to enable its operation in legacy networks and uses the signalling messages of these protocols to create a communication channel between the controller and sensor nodes. Since the controller is responsible for any modifications made in the network,  $\mu$ SDN takes less time to reprogram the entire WSNs. Their results demonstrated that  $\mu$ SDN achieved the energy consumption levels almost equal to traditional protocol that achieve low power consumption. However, as compared to the original AODV and LQRP,  $\mu$ SDN has an increase in signalling overhead.

[94] investigated the possibility of an SDN based localization algorithm, whereby they proposed a Cramer-Rao Lower Bound (CRLB) based localization node selection algorithm. They formulated an integer linear programming problem based on energy satisfaction by using the controller's network global intelligence. The most reliable nodes for localization were chosen by calculating the contribution of each anchor node in the CRLB metric. Their results indicated that their SDN based localization algorithm achieves considerable improvements. In our view this adds to be a significant 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. Even though SDWSN is regarded as a potential solution for some limitations experienced in WSNs, there is still a serious lack of available strategies or implementations of clear and stateful SDWSN developments, as much as there is little literature regarding this venture.

Some few developments in this field, have at least proposed on some strategies of OSs as well as on the possibility of implementing multiple controller architecture. However, clear technologies in this venture are yet to be done. Of course, ongoing research using different

simulation platforms is currently the hope to achieving the best outcome of this proposed strategy to improve WSN technologies. A work in [95] has proposed a method of testing new protocols presented as TinySDN tool. Their proclamation is that, this tool enables some SDWSN functionality using TinyOS. They also presented that their architecture provides an opportunity of having multiple controllers for SDWSN. Authors in [96] proposed TinySDM - a Software Defined Measurement (SDM) architecture for WSNs. TinySDM has numerous abilities such as providing support for conducting various measurement tasks, allowing easy customization of various measurement task and allowing efficient deployment of new measurement tasks. They implemented TinySDM on the TinyOS platform and evaluated its performance in a testbed comprised of 60 nodes. Their results suggest that TinySDM is flexible, efficient and easily programmable.

In another attempt on applying the SDWSN concept as a solution to WSN challenges, the authors in [97] proposed an approach of TinySDN, a hardware independent TinyOS based SDWSN nodes framework that enables multiple controllers within the wireless sensor network. It incorporates the SDN enabled sensor nodes and a programmable SDN controller. This framework aimed at addressing; leveraging in-band control, higher communication latency, smaller link layer frames and limited energy supply. Their results revealed that TinySDN enables the achievement of flexibility provided by the SDN concept, while it also introduced a memory overhead, but it does not hinder features related to WSN applications.

One of the main functionalities for SDWSN is to accredit simple-programmability as well as to simplify WSN management for any applicable commitment –that is; being able to implement some level of programming on these sensor networks and at the same time being able to easily manipulate them for any network demand. Moreover, SDWSN deployments or systems must be less or considerably expensive since this architecture promotes a system where less sophisticated network components such as more routers and switches are used. Some work in this area have considered sensor network adapt- ability measures depending on the prompts or the demands lifted by the network, whereas other proposals focused on optimizing nodes functionalities by implementing their systems using sensor nodes that can be configured with some level of programmability. Some technological achievements have

been realized such as in [98], where software defined sensor nodes, which can be configured for dynamic operation, can change due to sensor specific task requirements.

WSN Application development and Resource Management (WARM) - an architecture that dissociates sensor nodes at the data plane and reduces management tasks was proposed in [99]. WARM relies on web services and SDN features for flexible resource management, parameterized task scheduling and simplified development of WSN applications. An SDN base model that dynamically selects an appropriate sensor allocation strategy to meet the requirements of an application in HWSNs is proposed in [100]. This proposed model considers the distinct physical characteristics of sensor nodes in HWSNs and the various requirements of the applications request to choose an appropriate allocation strategy in an on-demand manner in order to achieve a certain goal. The Goals may range from optimising energy usage, prolonging network lifetime and minimizing the data exchanged in the network. The preliminary results demonstrated that the proposed model can handle the selection of allocation strategies in an on-demand manner.

Tomovic and Radusinovic [101] proposed a task allocation algorithm that takes advantage of the remaining energy and the location of sensor nodes to allocate sensor nodes to perform application tasks in shared SDWSNs. To optimise energy efficiency in the network, the controller defines paths to be used by cooperative sensor to reach the sink. The results of their experiments illustrated that the proposed task allocation algorithm achieved performance improvements with regards to network lifetime and application acceptance rate.

A novel approach that takes advantage of the SDN-WISE (-Wireless SEnsor) envisioned state information to support QoS provision in WSNs was proposed by [102]. Their approach is based on using a state to report the level of congestion on each node to the controller. A software Defined QoS provisioning strategy for fog computing advanced WSNs was proposed by Huang et al. [103], their proposed strategy dynamically configures QoS policies for various fog services relevant to the data content. The results of their simulations showed that their strategy can improve latency up to 84%.

Han et al [104] proposed an SDN based routing scheme that computes an optimal path based on the QoS requirements of a particular flow. The SDN controller first searches for a suitable

path that meets the QoS requirements of a particular flow, if no path was found the path will be chosen by the proposed routing algorithm based on the flow type (i.e. delay-sensitive, bandwidth-sensitive, and best-effort traffic). Their results demonstrated that their proposed scheme outperformed RIP and OSPF with regards to throughput and delay for audio data transmission.

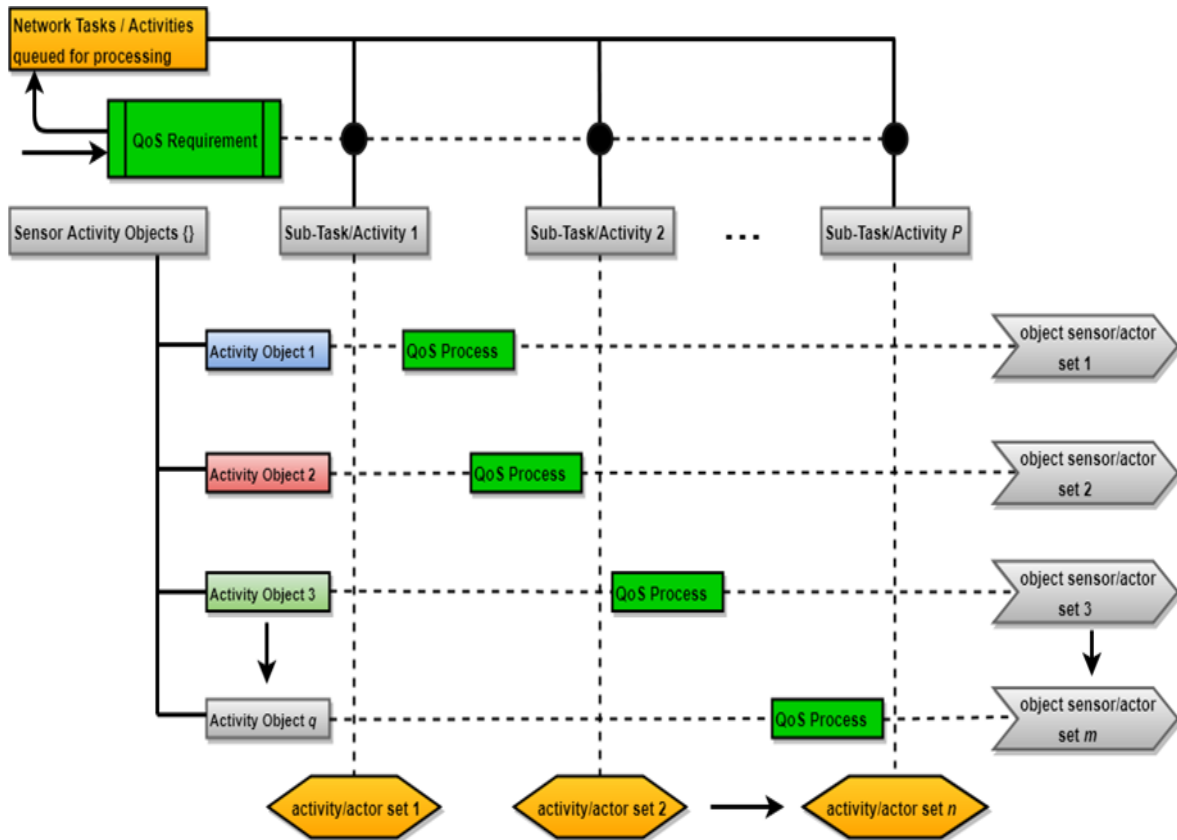
SensorSDN is a novel SDN based architecture for WSNs that can be used for various IoT systems which was proposed by [105]. Firstly, they proposed new control plane services for reinforcing automatic topology discovery, sensor mobility, sensor virtualization and managing network policies. Moreover, they proposed SDN based customizable flow tables on existing Low-Rate Wireless Personal Area Networks (LR-WPAN) technologies to meet the requirements of different sensor packets. Finally, their architecture allows a programmable cross-layer optimization between the MAC layer and the network layer as well as data aggregation to support fine grained flow processing.

Flauzac et al. [106] proposed an application of SDN in WSNs with a structured and hierarchical management. They argued that their approach of applying structured and hierarchical SDN, promises to solve some of the existing problems in WSN management. Their work proposed a cluster-based architecture with multiple base stations as hosts for performing SDN control functions and also as cluster heads at the same time. They further proposed a Software Defined Cluster Sensor Network (SDCSN), a general architecture where the controller can communicate with other SDN domains through some border controllers. An SDWSN Control plane based on Constrained Application Protocol (CoAP) was proposed in [107]. The authors provided extensive specifications of their proposed architecture with regards to communication infrastructure, control plane protocol, topology discovery and maintenance and flow control. The proposed control plane was deployed in Contiki OS and simulations were performed using Cooja. Their simulations measured control traffic overhead and the authors claimed that their proposed control plane showed potential due to the insignificant overhead introduced. However, the authors did not perform extensive tests to validate their claims. Other work such as in [108] explored how northbound interfaces could be improved.

### 2.6.1 Network Resource Management

The core objective of network resource management is to realize network-computing structures that supports manipulation of system structural resources to enhance the performance of the network. The efficient management of network resources contributes to achieving network QoS. Thus, implementing resource management strategies in network computing systems, speaks directly to QoS provisioning. However, efficient network resource management can significantly simplify the computing complexity of implementing network QoS. QoS provisioning in WSNs particularly, is not easily achievable given the characteristics of wireless sensor nodes. The effective application of well-managed network resources advances the opportunity for simplified network management through resource sharing. If network resources are efficiently share by network services and activities, the network becomes flexible and easy to configure.

The role of SDN in WSNs, to introduce network elasticity and computing innovation. This computing paradigm is well suited for implementing flexible resource management strategies. The network programmability nature of SDN provides the potential to abstract specific network applications to meet the ever-growing networking demands. QoS requirements must be catered alongside network resource management since networking aspect such as resource access, activity acknowledgements, flow- or packet-based control benefit from effective QoS measurements and operations. Therefore, it is through efficient network resource management that, increased network performance and flexibility can be achieved. Based on the implemented SDN strategies in this work, we formulate a strategy for efficient resource allocation. Figure 2.6 illustrates the formulated strategy in terms of different network tasks execution, service association and resource allocation with QoS requirements.



**Figure 2.6:** Formulated SDN strategy in terms of different network tasks execution, service association and resource allocation with QoS requirements.

From the figure 2.3, sensor cooperative data aggregation is achieved by associating and allocating different network resources to demanding network activities. Sensor activity objects (SAO) invoke the structured resource allocation and task execution methods of the system, which are implemented by the controller by re-configuring the network to meet different network service demands. This controller activity propagates through all network resources for the purpose of controlling network traffic as well as allocating resources efficiently. The controller then implements diverse levels of QoS requirements to sustain best network performance and to also facilitate efficient computing functionalities.

A traffic and resource-aware model- SDN-TAP [109] was formulated as 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.

Esguerra et al. [110] presented Delay-inducing Congestion Mitigation (DICM) - an SDN based adaptive algorithm that operates at the network level. DICM monitors the aggregate input rate to detect network congestion. In order to reduce the packet losses that are caused by congestion, DICM delays sensor-generated data travelling through a network by making use of the queues that temporarily store packets until the network traffic conditions have improved. Their results demonstrated a trade-off between loss reduction and an increase in the delay experienced by the packets.

Li et al. [111] attempted solving the traffic load minimization (TLM) problem by optimal selection of relay sensor nodes and reduction of redundant packet transmissions. An optimal objective function of TLM problem is formulated as the metric for selection of the relevant relay sensor nodes. A Levenberg–Marquardt algorithm was used to solve the formulated TLM problem. They proposed a Flow Splitting Optimization (FSO) algorithm which aims at finding an optimal path from source nodes to the sink and ensure that the traffic load in the SDWSN is reduced. Their results indicated that the FSO algorithm achieved remarkable Packet Delivery Ratio (PDR) reduced redundant packet transmissions.

### **2.6.2 Network Flexibility**

An adaptive error control implementation framework for SDWSN was proposed in [112]. Their framework exploited the features of SDN and Forward Error Correction (FEC) to improve communication, reliability and flexibility in SDWSN. The framework is said to support adaptability at the transmitter and the receiver and also permits the use of FECs in various sections of the network or link. The authors claimed that the framework is capable



of providing both iterative and non-iterative codes depending on the demand. Compared to non-iterative codes, the use of iterative codes offers more flexibility and adaptability.

SD-WISE- a fully operational Software Defined solution for Wireless Sensor and Actuator Networks (WSANS) and WSNs was presented in [113]. The authors extensively described the major operations of SD-WISE and tested its effectiveness by demonstrating its features in various scenarios. The system is said to have extended SDN approach to WSNs and introduced some novelties as compared to its existing counterparts. Firstly, it introduced a more flexible way to define flows and the possibility to control the duty cycles of the sensor node radios in order to improve energy efficiency. Secondly, it supported NFV and geographic routing was implemented using NFV. Finally, SD-WISE exploited the tight interplay between trusted hardware and software to ensure that sensor nodes comply with the rules imposed by the remote recognised authority.

Considering that link failure/ fault is one of the common issues in WSNs, authors in [114] proposed a lightweight flow management model for reconfiguring flow entries in the flow table when link failure is experienced in the SDWSN forwarding plane. The authors claimed that their proposed model aimed to reduce control messages circulating over the network.

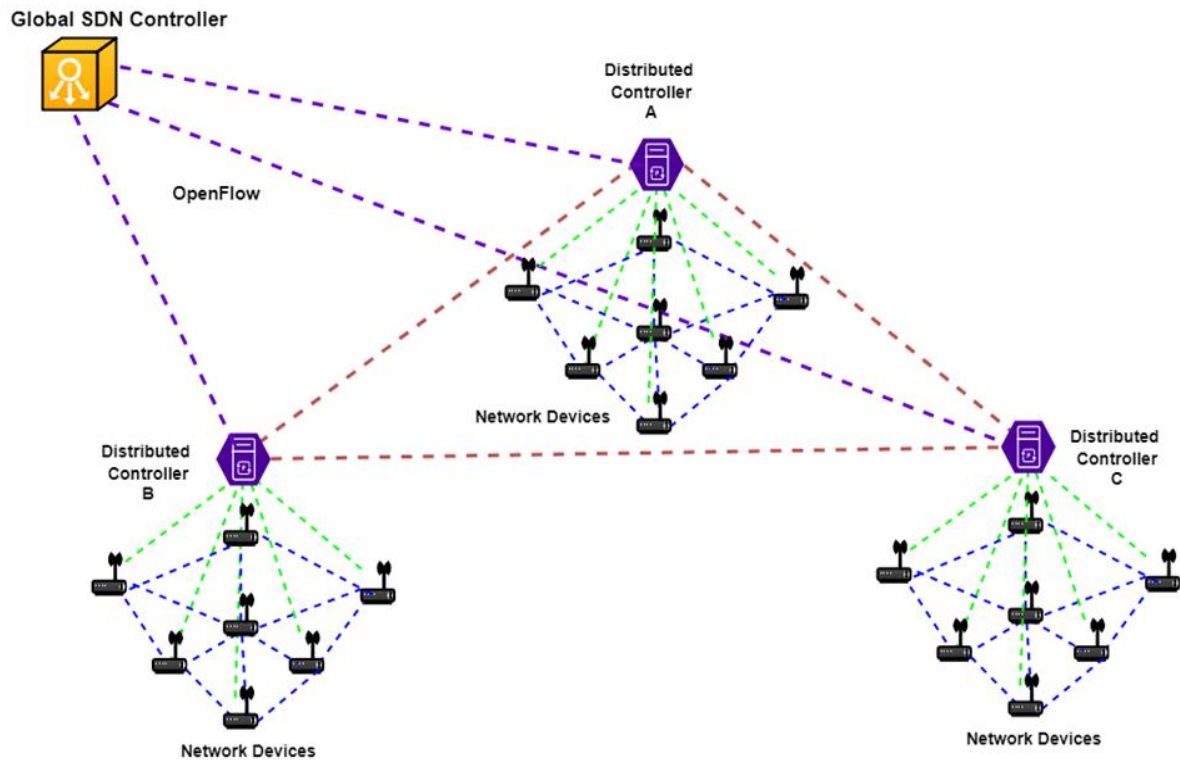
### **2.6.3 SDN for Scalable WSN Networks**

Network scalability is an optimization aspect in computing networks. This aspect is critical as it describes the ability of the network to handle growth in terms of network services and devices. However, to realize a scalable network, the networking architecture must be designed and developed to support this aspect of networking. SDN has the potential to support network adaptability and scalability given its software-oriented strategies. Network scalability can be realized in SDWSN systems, through:

- Increasing the number of SDN controllers in the system. This will increase the controlling functionality of the system in terms of managing a higher number of resources. Distributed controllers will also increase application flexibility as well as promoted support for efficient resource management.

- 
- Introducing automated network services, which will assist in supporting on demand network service applications.
  - Implement abstracted network services to handle specific network functions such as virtualization, security and remote configurations.
  - Increasing the network infrastructure's ability to share computing resources efficiently. This will improve service availability as well as the reliability of the network.
  - Communication interfaces must be optimized to support different network services and a higher number of networking devices.

Figure 2.7, illustrates a network architecture of distributed SDN controllers. In this network setup, the global SDN controller communicates with dedicated distributed controllers to understand various network domain activities. Dedicated domain / distributed SDN controllers manage their respective network infrastructures. Relevant communication interfaces such as east-/westbound interfaces are optimized to support wireless communications between these domain controllers and the underlying network infrastructure. Distributed controller maintains network status of their dedicated network infrastructure and communicate this to the global SDN controller through the OpenFlow protocol. In this case, sensor nodes can also transmit data amongst themselves using wireless communication standards. This infrastructure has the potential to support network innovation as well as improve network scalability.



**Figure 2.7:** A distributed SDN controller network infrastructure.

Provide that, careful planning is adhered to and they are well implemented SDWSN frameworks have a potential to improve application capacities many sensor related systems. The technologies can greatly benefit from rich network application services that can be developed using high level programming languages. SDN has the capacity to support and handle large distributed functionality networks provided they are developed and implemented with intelligent QoS requirements.

#### 2.6.4 Machine Learning Approaches in WSNs and SDWSNs

Researchers in [115] used a Support Vector Machine (SVM) to attain efficient fault detection in WSNs. According to their results, they have reposted that their fault detection method, which is based on classification and data learning, was applied on a cluster head to detect faulty sensors nodes on the cluster. To evaluate their proposed approach, they compare it to two different fault detection techniques commonly applied in WSNs, namely; Detection Accuracy (DA) and False Positive Rate (FPR). However, not disregarding what they have

reported to have achieved in terms of accuracy and fault detection, their results and implementation methodology could be have been much well presented.

In [116] a Parallel Concentric-circle Itinerary-based KNN (PCIKNN) query processing technique was proposed particularly as an effort to derive various KNN query itineraries for different system performance requirements. Based on their approach, they managed to derive analytical models with regard to energy consumption and latency. Their experimental results further indicated that, PCIKNN had better performance in terms of other itinerary oriented KNN algorithms in terms of query accuracy and latency, energy utilization, and scalability. Machine learning methods have also been applied to wireless medical monitoring systems. In [117], a Gaussian Mixture Model (GMM) and the Ant Colony (AC) were used to detecting and isolating anomalous data in a WSN medical monitoring system. Based on their experimental results, they have reported increased levels of detection accuracy of irregularities of wireless measuring medical systems. They reported that, their using their approach, a 100% detection ration can be achieved with false alarm measurements of only 9%. This work has shown smooth coupling of two different machine learning methods to achieve a key objective successfully.

The integration of AI methods to WSNs has evolved their applications and operation in many fields of engineering. In [118], GMM and Bayesian classification were used detection techniques for induction motor bearing faults through the analysis of changes in signals phase space. In this implementation, sensor nodes are used to read electric current and vibration signal. Using GMM, the vibration space's phase signal is constructed to get the motor's healthy and faulty operation conditions. The constructed models are then classified based on their phase space distribution. Their experimental results claim an improvement in terms of faults detection accuracy and that their proposed system can be applied to many related diagnostic systems for different tasks. Vallabh et al. [119] used machine learning techniques to detect falling.

In [120], a learning-based approach was implemented on a heterogeneous sensor network. This implementation proposed a two-phase event detection framework, namely, Watchdog-for vehicle and building traffic detection. According to their report, this framework takes

precaution for sensor energy utilization. This framework used a sentinel sensor clustering method for simplified detection decisions as its first phase. From this task, if the current detection decision has less confidence, the framework then applies a reinforcement sensor clustering to guarantee user requirements. They claim that, their system provided a confident clustering capability for sensor event detection and that it could be applicable to a wide range of deployments.

The rapid growth of cellular network provisioning has also triggered a lot of attention due to the increased number of users that should be served. Optimization methods are being developed and tested so as to provide rich network computation. However, the energy issue is still one defeating constraint in mobile and WSNs for the provision highly responsive systems. Researchers in [121], focused on reduction of utilized Base Stations (BS) as an effort to minimize high energy consumption by them. To achieve this, they used a probabilistic neural network technique to predict user crowded areas so as to effectively reduce these base stations through deactivation or activation during certain periods. Using a number of connected users from twenty BS on a period of eight days, their experimental results report that, using Multilayer Perceptron, SVM and Probabilistic Neural Network (PNN) techniques to provide knowledge of crowd levels on BS.

Even though WSNs application systems have extensive known challenges, they still find relevance in most industrial and security systems. One application system is that in [122], where a combination of acoustic wireless sensors and machine learning methods were applied to detect and evict unwelcomed Unmanned Aerial Vehicles (UAVs). This was driven by the concern for privacy and security as in most cases, these UAVs are not adequately controlled. In their approach, a Software Defined Radio (SDR) receiver with a data training SVM classifier, was integrated with an acoustic sensor, to form a framework that specifies and decodes the telemetry protocol of a UAV. With the UAV's decoded automated communication information, their framework can now, assume the control of an unwelcomed UAV, before even destructing it with their surveillance UAVs. However, they have pointed out that their framework, may not be successful in its task if the unwelcomed UAV, uses an unknown protocol or it is highly encrypted. Even though this was highlighted as a major concern in their framework, their work has opened opportunities for further

improvements or future research directions of their work, in terms of the experienced limitations of their system.

The use of SDN techniques to WSNs has seen a lot of attention in recent work due to the software technicality that SDN brings to them. Significant systematic achievements have been realized through this venture, as application technologies have been revolutionized to provide remarkable functionalities. In [123], the integration of SDN strategies to WSNs has led to resource flexibility and simple control. In this work, an energy-efficient algorithm whose main purpose was to facilitate routing, was proposed for a SDWSN system. In this work, dedicated sensor nodes were dynamically assigned different networking tasks with caution on their residual energy. In addition to their control node selection algorithm, they used a particle swarm optimization technique for clustering functionality, as an effort to enhance sensor energy utilization. Their results reported an increase in network lifetime.

Due to the capacity of SDN techniques to evolve WSNs in terms of traffic routing, flow management, energy conservation methods and the introduction of flexibility, the candidate also advocate that machine learning methods coupled with SDN paradigms, will extend the capacity of WSN application technologies. In [76], a reinforcement learning technique was developed to perform load-balancing and redundancy filtering so as to improve the system's adaptability function and energy efficiency. Based on their results, they have reported that through the use of machine learning techniques and also by applying light-weight flow table execution in their approach, network control and intelligence can be achieved. They have reported their approach as beneficial for SDWSN monitoring systems. Therefore, these results, suggest a remarkable improvement in SDWSN systems when coupled with machine learning schemes.

Researchers in [124] formulated a resource and revenue sharing for Cloud-assisted Software-Defined Wireless Sensor Networking (CSDWSN) coalition development. This approach was encouraged by the increased mobile and wireless user demands of powerful computational resources in online services providers. Their proposed method formulates a coalition game scheme amongst collaborating CSDWSN service providers. In this work, a Cooperative Energy-aware Opportunistic Game (CEOG) model is deduced to facilitate

resources and income sharing for collaborated development amongst these CSDWSN providers. To effectively achieve a dynamic pricing strategy for these CSDWSN's, so as to maximize their profits, a reinforcement Q-learning technique is adopted. In addition to this, a Markov chain-based method for CSDWSN federation forming is applied. Their experimental results suggest that, their proposed scheme, provides an opportunity for shared capacity by these CSDWSN providers in terms of revenue.

## 2.7 CHAPTER SUMMARY

In this chapter, a survey of related work has been conducted. This survey has provided a clearer understanding of strategies implemented in this area and has assisted in developing potential strategies to enhance SDWNS application system. The survey has also motivated the candidate to implement abstracted top-layer services with QoS requirements to improve network resource sharing. The surveyed related work extends around resource management, QoS implementation, information presentation, traffic control and data aggregation, SDWSN for flexible IoT networks and machine learning methods to WSNs and SDWSN systems. In each section and sub-section of this chapter, a survey of related work is conducted with the purposed to form a directive towards the development of efficient SDWSN network computing strategies.

- Section 2.1 draw a focus on the objectives of this study. These objectives form a direct response the formulated research question.
- Section 2.2 looks into different routing protocols applied in WSN. Understanding these routing protocol contributes to their efficient implementations. It further looked into communication protocols for efficient controller to devices communication. To understand their various capabilities, design constraints of these routing protocols were studied with the objective to formulated efficient SDN strategies for enhanced WSN routing protocols.
- Section 2.3 looked into application and system challenges of WSN technologies, in particular for monitoring applications. SDN strategies were therefore formulated to deal with the WSN application system challenges discussed.

- 
- In section 2.4, a comprehensive study of the SDN paradigm was conducted, with the purpose of drawing its impact in wireless computing systems. This paradigm is discussed with critical analysis of what computing opportunities can SDN bring to WSN monitoring application systems. Developmental strategies are then projected in focus of developing SDWSN monitoring application systems that are efficient in terms of routing, query processing, resource management, data handling and aggregation as well as information presentation. A study of SDN controllers was also conducted, with the purpose of providing a description of these controllers and what networking services they support.
  - Section 2.5 discussed opportunities for flexible resource management for IoT network. This study was conducted on the basis that, WSN systems also form the architecture for wireless IoT systems especially those that can be deployed for monitoring applications. Systematic opportunities or benefits that can be achieved through SDN strategies are then discussed and outlined. Computing aspects such as efficient data retrieval using fast cache memory, IoT objects communication capabilities and possibilities for improving wireless IoT systems security, were also outlined and possible practical strategies were formulated.
  - Section 2.6 discussed SDN opportunities for WSN application systems. These discussions were drawn from surveyed related work for the proposed application area. Application improvement is studied in terms of programmable strategies for efficient resource management, network flexibility as an opportunity for automated network services and network scalability as a critical aspect for the capability of a computing network to handle growth and associated network applications. Also, in this section, a study of applying machine-learning methods to WSN and SDWSN application systems was conducted. Related work on these technological areas was carried out with the purpose of formulating and developing efficient methods to improve SDWSN monitoring application systems.



# **CHAPTER 3                      SDWSN: A FLEXIBLE WSN NETWORK**

## **3.1 CHAPTER OBJECTIVES**

This work implements an SDWSN approach coupled with Discrete Event Simulation (DES) and a highly extensible and scalable SDN controller – ODL, to implement a software-oriented network environment to increase network service adaptability and simplify network management.

## **3.2 SIMULATION ENVIRONMENT**

This section provides critical information regarding the NS-3 simulator's model library and different API support capabilities. However, more details will be provided based on the space of this work, with the basis to form a research direction of this work in terms of its contribution. A direct emphasis is put on applications and the NS-3's packet design model.

### **3.2.1 NS-3 Network Modules and Model Library**

NS-3 allows discrete event simulation of several network models and routing protocols that can be modelled either using Python or C++ as its simulation modelling language. The NS-3 framework supports a lot of modules and libraries for different devices and networking models. This makes it a rich simulator for many network simulations and model building. Due to the scope of this work, being SDWSN, NS-3 environmental capabilities for wireless

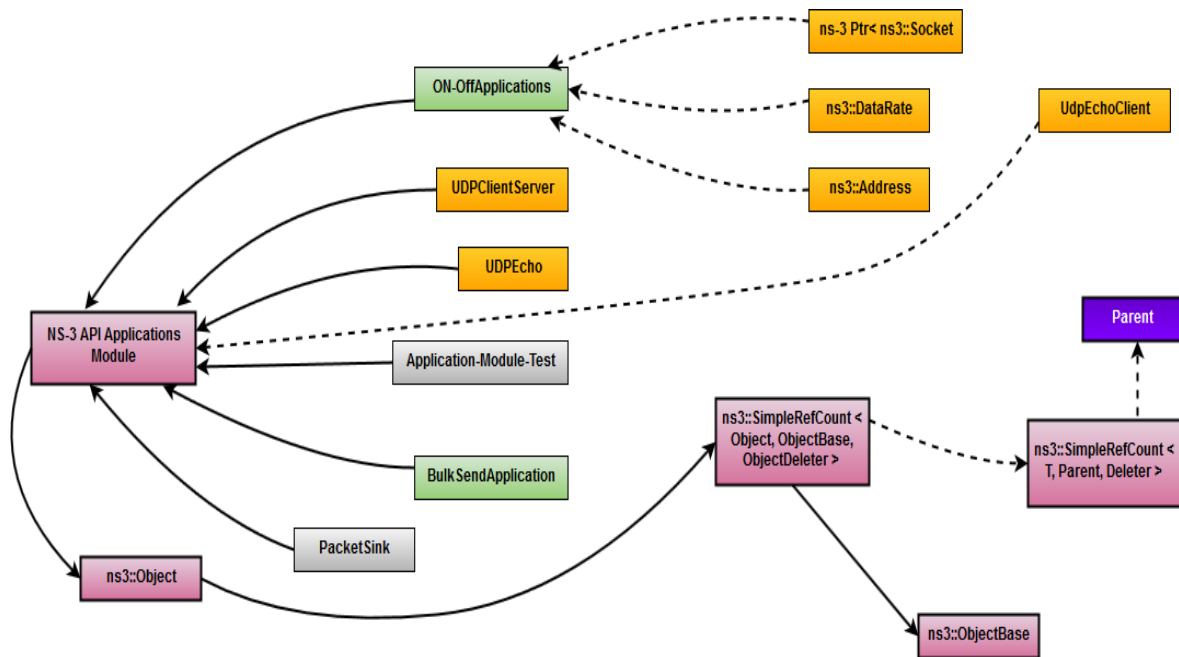
communication systems will be provided extensively but not disregarding its capability for modelling other networking infrastructures or scenarios. Commonly known or used modules, which are also well supported in NS-3 include but not limited to those appearing in Table 3.1.

**Table 3.1:** Critical modules of the NS-3 simulator with their descriptions.

Module	Description
<b>Applications Modules:</b> <ul style="list-style-type: none"> <li>– Packet Sink</li> <li>– On-Of Applications</li> <li>– UDP Client Server</li> <li>– UDP Echo</li> <li>– Applications Module Test</li> <li>– Topology Generator</li> <li>– CSMA Network Device</li> <li>– Configuration Store/Load</li> </ul>	These are some of the main application modules of the NS-3 environment. They serve the purpose of modelling different networking developments. These applications modules provide, data transmission, network state control, network bridging functionalities, channel computations, the storage and loading of various simulation attributes. Provides functionalities for generating network topologies using the environment’s modelling procedures. Also provides nodes functionalities in sensor networking for avoiding packets collisions by waiting for the ready state of transmission channels.
<b>Core Processing Modules</b>	These modules provide the core functionalities of the NS-3 in terms of data collection and manipulation, traffic handling and control, scripting, interfacing, etc.
<b>AODV Routing</b>	Provides routing support for different ad-hoc network models.
<b>6LoWPAN</b>	Provide compression techniques for low energy computation for devices using IPv6.
<b>Network Module</b>	A very critical module of the NS-3. Provides networking functionalities to support different networking aspects and devices.

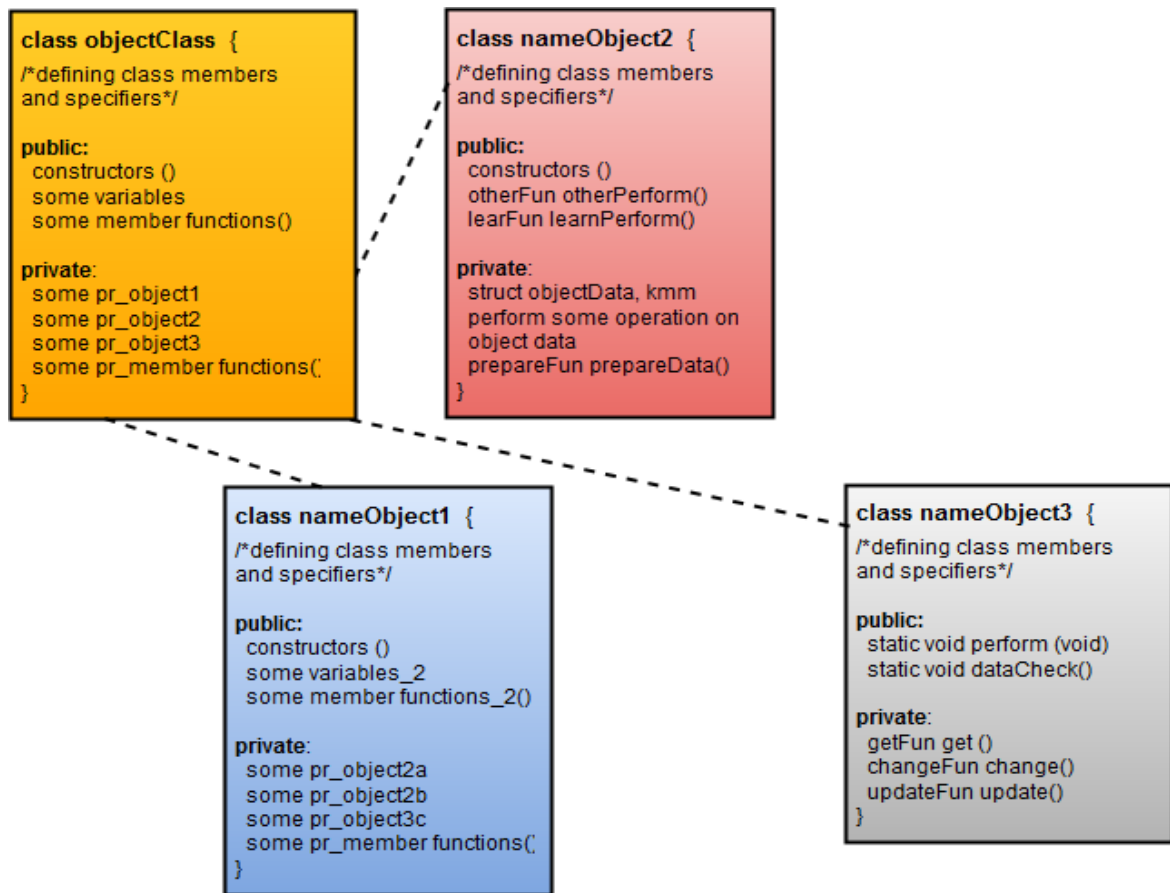
<b>Internet Module</b>	Facilitates the networking model of the NS-3. Provides functionalities for addressing, client-server requests and responses.
<b>Flow Monitor</b>	Another critical aspect of the environment, which provides services for collecting and storing simulation data.
<b>Statistics</b>	Provides network features to simplify collection of data from various simulation tasks.
<b>Mobility</b>	Provides functionalities for tracking positions of system objects and support for nodes placement classes
<b>OLSR Routing</b>	Another critical module of the environment. Provides functionalities for device communications and to build spectral analysis
<b>Traffic Control</b>	This module provides traffic control operations for various IP stacks
<b>Utils Directory</b>	Provides services for facilitating coding tasks, testing and objects parsing functionalities

As indicated before, NS-3 is a discrete event simulator (DES) with powerful capabilities to model different network infrastructures. It supports different computing extensions for net-centric communications, programmability, device modelling, etc. With optimal device and communication interface definitions, settings and parameters, the NS-3 simulator is well suited for modelling WSN networks, given its powerful network model and libraries. Hence, the SDN networking model takes advantage on the model libraries forming the basis functionality of NS-3. SDN's powerful communication standard- OpenFlow, is well supported by the NS-3 model library. The NS-3 support high-level programming languages to manipulate network devices and services, which maps well with the SDN paradigm for improved network application systems. Therefore, with SDN and OpenFlow capabilities extended to the NS-3 simulation environment, efficient SDWSN models can be built, with high-level performances. NS-3 rich support for API applications module. This allows it to model different network models with well-organized network architectures. A collaborative API application module overview is give in Figure 3.1.



**Figure 3.1:** A collaboration diagram for certain applications with the object base and the parent instance.

NS-3 also provides functionalities to support high-level programming class operations. This allows NS-3 to support SDN programmability strategies with its rich class application modules. It also facilitates programmable abstraction operations from most high-level programming languages such as C++, Java and Python. NS-3 can be easily integrated with modern SDN controllers for different capabilities to manipulate the modelled network. In this work, NS-3 has been integrated with the ODL controller, which provides the global view and intelligence of the network. NS-3 allows rich functionalities for manipulating data packets using classes. This capability of NS-3 extends to its support for the provision of abstracted services. Figure 3.2 provides a simple description of an object class implementation in NS-3 and its related classes.



**Figure 3.2:** A description of an object class and its related class implementation in NS-3.

Different member functions of related classes access the main object class through some function specifiers. This allows rich object data manipulation from different member functions defined on the global scope of the implementation. Various methods are also declared a part of specific classes to deal with objects data at the class level. This implementation also allows inheritance function operations on real object data to meet different requirements.

### 3.2.2 ODL: Enabling the ODL Controller

ODL is an open source SDN oriented project that provides capabilities to manage and visualize network components [125]. To effectively centrally manage our simulated network, an ODL controller was used. This is an SDN controller that offers a list of functionalities and capabilities since its development focuses primarily on SDN environments and network optimization. This controller was coupled with the simulation

tool to render a visual representation of the system. ODL provides some level of programmability onto the network to change how computing networks function, process data and manage underlying network hardware and software components.

The ODL controller's powerful capability is subject to the rich resource structure of the ODL framework. To understand some critical capabilities of the ODL framework, Table 3.2 is provided below, with core network applications and platform services that forms its architecture. The ODL supports extensive southbound interfaces and plugins. It also supports commonly applied APIs on its northbound interface. These interfaces provide rich communication functionalities between the controller and both the top application and data layers. The framework has also powerful platform service which allows it to perform various data manipulation functions. Hence, the ODL framework provides stable data management functionalities, through these platform services.

**Table 3.2:** Core platform services and applications for the ODL framework.

Network Services and Applications	Platform Services	Northbound APIs	Southbound Interfaces & Plugins
<ul style="list-style-type: none"> <li>– Application layer traffic optimization</li> <li>– Bit Index Explicit Replication (BIER)</li> <li>– Controller Orchestration Engine</li> <li>– Group-Based Policy</li> <li>– Genius Framework</li> <li>– Honeycomb Virtual Bridge Domain</li> <li>– Lisp Service</li> <li>– Neutron Service</li> <li>– Network Modelling Engine</li> <li>– OpenFlow L2Switch</li> <li>– Packet Cable Service</li> <li>– Service Function Chaining (SFC)</li> </ul>	<ul style="list-style-type: none"> <li>– Authentication, Authorization and Accounting</li> <li>– Data Export Import</li> <li>– Infrastructure Utilities</li> <li>– JSON-RPC Extension</li> <li>– Time Series Data Repository</li> </ul>	<ul style="list-style-type: none"> <li>– REST</li> <li>– RESTCONF</li> <li>– NETCONF</li> </ul>	<ul style="list-style-type: none"> <li>– OpenFlow</li> <li>– OFConfig</li> <li>– P4</li> <li>– BGP</li> <li>– PCEP</li> <li>– LISP</li> <li>– NETCONF</li> <li>– OVSDB</li> <li>– SNMP</li> <li>– SXP</li> <li>– USC</li> <li>– PCMM</li> <li>– COPS</li> </ul>

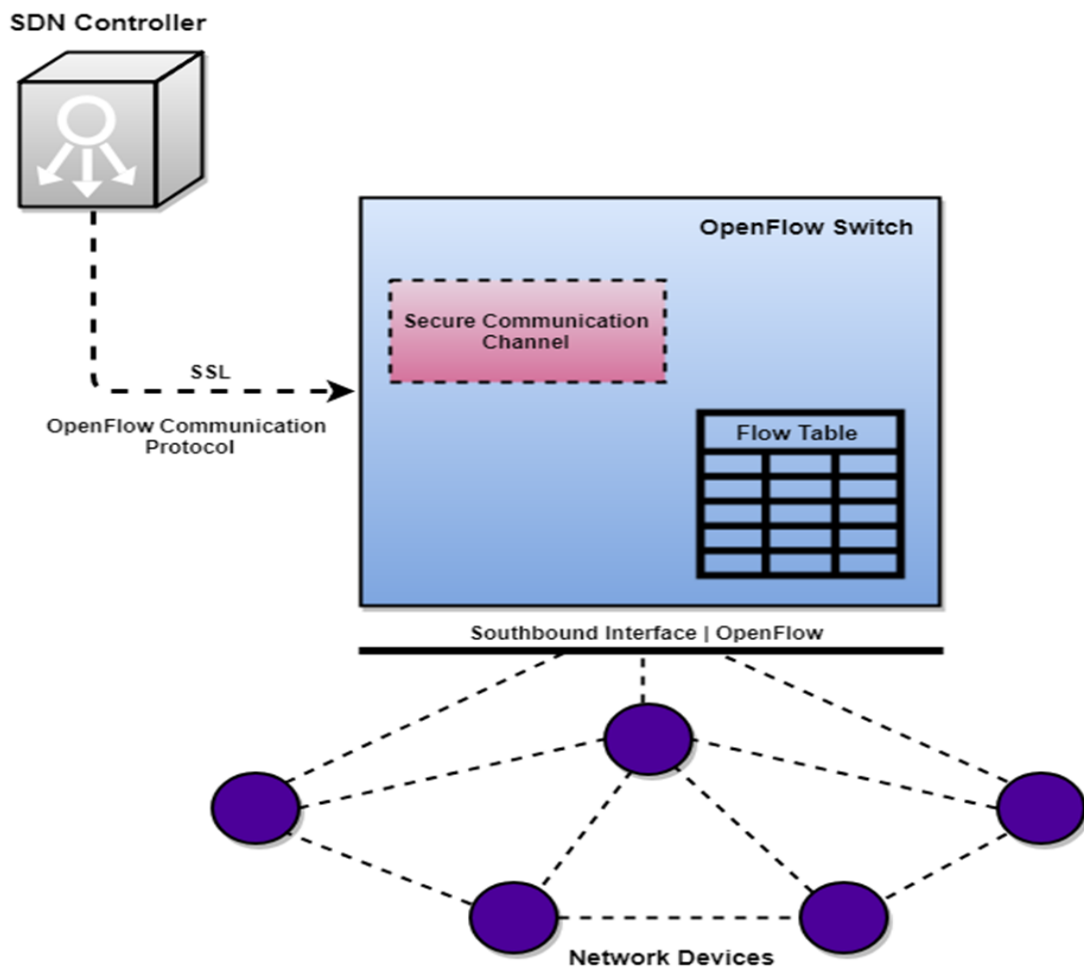
<ul style="list-style-type: none"> <li>– Unified Secured Channel Manager</li> <li>– Virtual Tenant Network</li> </ul>			
---	--	--	--

The ODL framework fully supports applications and services abstraction. ODL's support for high-level programming languages allows opportunities to develop high-level networking applications. ODL also supports container-based services and applications. This also provides opportunities for developing scalable implementing robust network applications using the SDN ODL controller. With ODL's group-based policy and application layer traffic optimization services, efficient QoS provisioning can be realized. This also contributes to the development of highly scalable network application systems. ODL has rich functionalities for implementing secured networking applications through its unified secured channel manager. Another powerful ODL application is the, controller orchestration engine. This application allows ODL to be implemented as a highly scalable and extensible SDN infrastructure. Controller orchestration also allows the ODL controller to be implemented with distributed functionalities. This increases system robustness and rich network application services. The ODL framework's network applications and platform services enables it to implement powerful networking aspects such as NFV and network automation. With these capabilities, network resource management can be performed with high adaptability.

### 3.2.3 OpenFlow Communication Standard

The OpenFlow communication protocols provides rich support functionalities for SDN based networks [46] by facilitating communication between the SDN controller and the associated data plane[126], [127] and simulation environments [128]. The OpenFlow communication standard supports a wide range of network services such as; service automation, abstracted configuration services, programmable flow injections and service manipulations. Network services can greatly benefit from SDN based OpenFlow applications especially in large network infrastructures. On demand network applications require the best communication technologies to support abrupt and concurrent network processes. These services can benefit OpenFlow based SDN networking infrastructure through efficient resource management and flexible service manipulations.

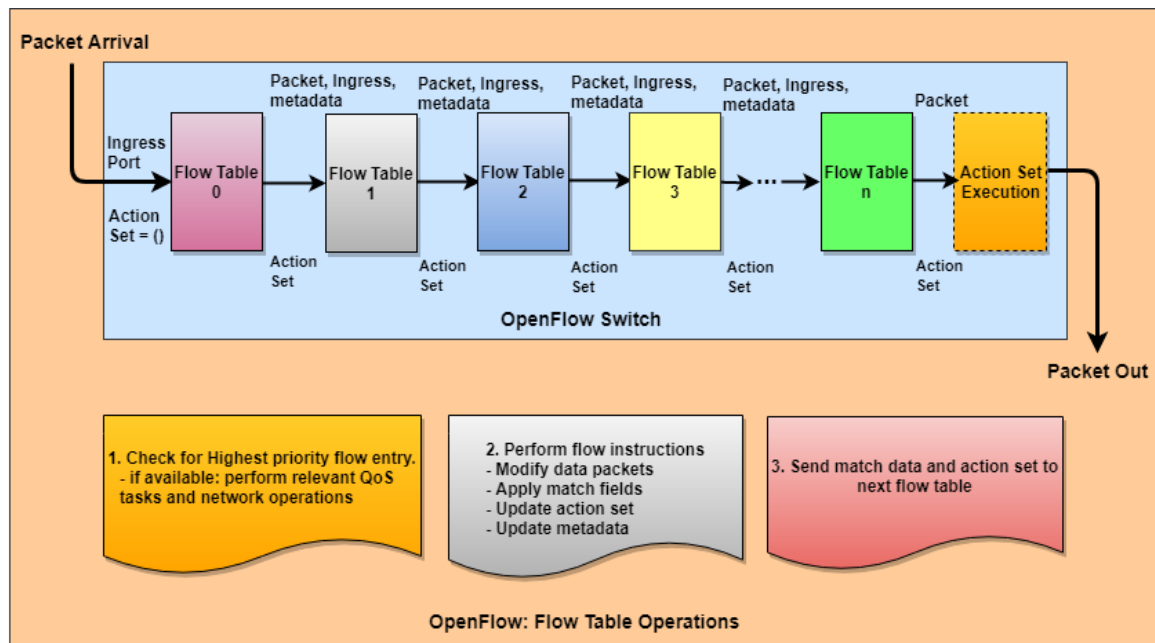
OpenFlow supports network service and resource flexibility and as a result can be used to map a larger number of packet flow entries to various flow queues depending on their output ports [129] [130], which relate to different applications. OpenFlow can be used to develop and implement new routing techniques on SDN based networks [131] [132] and simplify management of WSN systems [133] [134]. OpenFlow is well documented and maintained with rich controller communication support [135] that justifies its support for SDN based deployments. With competent implementation, OpenFlow also has the potential to perform critical network functions such as efficient load balancing and traffic routing [136]. Figure 3.3, presents an OpenFlow based SDN network setup with communication support between the SDN controller and the underlying data plane.



**Figure 3.3:** OpenFlow based SDN infrastructure with data plane communication support.



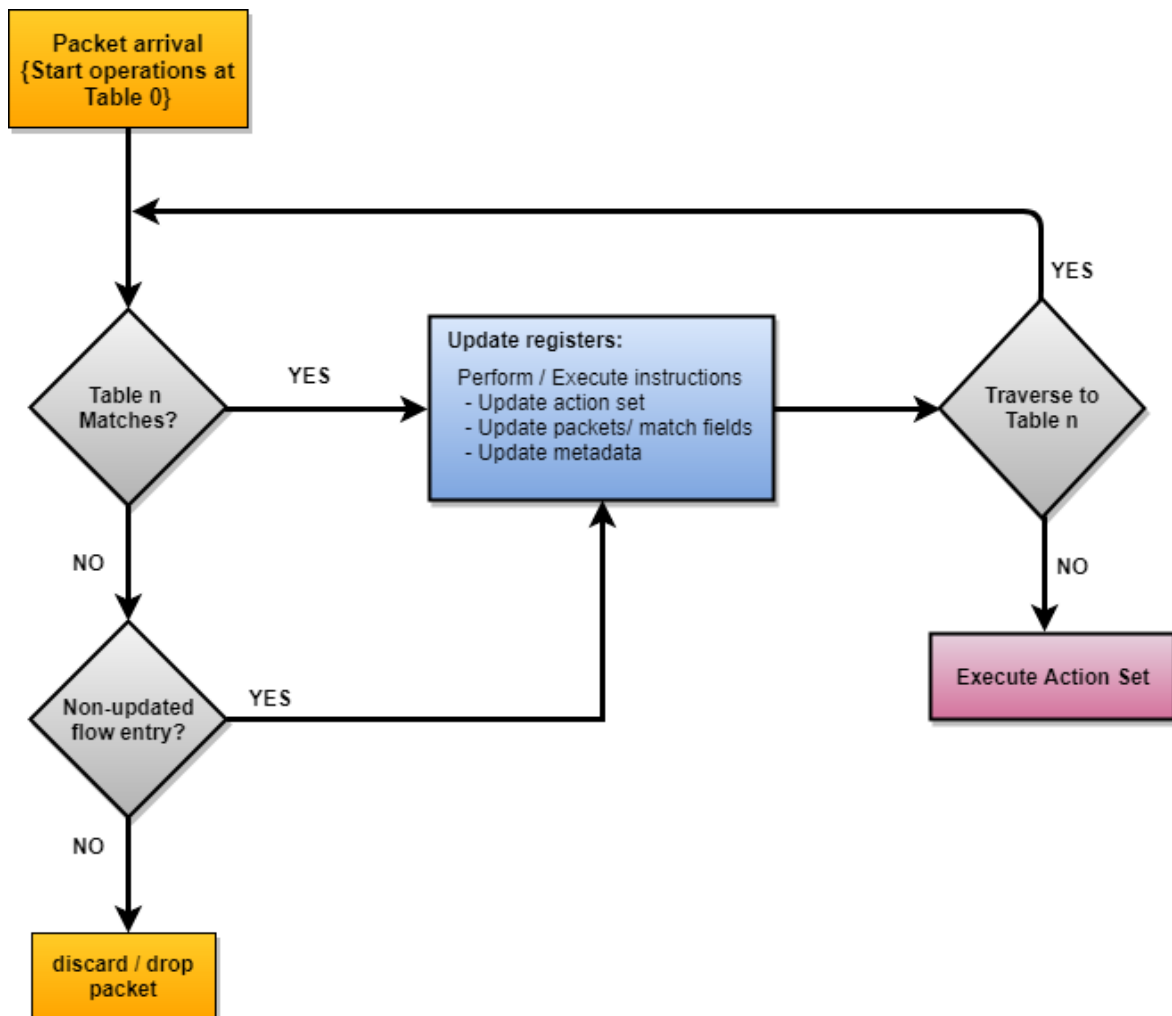
OpenFlow promotes rich controller to data plane communication functionalities through flow manipulation. Flow table entries are update from the SDN controller to the OpenFlow for directing relevant flow updates to underlying network infrastructure. With this cooperative capability between the SDN controller and the OpenFlow communication protocol, remote network services can be realized easily. SDN programmable strategies complements OpenFlow's communication capability to implement effective flow based rules to targeted network environments or devices. Increased network visibility and resource sharing in SDWSN systems can be easily achieved through this cooperation. Figure 3.4, presents a systematic view of OpenFlow processes in terms of flow operations.



**Figure 3.4:** OpenFlow's flow entry operations on different flow tables.

OpenFlow's capacity to manage flow entries, promotes the SDN paradigm's mission for achieving reliable network operations through programmability. OpenFlow manage operate on flow tables by implementing SDN controller functionalities to dedicated network environment. Its purpose in a SDN infrastructure is to check flow compliance and correctness to achieve the implemented controller actions. Thus, the OpenFlow implements these action sets in terms of controller instructions and associate them to checked and matched flow table

entries. Figure 3.5 below, illustrates a procedural flow of an OpenFlow switch for manipulating flow entries.

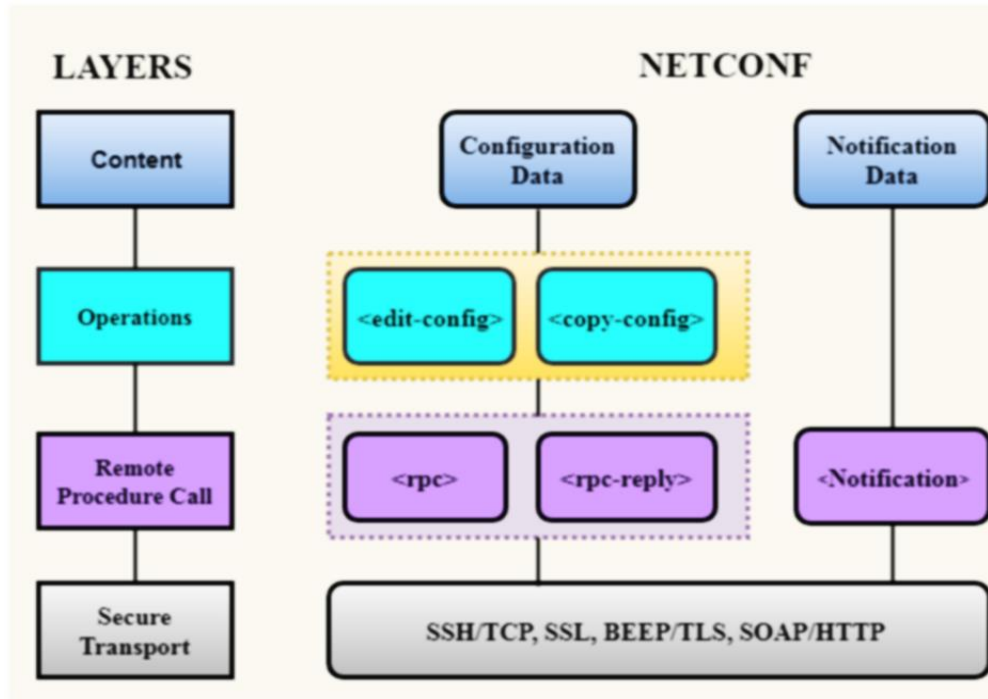


**Figure 3.5:** An OpenFlow's process flow operation on flow entries.

When packets arrive in a specific port or at an interface, OpenFlow operations are started to apply control actions. Flow entry matching processes are performed on tables so as to apply relevant table updates. To update flow entries, memory register must be manipulated accordingly to allow the implementation of current system applied changes. This process is critical as it directly influence general network operations. The OpenFlow protocol perform these operations on relevant flow tables, as different activities for different services are stored on different table. The protocol the traverse these tables to apply network required action sets. If there are no new flow entry updates, the protocol updates the memory registers

accordingly. Otherwise, if the same entries are updated repetitively and transmitted to the same table, the protocol uses registers to match these same entries and drop packets flows if their state are not necessary.

### 3.2.4 BGP AND NETCONF

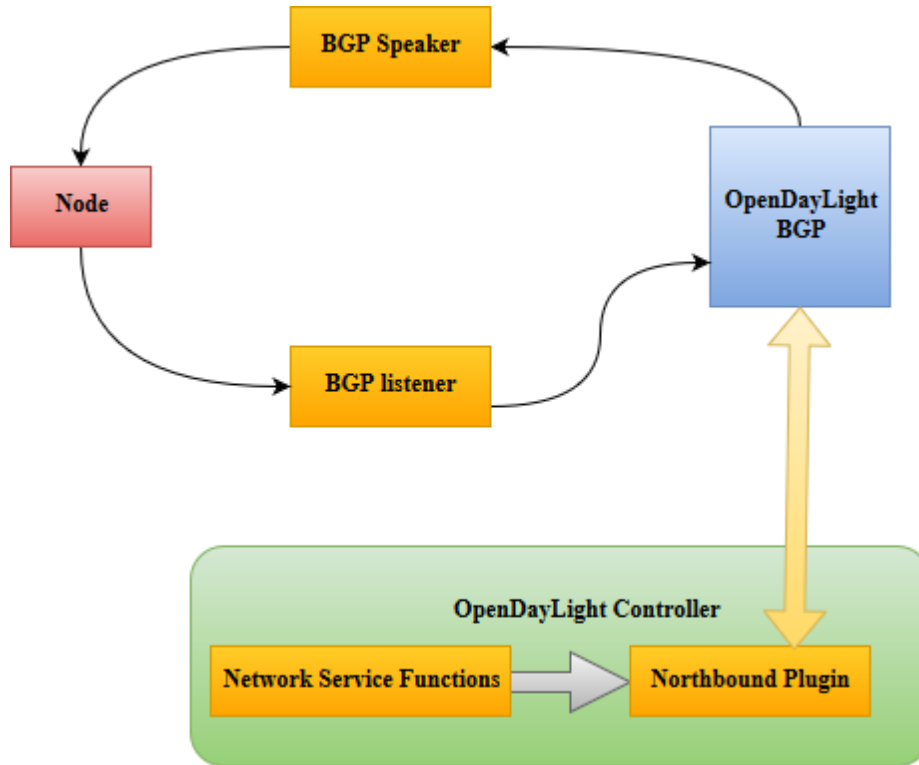


**Figure 3.6:** NETCONF Protocol Layers

NETCONF is partitioned into four layers: content layer, operations layer, Remote Procedure Call (RPC) Layer and Secure Transport layer.

1. Content Layer provides the configuration and notification data to the operations layer.
2. Operations Layer defines a set of primitives invoked as RPC methods with XML-encoded parameters to manage configurations. The content in content layer and operations layer is modelled in the form that is human readable using yang models.
3. RPC Layer provides mechanisms for encoding transport protocol-independent RPCs and notifications.

4. Secure Transport layer provides a secure communication path between client (NETCONF Agent) and server (NETCONF Manager). This transport is usually through SSH protocol.



**Figure 3.7:** BGP data flow diagram

Northbound plugin serves as a collector, it is connected to the network service functions and gathers network state information in order to compute the global network view. It achieves this by querying the network nodes through MD-SAL. The OpenDayLight module sets policies and send BGP messages across the network. The BGP speaker sends BGP messages and also advertises routes to the network nodes. The BGP listener sends BGP messages and also advertises flow configuration requests to the controller.

### 3.3 EXPERIMENTAL SETUP

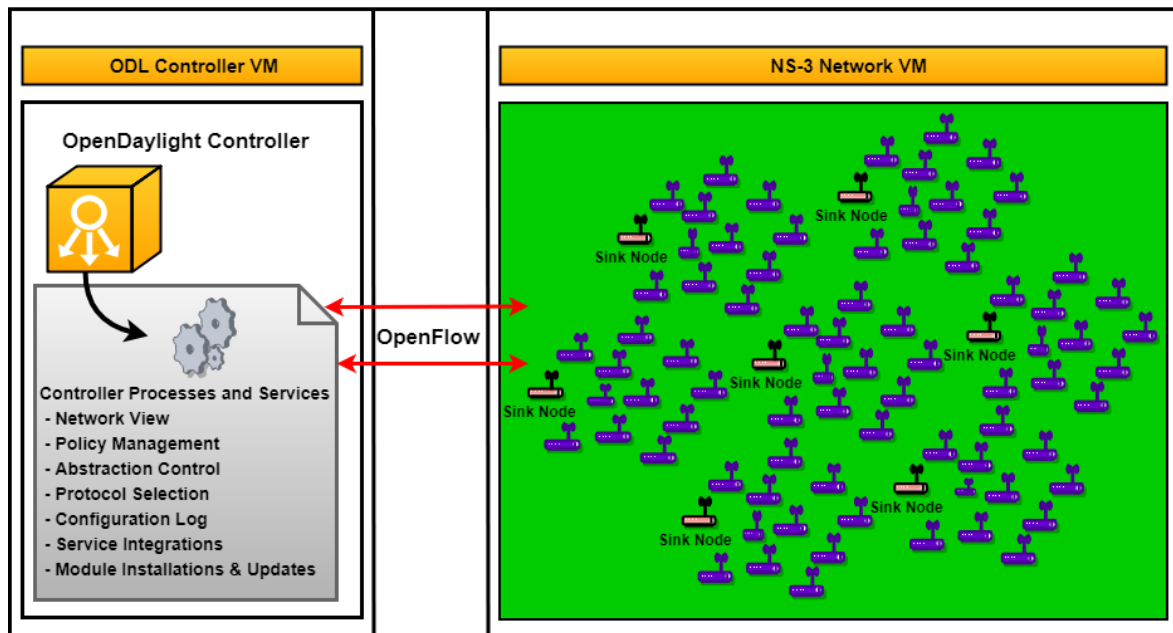
NS3 was used for modelling the sensor network with parameter settings and options as illustrated in Table 3.3. Based on the proposed approach, OpenFlow (version 1.5.1) was used

as the key communication protocol between the ODL controller and the modelled network. However, to evaluate its networking performance, the same setup used both NETCONF and BGP protocols. Virtual machine for both NS3 and the ODL controller were running on the Linux environment, with Ubuntu 16.04 OS. BGP ensures that data is transmitted with minimum latency over the Internet between autonomous systems and provides alternative paths in case of route failure [137]. NETCONF provides control and network management functionalities in SDN networks [138]. These protocols entail operational SDN strategies such as those of OpenFlow. Hence, they were applied in this work to compare their performance against OpenFlow.

**Table 3.3:** Environment and System Parameters

<b>Parameter</b>	<b>Value by Type or Version</b>
Node Mac/Phy Protocol	IEEE 802.15.4
Communication Protocol	OpenFlow 1.5.1
Network Size	400*900
Node Count	100 Sensor Nodes
Packet Size	128 bytes
Transmission Rate	250 kb/s
Simulation Time	60 (s)

A pictorial of the system setup is give in Figure 3.8 below. Both the ODL controller and the NS3 modelled network run on separate Virtual Machine (VM) instances. This allows the actual separation of the control plane and the simulated environment. In the complete system setup, the ODL controller is accessed as an external controller through the OpenFlow protocol. Simple network service such as; tracing the network, viewing and changing certain simulation parameters can be done from the controller. The controller also allows the view of the actual network nodes with connection statuses of all the network nodes.



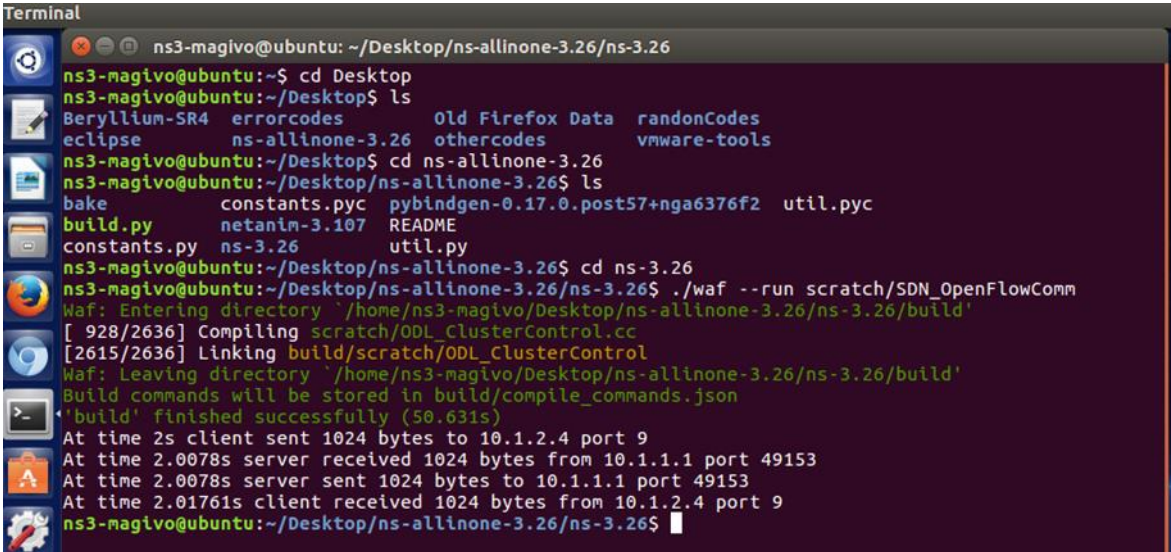
**Figure 3.8:** Presentation of the Network Model.

The sink nodes are also illustrated on the NS3 VM instance in blue colour. These sink nodes communicate with the simulated sensor clusters and cooperatively transmit aggregated traffic to the external controller via the OpenFlow channel. The ODL SDN controller is responsible for different network critical services such as; network reports, building network topology models, policy and feature installations. Furthermore, the controller can be connected to application layer services through the northbound communication channel to associate or integrate application layer facilities or processes with the underlying network. Examples of application services can be; web-based applications, authorised remote access to the system via the internet and other simple top layer APIs. In our approach, flow state rules such as QoS roles performed on communication protocols, simple cluster-based functions executed on sink nodes and controller services such as routing and load balancing commands were implemented. To further explain our implementation.

To simplify and improve network management, the implemented approach uses the ODL's Model-driven Service Abstraction Layer (MD-SAL) and the Yet-Another Network Generation (YANG) models to apply system specific tasks such as; installing new network device features, discovery procedures, data and response handling services. To achieve these SDN controller services and functionalities this implantation takes leverage of the ODL's OFConfig (OpenFlow Configuration) rules to abstract OpenFlow network resources to influence their management as well as the YANG model's NETCONF services to manipulate and enable communication between controller services and network devices. Throughput refers to the average communication channel's packet transmission rate with success delivery from the source to the destination device as expressed on the sensor nodes' packet radio measure. In this instance, the average flow transmission rate is measured on the three applied SDN communication protocols (*OpenFlow*, *BGP* and *NETCONF*) for different datagrams. The throughput measure is expressed from the communication protocol's capacity to successfully transmit nodes' sent packets over time.

### 3.4 EXPERIMENTAL RESULTS

This section gives details with analysis of achieved results based on the experiments conducted with system parameters as tabulated in Table 3.3 above. The main focus in terms of results analysis is directed to how SDN strategies improve WSN system applications in particular for monitoring applications.



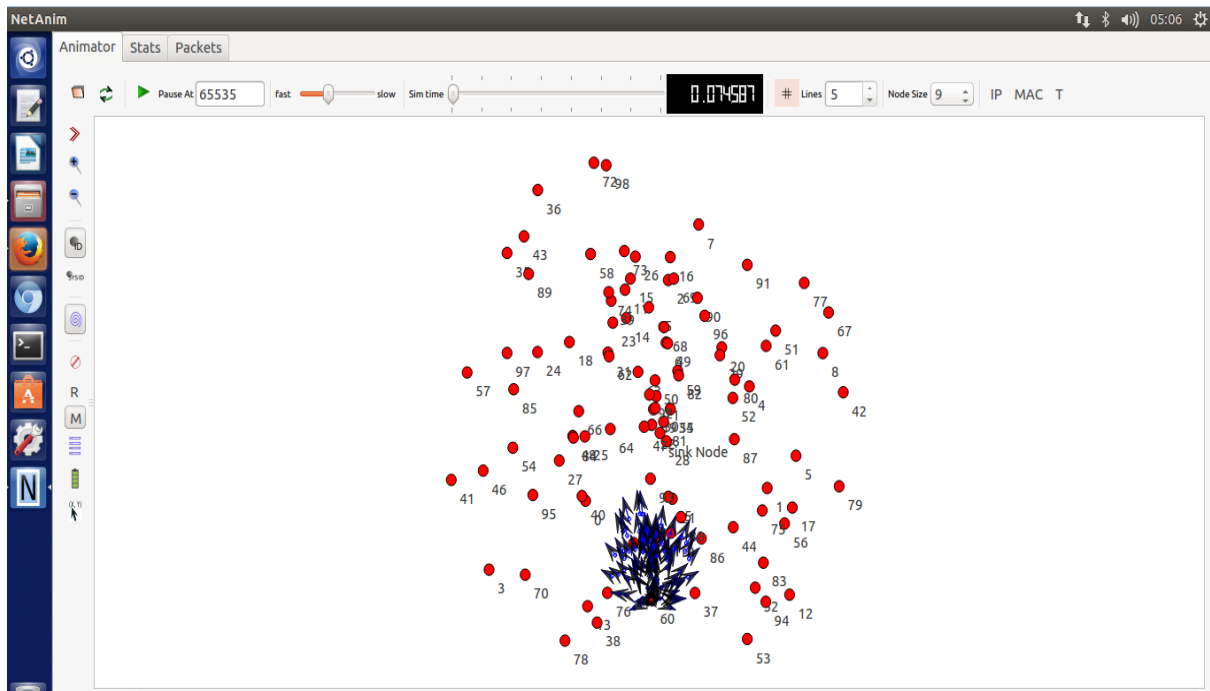
```

Terminal
ns3-magivo@ubuntu: ~/Desktop/ns-allinone-3.26/ns-3.26
ns3-magivo@ubuntu:~$ cd Desktop
ns3-magivo@ubuntu:~/Desktop$ ls
Beryllium-SR4  errorcodes          Old Firefox Data  randonCodes
eclipse       ns-allinone-3.26  othercodes        vmware-tools
ns3-magivo@ubuntu:~/Desktop$ cd ns-allinone-3.26
ns3-magivo@ubuntu:~/Desktop/ns-allinone-3.26$ ls
bake          constants.pyc  pybindgen-0.17.0.post57+nga6376f2  util.pyc
build.py      netanim-3.107  README
constants.py  ns-3.26        util.py
ns3-magivo@ubuntu:~/Desktop/ns-allinone-3.26$ cd ns-3.26
ns3-magivo@ubuntu:~/Desktop/ns-allinone-3.26/ns-3.26$ ./waf --run scratch/SDN_OpenFlowComm
Waf: Entering directory `/home/ns3-magivo/Desktop/ns-allinone-3.26/ns-3.26/build'
[ 928/2636] Compiling scratch/ODL_ClusterControl.cc
[2615/2636] Linking build/scratch/ODL_ClusterControl
Waf: Leaving directory `/home/ns3-magivo/Desktop/ns-allinone-3.26/ns-3.26/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (50.631s)
At time 2s client sent 1024 bytes to 10.1.2.4 port 9
At time 2.0078s server received 1024 bytes from 10.1.1.1 port 49153
At time 2.0078s server sent 1024 bytes to 10.1.1.1 port 49153
At time 2.01761s client received 1024 bytes from 10.1.2.4 port 9
ns3-magivo@ubuntu:~/Desktop/ns-allinone-3.26/ns-3.26$

```

Figure 3.9: ODL OpenFlow communication.

Figures 3.10, 3.11 and 3.12 represent a simulated model of 100 sensor nodes. The control is provided by the ODL controller through OpenFlow. The simulation shows packet aggregation from different sensor nodes. Flow transmission is forwarded from a certain node to its closest or reachable neighbours. This process occurs until dedicated packets reach their respective destinations.

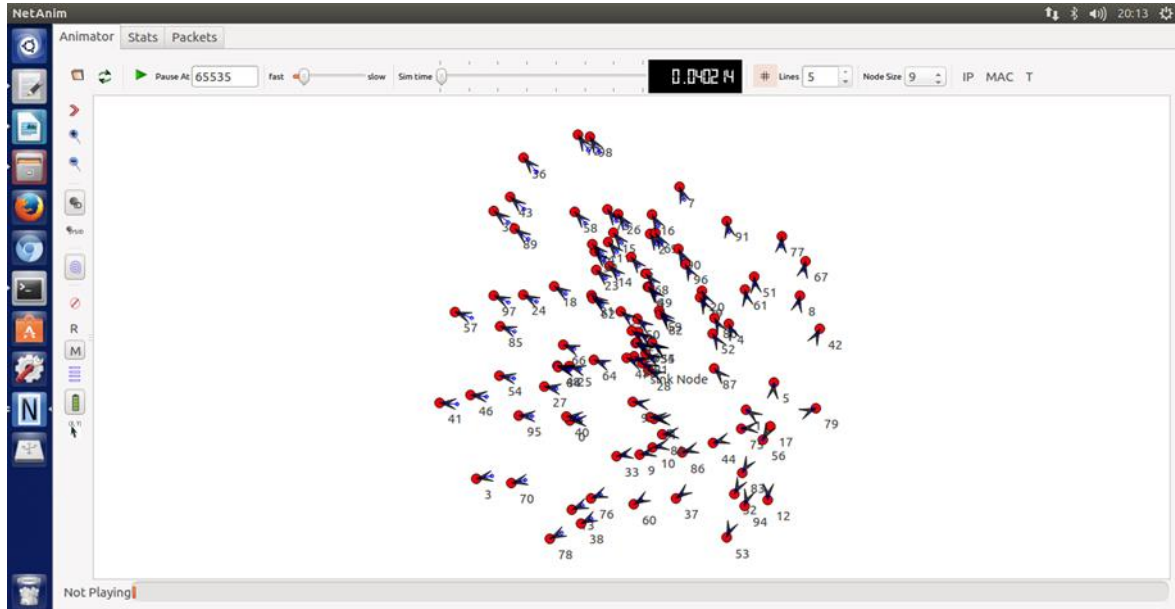


**Figure 3.10:** SDWSN packets flow illustration.

Sensor nodes transmit packet flows cooperatively to rich dedicated or targeted nodes within the network. As flow instructions are initiated and implemented by the ODL controller, the OpenFlow protocol facilitates communication between the controller and the developed network infrastructure. Hundred sensor nodes are modelled in NS-3 and receive communication instruction through the OpenFlow protocol. The sink node forms part of the sensor network but has better communication capacity than the normal sensor nodes. The sink nodes relays packet flows to the ODL controller for further network operations. In this simulated scenario, packets are generated in terms of status check on each node. The network infrastructure receives the controller instruction and perform data transmission accordingly.



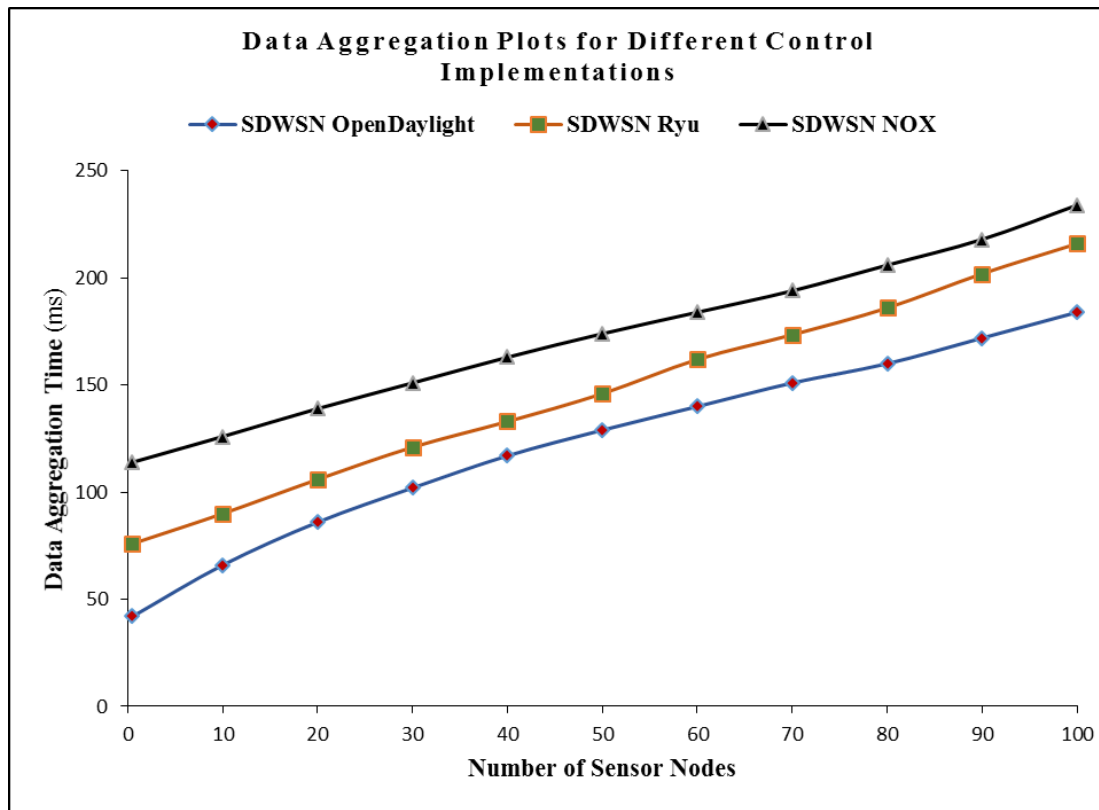
Sensor nodes transmit to their neighbours and until communication reaches a formed cluster for associated cluster queries.



**Figure 3.11:** Random sensor traffic transmission

To validate network flexibility and simplified resource management, the framework was tested by applying data manipulation and configuration operations on network resource. Service committing functions were checked against the original status of network resources such as abstracted services and commanding network devices. Device access and response times for various network transactions were confirmed by checking changes in network behaviour. Our experiments indicate that using YANG models and MD-SAL services on the implanted approach, improved network resource access and manipulation. Services such as changing flow routing information and abstracted services –in our case checking cluster level status was performed with considerable simplicity and flexibility.

Figure 3.12 shows data aggregation plot for three different SDN controllers, whereas Figure 3.13 shows packet error rate over sensor traffic for different throughputs.

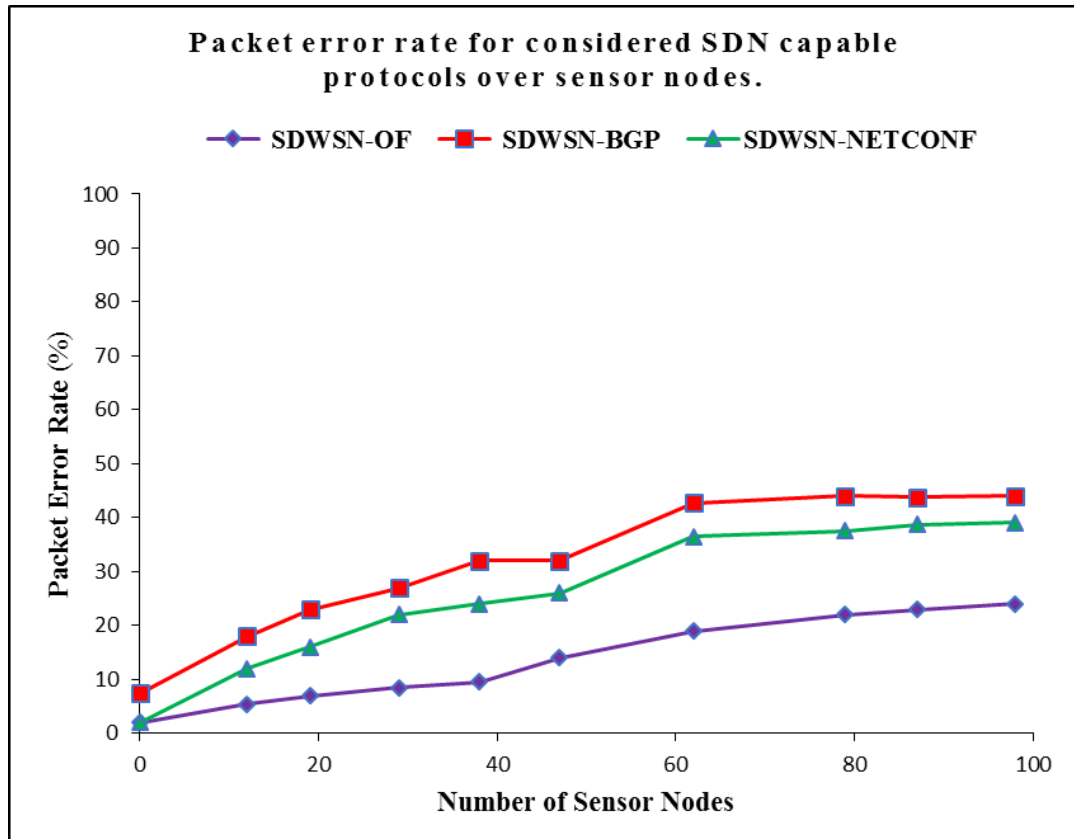


**Figure 3.12:** Data Aggregation Plot for Three Different SDN Controllers.

From Figure 3.12, it can be seen that; the systematic setup using ODL SDN controller performed best compared to both the implementations using Ryu and NOX SDN controllers. The ODL implementation produced lower aggregation times for data aggregation compared to both Ryu and NOX. This is a significant phenomenon by the ODL setup as it proves to have the least communication delay compared to the other two setups. To further test the proposed implementation, a packet error estimation was also observed as in Figure 3.13, on separate setups using an external controller implementation (when the controller and the simulated network are on different VMs) and a local controller implementation (when the controller and the simulated network are under the same VM and same network setup).

Figure 3.13, shows packet error rate plot for applied SDN capable communication protocols. SDN capable communication protocols considered for this work are; OpenFlow, BGP and NETCONF (in particular Request for Comments (RFC) 6241), since this version has SDN

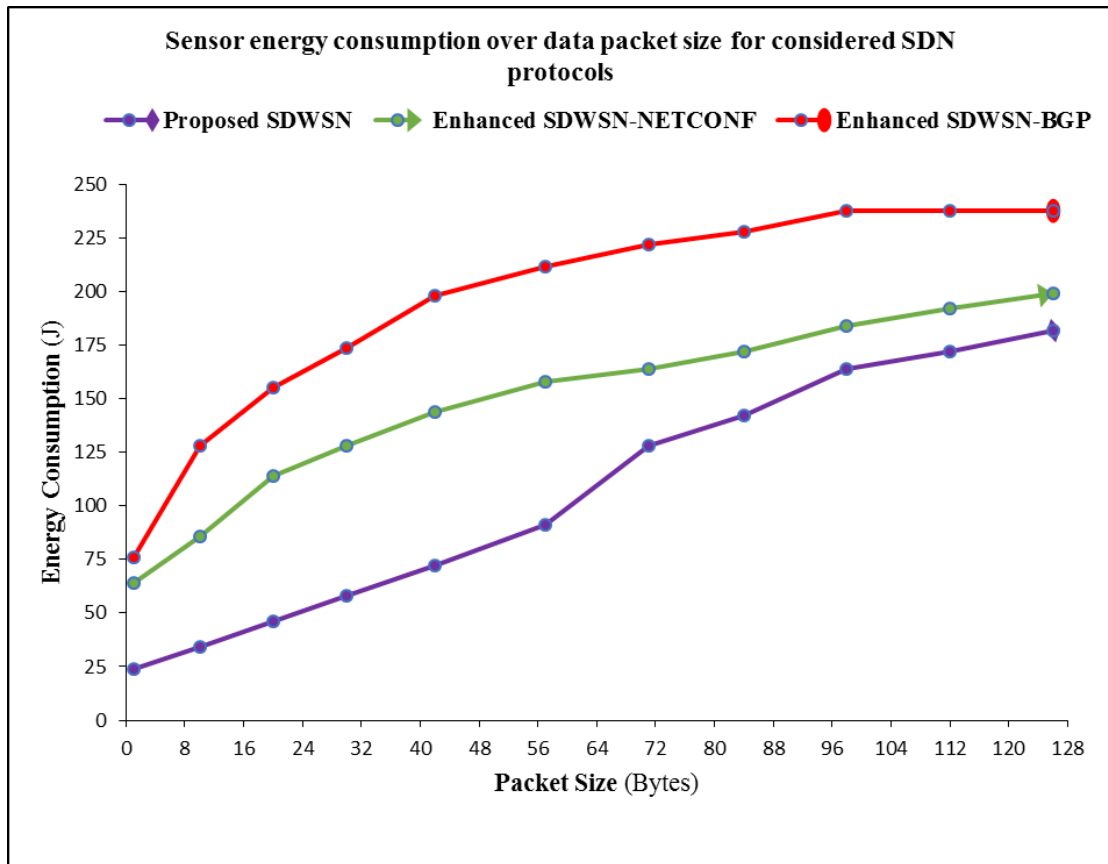
communication capabilities and can support proprietary network devices that have NETCONF interfaces.



**Figure 3.13:** Packet Error Rate per Number of Sensor Nodes.

Our experimental results show BGP and NETCONF indicating almost the same performance for successful packet sent from the source device as compared to packet received at any destination device. In average, these two communication protocols indicate packet error rate of about 60%, whereas the OpenFlow protocol produced only 18% packet error rate. However, there is potential to optimize these protocols using SDN strategies according to system design requirements. Based on our experiments, these protocols outperform each other on different network operations. For example, NETCONF performs better than both OpenFlow and BGP in terms of configuration speed. BGP is most efficient for traffic routing between base stations of gateway devices and the internet. OpenFlow promotes network innovation as proposed by the SDN strategy for abstract network computing. Hence, these

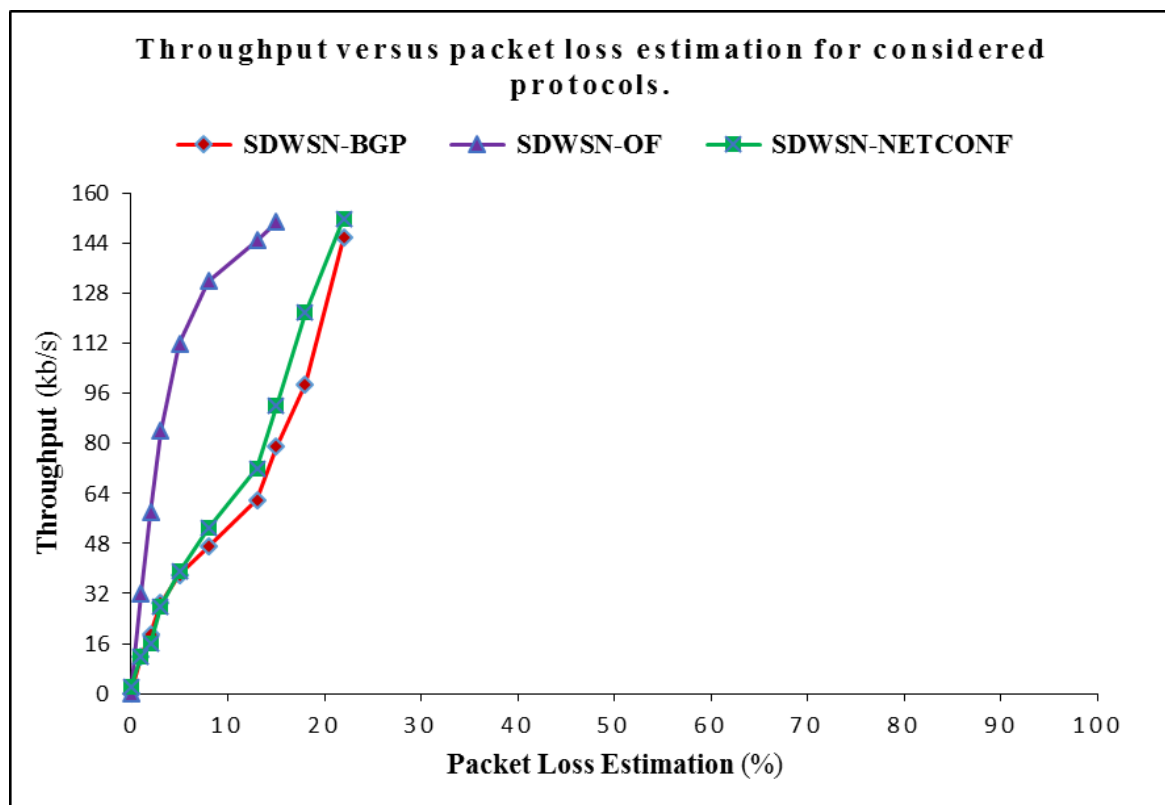
communication technologies can be used together to achieve different goals within one network.



**Figure 3.14:** Energy Consumption for over Data Packet Size for Considered SDN Protocols.

FIGURE 3.14 shows a performance comparison for three different SDWSN implementation strategies for sensor nodes energy consumption versus data packet size. Performance comparisons were done amongst the proposed SDWSN, the enhanced SDWSN-NETCONF and the enhanced SDWSN-BGP strategies. The two enhanced versions of NETCONF and BGP use the OpenFlow SDN data forwarding techniques as an enhancement approach. These two protocols use QoS routing roles defined by the SDN controller as service parameters to ensure efficient transmission from the controller to data forwarding plane. Hence these protocols are SDN enhanced to support software-oriented strategies as defined by SDN framework. However, the proposed strategy indicated efficiency in energy utilization due to its simplicity of packet routing by the OpenFlow.

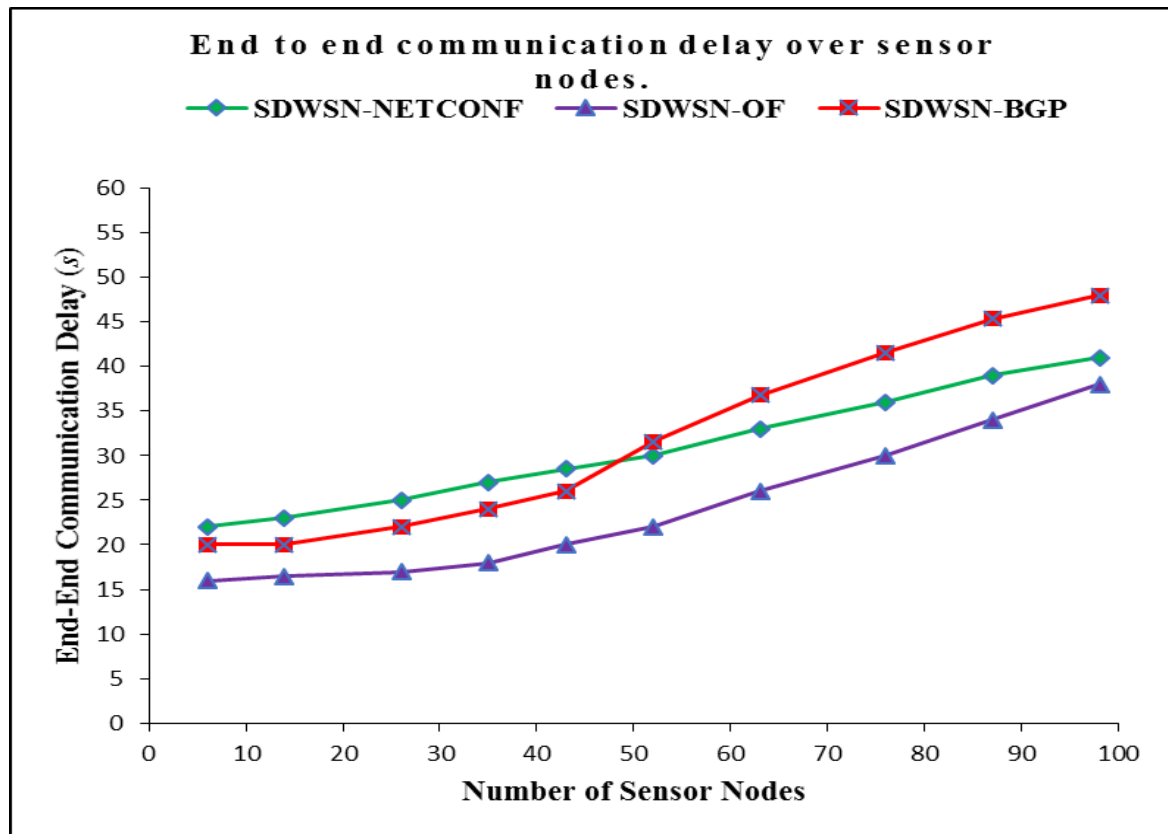
As opposed to this, the enhanced SDWSN-NETCONF showed better performance compared to the enhanced SDWSN-BGP approach in terms of energy consumption. This is mainly due to the processes that are involved in these protocols for packet or data transmission. On the other hand, the enhanced BGP version showed remarkable results in terms of remote connectivity even though some challenges were experienced in terms of routing updates in the case where connectivity was lost. This also results in high energy utilization since targeted path updates should be rescanned to establish any defects.



**Figure 3.15:** Packet Loss Estimation over Throughput.

FIGURE 3.15 shows a performance comparison difference in packet loss estimation over data throughput that becomes successively transmitted within the network from source to destination nodes. Yet again, the OpenFlow implementation of the proposed system produced the best results in terms of packet loss estimation compared to both the NETCONF and BGP system implementation. The proposed system produced just about 14% packet loss estimation on average, whereas the NETCONF and BGP implementation produced almost the same rate

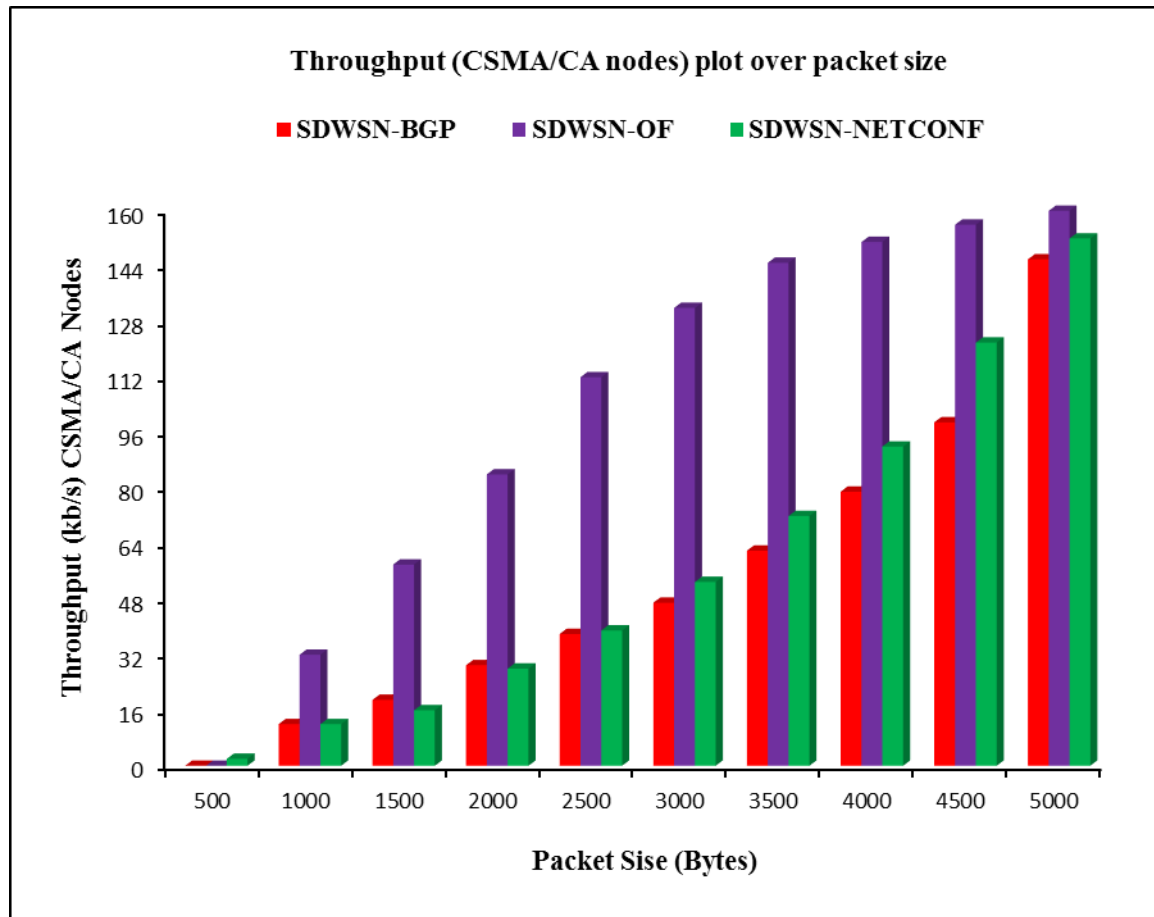
of packet loss. On average the BGP implementation produced about 22% performed just below whereas the NETCONF implementation produced 20% packet loss when estimated. This is a significant achievement by the OpenFlow implementation towards the overall improvement of WSN application systems. However, the NETCONF strategy showed a lot of potential for SDN applications especially in terms of device configuration.



**Figure 3.16:** End To End Delay Over Sensor Nodes.

FIGURE 3.16 shows a plot of associated end-to-end delay of the applied SDN communication protocols. The BGP implementation shows least performance however, almost the exact performance like that of the NETCONF implementation. The NETCONF strategy produced an average of about 26 seconds end-to-end delay compared to the BGP implementation which produced end-to-end delay performance of about 34 seconds on average. The NETCONF implementation was observed to be almost constant in terms of delay for aggregated sensor data, whereas the BGP strategy seemed to be gradually experiencing some small amount of

delay. The OpenFlow implementation produced best results compared to both the NETCONF and BGP implementations with about 18 seconds. Another observation was that OpenFlow produced minimal increase in delay for high data aggregation of a larger number of sensor nodes.



**Figure 3.17:** Throughput Plot over Transmitted Data.

FIGURE 3.17 shows a plot of throughput versus sensor transmitted packets. For sending packets over the communication channel, an experiment was performed with sensor nodes set to use Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA) in an effort to avoid packet drops due to collision. Again, the OpenFlow implementation indicated high throughput support compared to both the NETCONF and BGP versions. This achievement by the OpenFlow shows significant performance towards its use for SDN based WSNs applications. However, improvements can still be done on this communication standard to

enhance it for full scale SDN developments of any structure or design. On the other hand, NETCONF indicated incredible potential SDN communication tasks, especially for network configuration purposes. Even though in this instance BGP produced the least performance, it still holds significant capabilities for remote communication.

However, for very small packet sizes, transmission throughput tends to lower or decrease due to the transmission overhead introduced on the communication channel. This is because such smaller packet sizes, creates high transmission rate as they are quickly captured and transmitted through the channel and as a result creates serious communication overhead. In addition, systematic improvements on any computing scheme requires the best technologies for optimum performance, therefore, it remains the choice of the designer to opt for certain techniques over others for as long as best performances can be achieved.

Unlike previous works which mainly focused on energy management, communication overhead, load balancing and resource management, this work implements SDN communication mechanism with QoS role functionality to promote application abstraction for SDWSN systems. It further, applies software strategies as a driver to simplify network management and ensure network flexibility and adaptability. Given the programmability orientation of our approach, our experimental work considers with thorough focus, the impact of the proposed methods towards netter system intelligence, using object-oriented abstracted network services. Our analysis are mostly informed through SDN strategies and machine learning methods in terms of the overall system performance regarding efficient knowledge acquisition.

### **3.5 CHAPTER CONCLUSION**

This work proposed a SDWSN approach in an effort to improve the overall network performance especially for mission critical applications. NS3 network simulator was used for modelling a WSN, with an ODL controller used for the network intelligence. Two other SDN controllers – Ryu and NOX were also tested against the Implemented ODL controller to evaluate its performance. Performance comparison were done to effectively distinguish the best performing SDN controller and protocol for SDN developments in particular for this



work. In terms of southbound communication performances, two SDN capable protocols namely; BGP and NETCONF, were used and compared with OpenFlow for our SDWSN implementation. Our experimental results produced best and least performances for different tests.

For controller performance efficiencies, ODL produced the best results, with Ryu coming second and NOX with the least performance. The ODL implementation produced lower aggregation times for data aggregation compared to both Ryu and NOX. Even though Ryu and NOX produced lower performances, they showed great potential in terms of managing network devices. Hence these SDN controllers can still be optimized to offer best performances. The ODL controller together with the OpenFlow protocol, produced the best results in terms of; data aggregation over sensor data transmission, packet delivery ratio, end-to-end delay time and energy utilization. The NETCONF protocol can also be a choice communication protocol for SDN application systems if it could be fully optimized for such, since this protocol produced promising results in all experiments conducted. This however, does not rule out Ryu as an alternate controller since it also has great potential for simple SDN applications. Otherwise it could be enhanced for high end SDN computing applications.

Our proposed approach showed significant improvements on the overall network performance for SDN based WSN applications. Particularly in terms of network flexibility resource management and network adaptability. Notable levels of flexibility have also been achieved, as SDN services such as remote access, network device management and simple updates could be done with ease. ODL provided simpler methods to support the SDN southbound interface to easily communicate with underlying network devices using its Service Chaining Functionality (SCF). However, this implementation can still be enhanced or optimized to fully support the SDN programmability as this will greatly promoted SDN adoption.

# CHAPTER 4      APPLYING                      MACHINE LEARNING TECHNIQUES IN SDWSN

## 4.1    CHAPTER OBJECTIVES

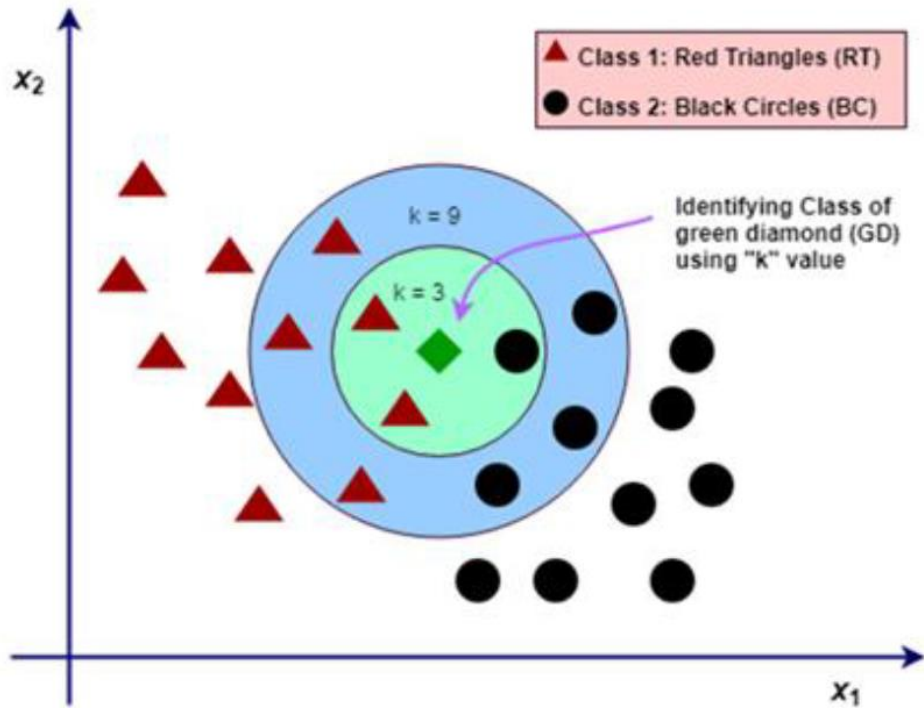
In this work, COSPA- an algorithm for sensor monitoring applications in SDWN systems whose functionality is complemented by collating its information acquisition method to that of the further proposed KNN CLIQ. This algorithm uses, open programmable SDN strategies for efficient sensor data aggregation and to also present sensor queried data in a more meaningful manner.

## 4.2    THE KNN ALGORITHM

To understand the computational process of the KNN algorithm, it is critical to have knowledge on how its input parameter impacts on the operational output. The use of KNN for regression-based predictions and query point classification forms its essential capability and popularity of application. Even though KNN is known for its performance in the problems mentioned, classification stands out to be its common application. Based on the distance function computation, the KNN builds new classification cases from the stored ones. This functionality describes the impact of the distance metric and the value of  $k$  applied during the KNN operation. The KNN algorithm is prevalently applied in data mining fields given its performance capability and simplicity of implementation. Popular application areas of KNN includes, but not limited to the following:

- **Pattern Recognition for Retail Operations.** In this case, the KNN can be used to perform matching functionalities on purchased retail items against their information on the retail database. It can also be used to avoid card fraud for customers using cards to make purchases. If customer information captured on the system manually rather than transacting with the actual card, the system can flag a security alert.
- **Recommendation Systems.** In this case, it can be used for predicting an online customer's commonly purchased products or items, and then recommend related selling products to specific customers based on their purchase history.
- **Computer Vision.** Two different objects of the same form can be scanned and compared for similarities using the KNN algorithm.
- **Gene Selection and Expression.** In this case, the algorithm can be used for performing a selection classification for gene test samples. A prediction can be made based on different classes of samples according to their properties.
- **Concept Search.** In this case, KNN can be used to perform a search for similar items. An example is when searching for documents with similar object names.

Figure 4.1 illustrates the operation of a KNN algorithm given different classes of feature vectors. An illustration in this instance is that; if there are  $N$  feature vectors belonging to classes 1 and 2, denoted in Red Triangles (RT) and Black Circles (BC). Where, RT's and BC's, are in a two-dimensional (2D) feature region as shown, the KNN algorithm identifies the class of "K" nearest neighbours of the Green Diamond (GD) which is another feature vector, using a distance method of classification. The classification is performed in a bounded region using the value of  $k$ . In addition to this, the distance metric or algorithm used provides the query coverage on the specific object being classified.



**Figure 4.1:** Classification operation of KNN algorithm using optimal “k” values.

#### 4.2.1 KNN CLIQ

KNN CLIQ is a modified supervised machine-learning algorithm to enhance the system’s information knowledge. KNN CLIQ adopts the original KNN’s functionality, with a slight programming modification for network event acquisition. In our approach, KNN CLIQ is implemented with focused technicality for cluster level querying and classification. To compute a case classification for sensor data query information, we apply the Euclidian distance function relative to its neighbours.

In addition to the original KNN’s functionality, the KNN CLIQ’s query region is abstracted, wherein light SDN strategies are appended to its processing for efficient cluster propagation and querying. In KNN formation, “K” represents a number of query points or samples in a specified “nearest neighbours” region wherein a query is propagated to make an estimation. Machine learning has the capacity to also evolve network computing systems especially where massive amounts of data is key for reliable system operation. In that regard, machine learning shows greater potential to be applied in WSNs, given the nature of data processing and transmission of sensor nodes. Sensor devices deal with the transmission of huge and

event-based data to fulfil their system purposes. Hence, this makes them relevant for using machine-learning techniques to advance their networking functionalities. The choice towards this machine-learning algorithm was informed by the basis that it is not computationally expensive and can be easily integrated with other applications since it not complex and lastly, it has the relevant computing methods for data aggregation.

### 4.3 EXPERIMENTAL SETUP

This implementation uses C++ as the core programming language for developing abstracted light-weight network services. The ODL framework provides a perfect fit for programmable networks, through its full support for SDN technologies. The network is implemented in NS3, which also allows the definition of container-based function to be written in C++. NS3 is a highly programmable discrete event simulator for distinct types of networks. It also allows Python scripts to be equally written as it does with C++.

The ODL controller is implemented on a separate VM to that of the modelled network. Communication between the ODL controller and the modelled network is achieved through the OpenFlow protocol (ver. 1.5.1). Simple network services such as changing network applications parameters can be performed on the controller. The controller then subjects these activities to the target applications. This implementation uses the Oxygen ODL controller, given its support for integrated workloads from VMs and containers. Provided below, in table 1, is a description of settings parameters of our experiments. Query observations are performed as specified from the ODL controller. The controller also sets the number of observation to be performed on a particular region.

This implementation uses an OpenDayLight (ODL) SDN controller as its central intelligence. Communication between the ODL controller and the underlying sensor network is achieved through the OpenFlow communication protocol. The system uses application modularity method to associate the KNN functionality process to the proposed (COSPA) algorithm's operational functionality. The ODL's MD-SAL and the YANG data model enable this association. The network is modelled using NS3 (Network Simulator 3). New network services are applied to the modelled network using the ODL's service abstraction

capability and the SDN programmability strategies. This approach takes advantage from the NS3's capability to include programming definitions to its network modelling interface.

The modelling language used for programming system definitions and applications is C++. This Object-Oriented Programming (OOP) language provides rich functionalities for writing dynamic computing applications. Given this new realization of SDN strategies in WSNs, it is worth mentioning that, not much has been done in applying machine-learning methods particularly in monitoring SDWSN systems. This implementation is one of the few based on the related work in this area.

Table 4.1 illustrates symbols and parameters used in this work. Note that each symbol or parameter is given a brief description in terms of how it was used or applied in either a mathematical model or an algorithm.

**Table 4.1:** Notations and parameters used, with their brief descriptions.

Notations/Parameters	Description
$u$	A test point in a query space or region.
$\{v_1, \dots, v_k\}$	Nearest neighbours of $u$ , in a classified query region.
$c(u)$	A determined class of $u$ .
$k$	A number of query points or samples in a specified "nearest neighbours" region.
$c(v_i)$	A class, $c$ of a particular neighbour $v_i$ .
$\delta$	A determining function for $c(v_i)$ , i.e. $\delta(q, r) = 1$ if $q = r$ .
$\hat{p}(c u)$	Probability function for the class $c$ , given $u$
$d_{Ec}(a, b)$	The Euclidian distance between two points $(a, b)$ , in a dimensional space, $D$ .
$D_{cr}$	Detection capacity of a particular sensor node in terms of range.
$Q = \{(x_1, c_1), \dots, (x_N, c_N)\}$	The actual query process, performed in various points, $\{(x_1, c_1), \dots, (x_N, c_N)\}$ .
$R = (w_1, \dots, w_n)$	KNN query results $(w_1, \dots, w_n)$ , in a region $R$ .
$(x_i, c_j)$	Results instances from the KNN operation.
$(x_i, w_j)$	Similarity instances matched against the query point during the KNN process.
$D_{w_j}^k$	Classified $k$ nearest objects to $w_j$ , from the KNN operation.

---

$(w_j, c_i)$ and $(w_j, c_i)$	Different class instances based on the object $w_j$ .
-------------------------------	---

---

This implementation uses C++ as the core programming language for developing abstracted light-weight network services. The ODL framework provides a perfect fit for programmable networks, through its full support for SDN technologies. The network is implemented in NS3, which also allows the definition of container-based function to be written in C++. NS3 is a highly programmable discrete event simulator for different types of networks. It also allows Python scripts to be equally written as it does with C++.

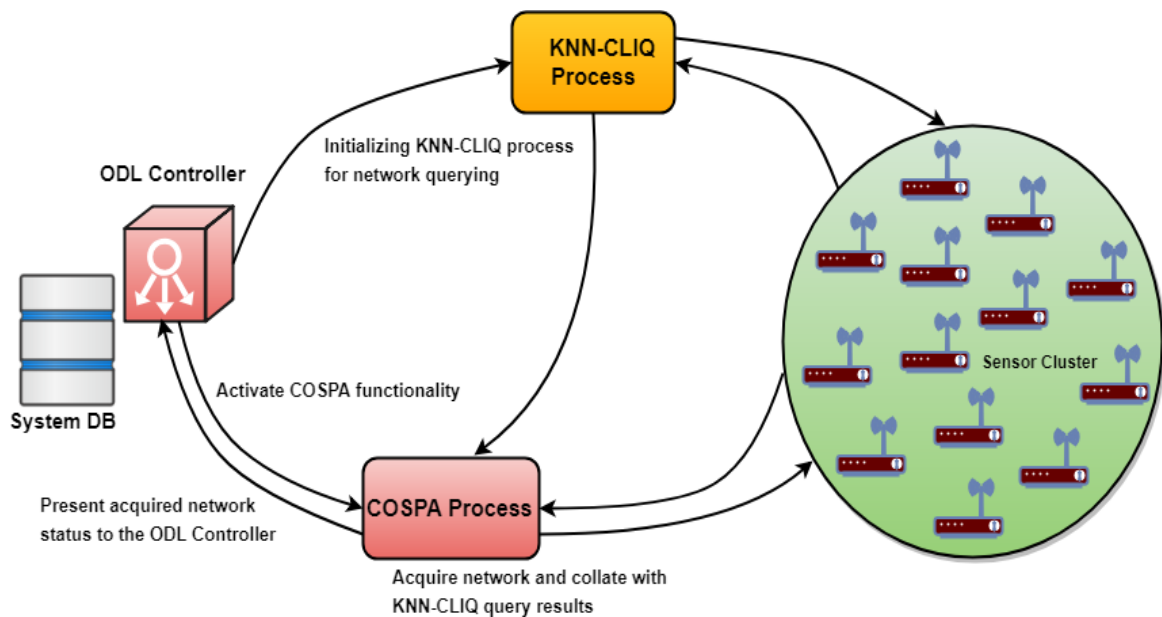
The ODL controller is implemented on a separate VM to that of the modelled network. Communication between the ODL controller and the modelled network is achieved through the OpenFlow protocol (ver. 1.5.1). Simple network services such as changing network applications parameters can be performed on the controller. The controller then subjects these activities to the target applications. This implementation uses the Oxygen ODL controller, given its support for integrated workloads from VMs and containers. Provided below, in Table 4.2, is a description of settings parameters of our experiments. Query observations are performed as specified from the ODL controller. The controller also sets the number of observation to be performed on a particular region.

**Table 4.2:** System parameters and values

Parameter	Value / Unit Size
Node Mac/Phy Protocol	IEEE 802.15.4
SDN standard protocol	OpenFlow 1.5.1
Network Size	600*900
Node Count	100 sensor nodes
Aggregate Efficiency	Sensor data / Total no. of nodes
Simulation Time	60 (s)

Our implementation couples the capability of COSPA's SDN programmability functionality and processing with an enhanced KNN machine learning algorithm- KNN CLIQ. The KNN

CLIQ's functionality implemented using object-oriented strategies for reading data so as to improve its data querying functionality. The coupling of these two independent algorithms serves the SDN's capacity to simplify the network, introduce flexibility and innovation, and yet increasing its operational power. This coupling is facilitated by means of abstracting mutual information parsing of both these algorithms. The KNN CLIQ's query results are conceded to the COSPA's acquisition process for collation and processing. The mastered network status information is then presented to the ODL controller for knowledge processing and information presentation. Figure 4.2 illustrates different processes flow of the proposed methods in conjunction with their cooperation with the ODL controller. A dedicated application server with storage capacity is setup to keep the ODL controller's log information for all network processes.



**Figure 4.2:** Processes Flows of both KNN CLIQ and the COSPA Algorithms Relative to the ODL Controller.

Based on our implementation, these processes operate on the network as initialized by the controller. The controller determines the activation of these algorithms' functionality as set in the network control settings. The classification operation of the KNN algorithm, given a number of feature vectors of different classes, is explained in a mathematical model below. Suppose, a test point  $u$ , and its  $k$  nearest neighbors  $\{v_1, \dots, v_k\}$ , are associated to this



test point and test voting is performed to estimate the most associative class of  $u$ , in a dimensional region, i.e. class of  $u$ , represented by  $c(u)$ . Then,  $c(u)$  is given by:

$$c(u) = \arg \max_{c \in C} \sum_{i=1}^k \delta(c, c(v_i)) \quad (4.1)$$

, whereby,  $c(v_i)$  represents the class of  $v_i$ , and  $\delta$  is a determining function, i.e.  $\delta(q, r) = 1$  if  $q = r$ . Thus, using a simple vote, KNN produces an estimation  $\hat{p}(c|u)$ , which is given by:

$$\hat{p}(c|u) = \frac{\sum_{i=1}^k \delta(c, c(v_i))}{k} \quad (4.2)$$

The estimation can also be presented in terms of the cumulative distribution function (CDF) for a given observation. This function produces an observation probability on a certain variable based on a determinant factor. The CDF can either be expressed in terms of a continuous or discrete distribution. A normal CDF is given in terms of the equation below:

$$\begin{aligned} CDF &= F(x) \\ &= P_{obv}[Q \leq q] \\ &= \alpha \end{aligned} \quad 4.3$$

, where  $P_{obv}$  is the observation probability in terms of the estimation variable  $Q$  which is approximated using the determinant factor  $q$ . The  $\alpha$  is the evaluated distribution function value based on the probability.

The Euclidian distance algorithm is given below, to make reference as to how the distance measure is calculated. We note that, given a D-dimensional region, the Euclidian distance,  $d_{Ec}$  of feature vectors  $a$  and  $b$  is given as in equation 3 below:

$$d_{Ec}(a, b) = \left[ \sum_{d=1}^D (a_d - b_d)^2 \right]^{\frac{1}{2}} \quad (4.4)$$

If we apply this distance measure to only two feature vectors, then their Euclidian distance equation can be simplified as in equation 2 below:

$$\begin{aligned}
 d_{Ec}(a, b) &= d_{Ec}(b, a) \\
 &= \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}
 \end{aligned} \tag{4.5}$$

This will hold provided that, on the Euclidian plane,  $a = (a_1, a_2)$  and  $b = (b_1, b_2)$ . Suppose that, in a query region  $R$ , a sensor node with optimal transmission energy, has a transmission radius  $r_k$ , given this region. However, taking note that, the transmission radii of all sensor nodes will not be equal given their different locations, i.e.  $r_n < r$ . Therefore, the detection capacity,  $D_{cr}$  of a particular sensor node in terms of range, can be given as in the equation below, which is adopted from the Area of a Circle.

$$D_{cr} = \pi r_k^2 \tag{4.6}$$

Algorithm 4.1, represents the adopted original KNN procedure. Pseudo codes of both the proposed KNN CLIQ and the COSPA algorithms are shown in procedures 3 and 4 respectively. The original KNN is also implemented in this work, for the purpose of efficiency comparisons with the proposed algorithms. The KNN CLIQ takes almost the same parameters as the original KNN, with slight SDN programming strategies.

**Algorithm 4.1:** The adopted original KNN algorithm

<b>Algorithm 1: The adopted original KNN algorithm</b>
<p><b>Input:</b> <math>Q = \{(x_1, c_1), \dots, (x_N, c_N)\}</math> // query data points  <math>R = (w_1, \dots, w_n)</math> // query region  <math>\rightarrow</math> // object to be classified  <math>k</math> // determinant value of nearest neighbors</p> <p><b>Output:</b> <math>R = \{(w_1, c_1), \dots, (w_n, c_n)\}</math> // kNN query results</p> <pre> 01 for ( each result instance <math>(x_i, c_j)</math> ) do 02   perfCal similarity<math>(x_i, w_j)</math>; 03   orderIn similarity<math>(x_i, w_j)</math> ascending, <math>(i=1, \dots, N)</math> 04   classify <math>k</math> nearest objects to <math>w_j</math>: <math>D_{w_j}^k</math> 05     for each <math>D_{w_j}^k</math> do 06       if <math>D_{w_j}^k \neq 1</math> then 07         create class instance <math>(w_j, c_i)</math> { <math>c_i = 0</math> } 08       else 09         create class instance <math>(w_j, c_i)</math> { <math>c_j = 1</math> } 10     end_for 11 end_for </pre>

Based on our implementation, the KNN CLIQ algorithm uses the definition process of the adopted KNN, defined globally on the developmental environment. The pseudo code of the proposed KNN CLIQ is presented in Algorithm 4.2.

**Algorithm 4.2:** The proposed KNN-CLIQ algorithm

Algorithm 2: The Proposed KNN CLIQ algorithm
<pre> 01 <b>init.</b> KNN_Cliq( _process) // start KNN_Cliq process  // Include COSPA for functional process 02 <b>include</b> "COSPA" // include COSPA to KNN_Cliq 03 <b>template</b> &lt;class netInter&gt; // net interface template 04 KNN_Cliq netInter(R, senID, senRad(_rQueryObject)) 05     <b>vector</b> &lt;netInter&gt; netVector(R) 06     <b>read:</b> → netInter( netVector(R) )  // Perform conditional processes 07 <b>while</b> ( this → <math>D_{w_j}^k</math> ) <b>do</b> 08     <i>traverse R: → <math>D_{cr}</math> to get sensor details</i> 09     netInter.R (netState(senID, remEn))  10     <b>if</b> senState != netState(senID, remEn) 11         <i>output: sensor not active</i> 12     <b>else</b> 13         <i>output: (sensor_details   R)</i> 14     <b>end_while</b>  // Copy KNN_Cliq results to COSPA, using netInter* 15 KNN_Cliq netInter.COSPA(netState(senID, remEn)) 16 netInter* KNN_Cliq = new netInter[_res] 17 netInter* COSPA = new netInter[_res] 18 copy(KNN_Cliq, KNN_Cliq + _res, COSPA) // Now exit the KNN_Cliq process 16 <b>exit_ KNN_Cliq</b> </pre>

A process flowchart of the KNN-CLIQ algorithm is provided in Figure 4.3. Applying the adopted procedural functionality of the original KNN, the proposed KNN-CLIQ adds SDN programmable functionalities for effective query performance. It implements a standard template library (STL) to define a class interface, which is used to access the examined query region. It also implements a vector interface to enable efficient query data presentation based on the observed region. The most important functionality of KNN-CLIQ algorithm is interface the COSPA algorithm for aligning its query results with those of the COSPA procedure. Conditional processes are then performed to traverse the network with the purposed of acquiring the current cluster status. The ultimate results of the KNN-CLIQ are

then mapped to the COSPA for implementing their coupled procedure. Figure 4.3, illustrates the functional processes of the KNN-CLIQ in terms of a flowchart.

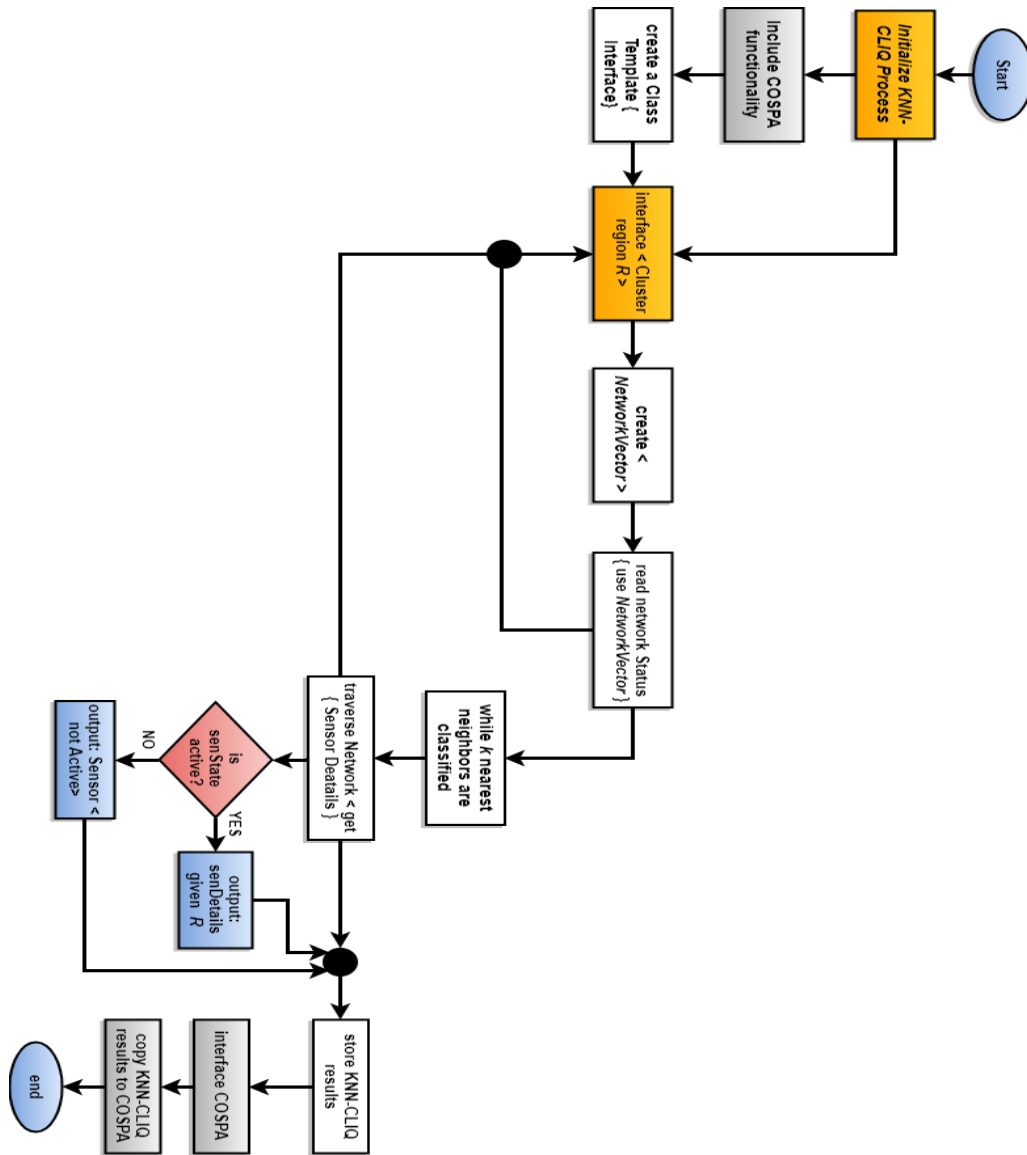


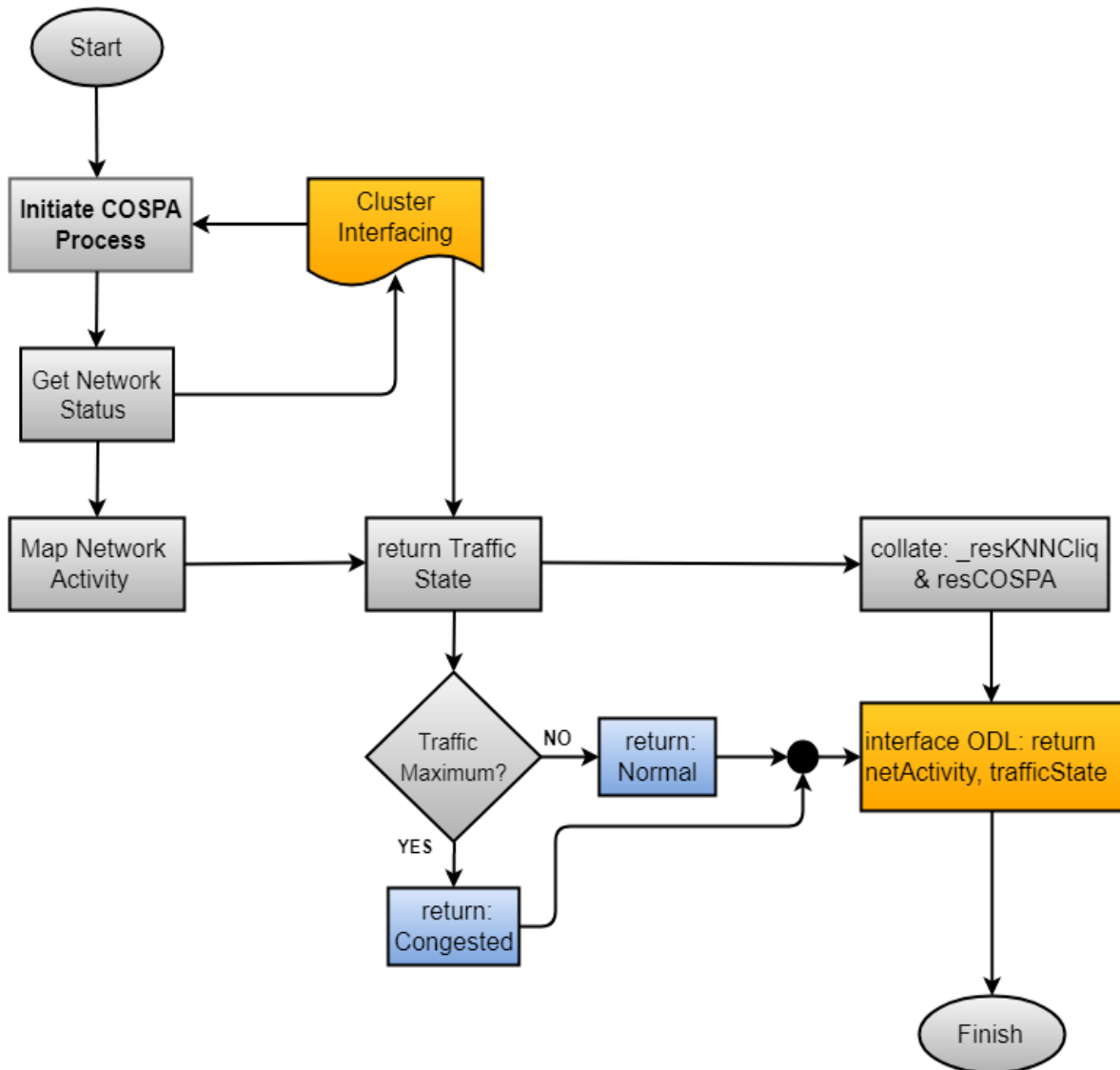
Figure 4.3: Process flowchart of the KNN-CLIQ algorithm.

Both the KNN CLIQ and COSPA algorithms, provides the core functionalities of the proposed system. The pseudo code of the proposed COSPA algorithm is presented in Algorithm 4.3. The ability of COSPA of obtaining KNN-CLIQ query results as its input makes it an intelligent algorithm, given that the KNN-CLIQ query results are first organized then effectively collated with the current COSPA results.

**Algorithm 4.3:** The proposed COSPA algorithm.

<b>Algorithm 3: The Proposed Controller based Open Status Presentation Algorithm (COSPA)</b>
<p><b>Input:</b> <i>KNN_Cliq</i> results {<i>_resknnCliq</i>} // take as input  <b>Output:</b> <i>_process</i> results to the ODL controller</p>
<pre> 01 <b>init.</b> COSPA functionality {<i>_process</i>} // start procedure 02 <i>interface:</i> cluster { <i>_nodes</i>   <i>R</i> } 03 <i>get():</i> { <i>netState</i>(<i>netLoad</i>   <i>R</i>) // get cluster net status 04 <i>netInter</i> set( new <i>netState</i> (<i>netActivity</i>, <i>noNodes</i>)) 05 <i>map::netState</i> (<i>this</i> → <i>netActivity</i> ( <i>clusterLoc</i>   <i>R</i> ) 06 <i>read::netState</i>( <i>get</i> {<i>netInter</i> (return <i>trafficState</i>())} )  07 <b>if</b> (<i>trafficCap</i> != <i>maximum</i>) // if traffic is normal 08   <i>output:</i> <i>netState</i> ( return <i>str</i>( <i>normal</i> ) ) 09 <b>end_if</b> 10   <b>else</b> 11     <i>output:</i> <i>netState</i> ( return <i>str</i>( <i>congested</i> ) ) 12   <b>end_else</b> 13 <i>collate:</i> <i>this</i> → <i>resultC</i>(<i>_resknnCliq</i>, &amp;<i>resCOSPA</i>) 14 <i>netInter</i> <i>_ODL:</i> <i>this</i> → <i>netState</i>(<i>netActivity</i>(), <i>trafficCap</i>) 15 <b>exit_COSPA</b> </pre>

To understand the procedure of the COSPA algorithm, Figure 4.4 illustrates the processes and routines that forms its capability in terms of a flowchart. This algorithm provides a cluster interfacing capability to access associated cluster sensors. This functionality is extremely important as it allows simplified acquisition of sensor information. The algorithm performs various processes depending on the activity of the network. Traffic state can be retrieved and get sent to the ODL controller for statistics. The results collation functionality of COSPA, allows it to acquire and match its query results with that of the KNN-CLIQ, then builds evaluative results, which is then presented to the ODL controller.



**Figure 4.4:** A flow diagram of the COSPA algorithm.

The ODL controller provides rich capabilities for writing simple network applications. Simple network configurations can be performed from the controller. Our system provides some level of process granularity, as parts of the system's functionalities can be performed independently. This is a key feature of our system as it advances the flexibility of the system. To get the distribution probability value for the number of query observations in our experiments, we adopt the Cumulative Distribution Function (CDF), which is given as;

$$\begin{aligned}
 f(k: n, \hat{p}) &= \hat{P}(X_{ran} \leq k) \\
 &= \sum_{i=0}^{|k|} \binom{n}{i} p^i (1-p)^{n-i}
 \end{aligned} \tag{4.7}$$

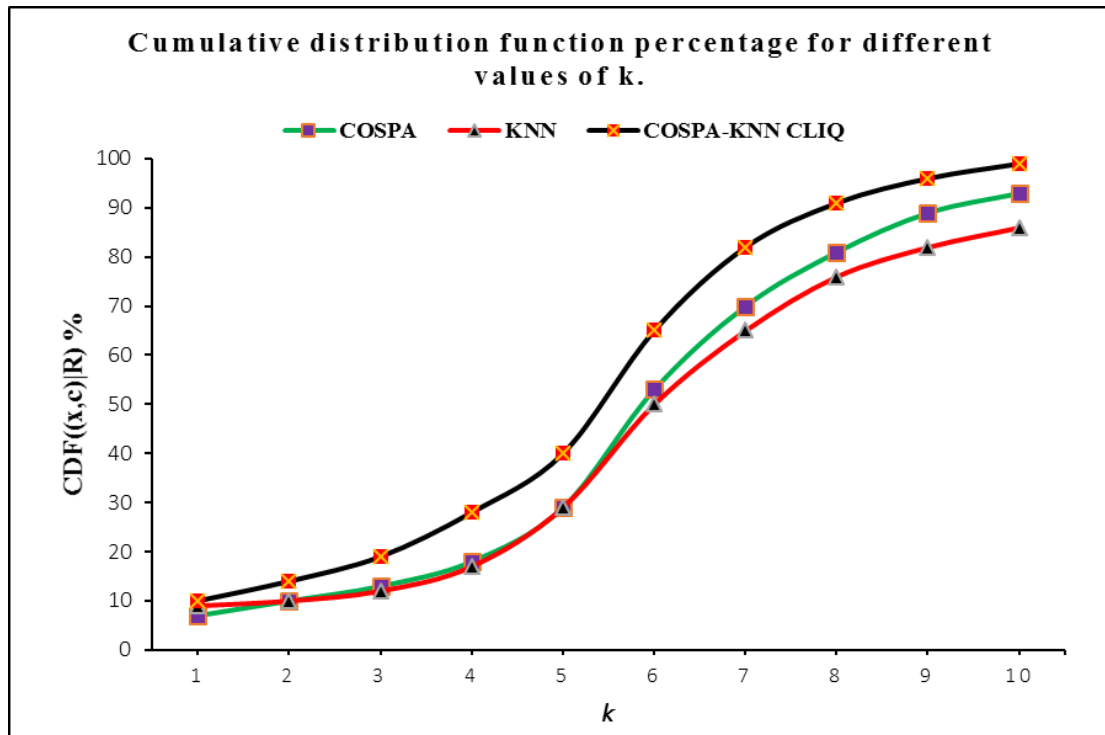
, where,  $X_{ran}$  is a random value point  $X$  and  $n$  denotes the number of observations in a query region given  $X$ -number of points.

#### 4.4 EXPERIMENTAL RESULTS

Given the programmability orientation of our approach, our experimental work considers with thorough focus, the impact of the proposed methods towards increased system intelligence, using object-oriented abstracted network services. Our analysis is informed through SDN strategies and machine learning methods in terms of the overall system performance regarding efficient knowledge acquisition.

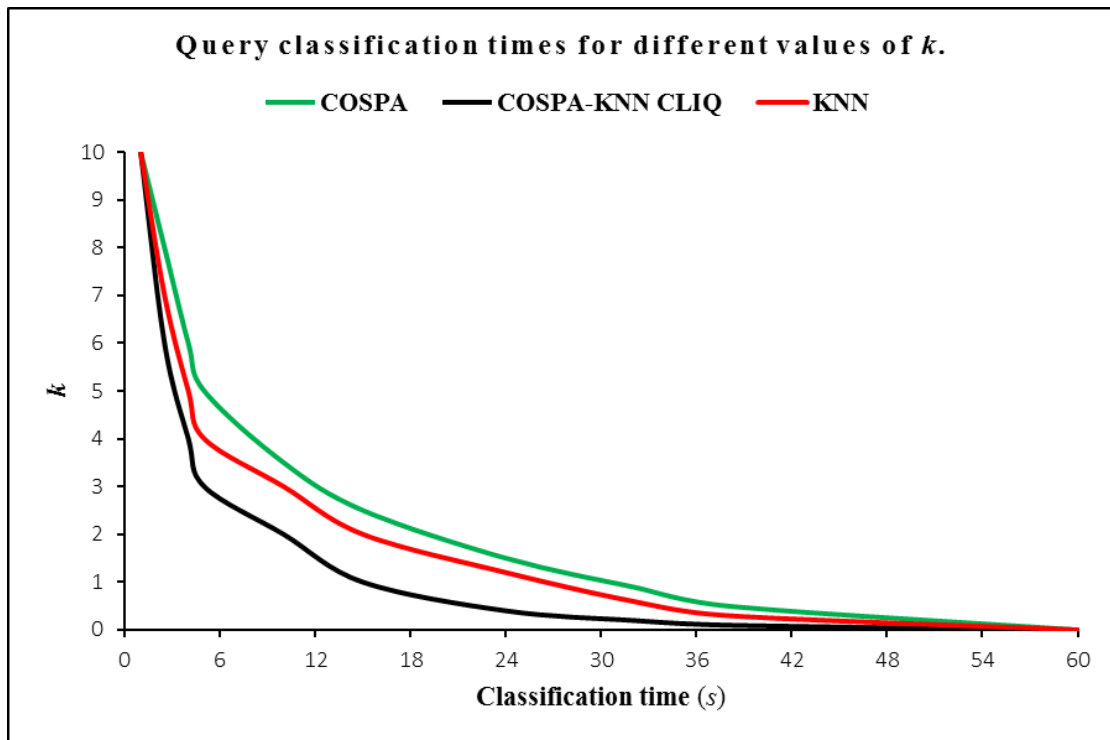
Performance measurements of the implemented communication protocols are evaluated in terms of the transmitted content accuracy and communication time. The implemented controller keeps record of the connection activity, which can be viewed as necessary for systematic statistics. Stored statistics assist in providing opportunities for decisional actions that can be made such as either changing system parameters to make a different observation or testing the system capacity. The accuracy of the system in terms of successful data transmission, since this metric is critical for remote system activities. The ODL controller illustrated efficient service and resource management through software-oriented network activities.





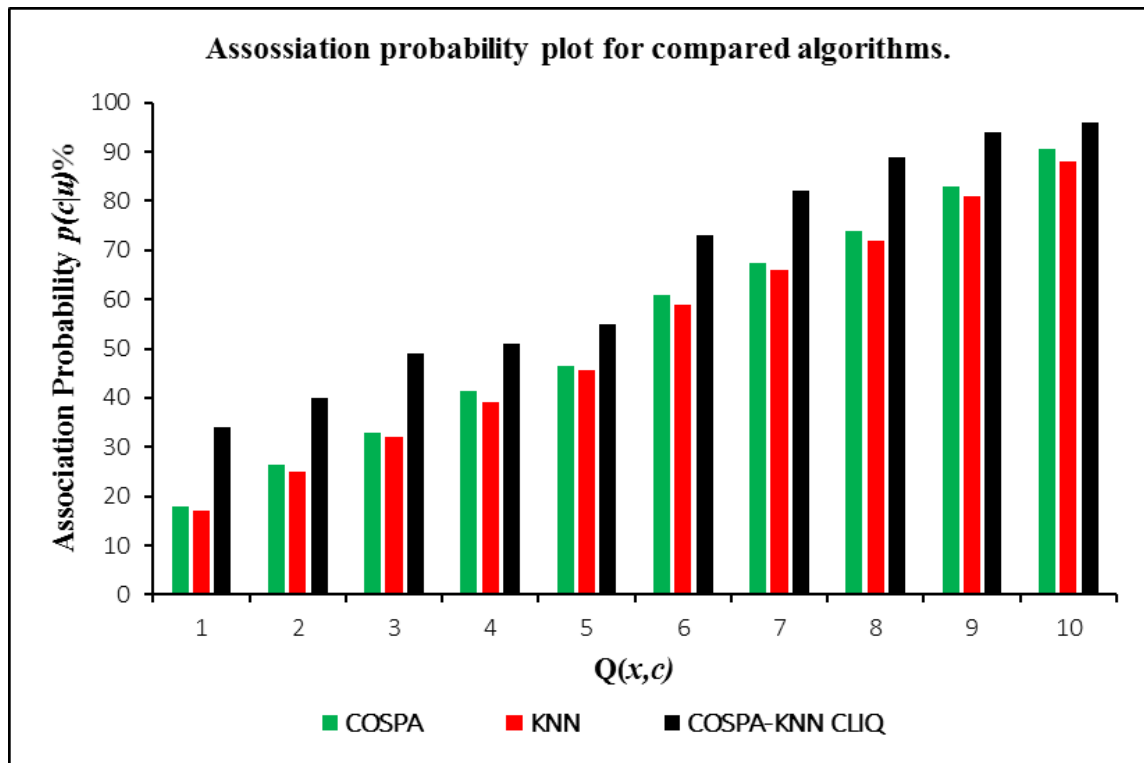
**Figure 4.5:** CDF percentage for different values of k

Figure 4.5, shows a class association CDF graph of the implemented algorithms. A comparison plot of the original KNN, and both the proposed COSPA and COSPA-KNN CLIQ algorithms is provided. The CDF about the value ranges of  $x_r$  and those of  $c_r$  (query points based on their Euclidian ranges) of these algorithms is given in terms of different values of k. For k equals to 1, all the compared methods produce the same cumulative results at 9%. For example, at k values {3, 5, 7 and 9}, the proposed scheme (COSPA-KNN CLIQ coupling) produce the best results. Of interest, at k equals to 9, the proposed scheme produces 98% class cumulative association whereas the original KNN and the proposed COSPA produces 86% and 88% association respectively. What is noteworthy is that, both the proposed COSPA alone and the proposed scheme achieved the best results against the original KNN.



**Figure 4.6:** Classification times for different values of  $k$ .

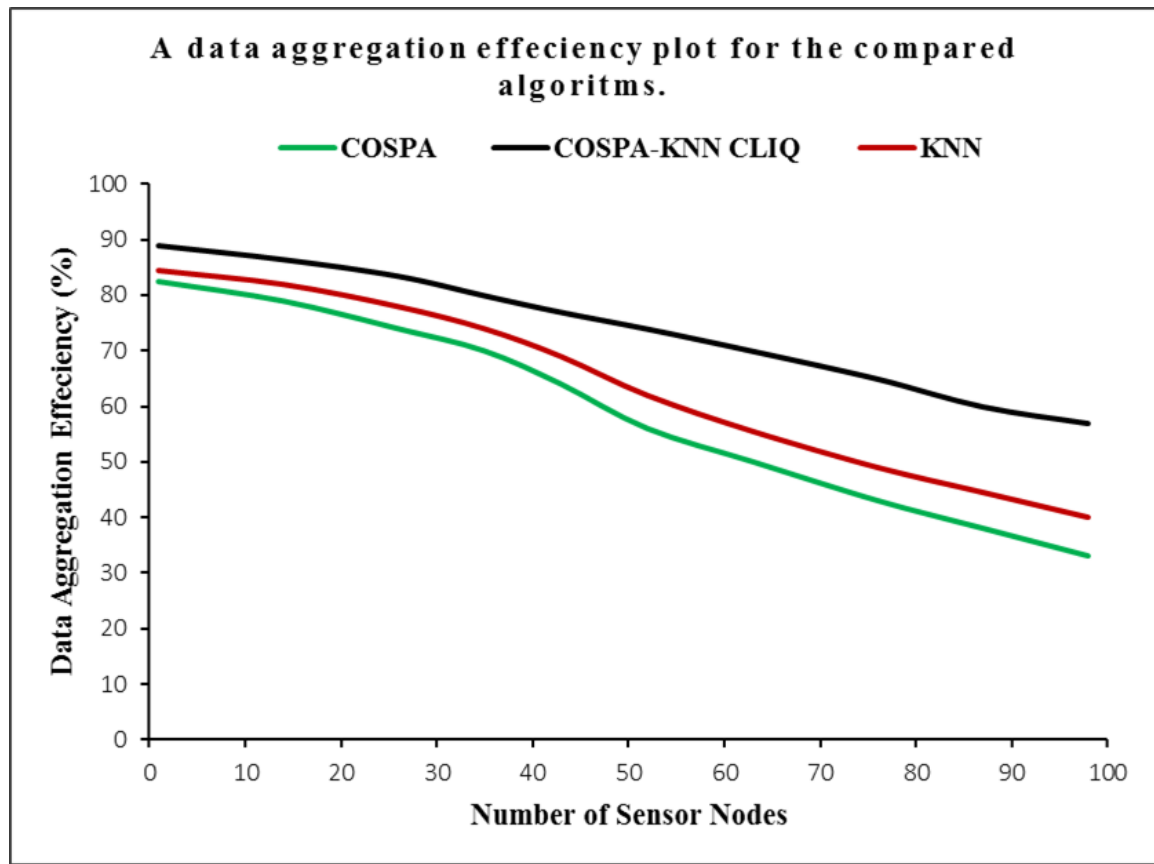
It can be seen from Figure 4.6, that the proposed scheme is even faster compared to the other two algorithms in terms of classification for larger values of  $k$ . On average, the proposed scheme takes between 14 to 30 seconds to perform a query classification, for values of  $k$ , ranging from 1 to 5. For values of  $k$  from 6 and above, the proposed scheme is much faster, with less than 5 seconds at  $k$  equals to 9. However, KNN performs better compared to the proposed COSPA alone. For values of  $k$  equals to 2 to 5, KNN has better performance compared to COSPA. For example, at  $k$  equals to 2, classification time for KNN is just over 18 seconds, whereas COSPA time is just above 23 seconds. This can also be seen at values of  $k$  equals to 3 and 4, where KNN produces better results compared to COSPA. For values of  $k$  at 1, and 6 to 9, KNN performs the same as COSPA.



**Figure 4.7:** Association probability plot for various query attempts on specific regions.

Figure 4.7, illustrates an associative probability,  $p^{\wedge}(c | u)$  in terms of the query process  $Q$  for a range of query points  $(x_r, c_r)$  in a specific region. This plot is generated to quantify the class estimation capacity of all the compared algorithms for different number of query rounds based on specific query points range in a specific region. Performances of the algorithms are discussed below, with analysis on different performance capacities of these schemes in terms of their capabilities for successful classification

As shown in figure 4.7, the proposed scheme (COSPA-KNN CLIQ), indicates the highest query association probability for different query attempts. This is a very significant aspect, since it provides the exhibit as to how efficient is this scheme in terms of correctly classifying query objects given a particular query process. On average, for different query attempt, the proposed scheme produces just over 72% classification probability, whereas, on average, KNN and the proposed COSPA produces just over 50% and 62% respectively. This performance indicates the level of accuracy that could be required for making better decisions.

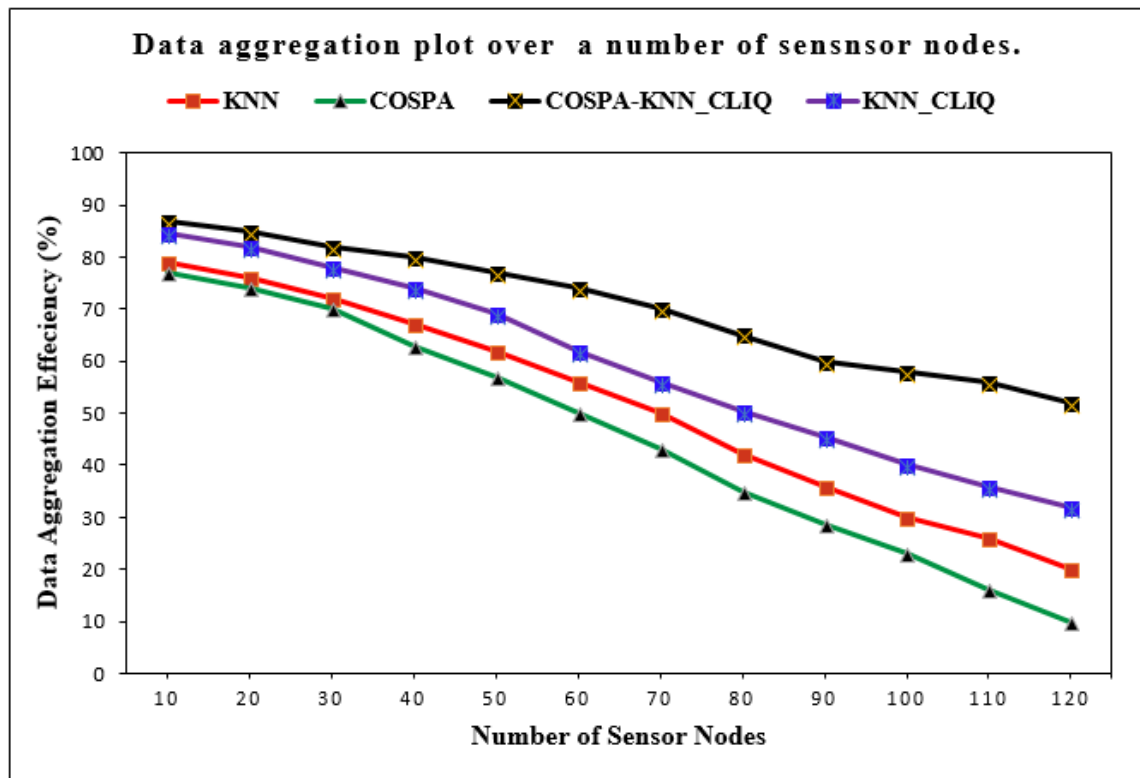


**Figure 4.8:** Data aggregation efficiency plot.

To verify the data aggregation efficiencies of these algorithms, we perform a granular data aggregation efficiency check for specific collected data from sensor nodes. To achieve this, the network simulation time is set to 5 minutes and the granularity aggregation check is performed in 1-minute intervals. Figure 4.8 illustrates data aggregation efficiencies for the compared algorithms. From Figure 4.8, it can be seen that, the proposed scheme produces the best aggregation efficiency. A granularity collection of specific sensor information was performed with mainly the focus on cluster traffic, sensor operational properties and sensor identities. On average when performing an aggregation efficiency on a number of sensor nodes as per cluster, depending on the query process, the proposed scheme produces approximately 88% effectiveness. The original KNN produces about 67% aggregation effectiveness on average, whereas the proposed COSPA alone produces just about 61% aggregation effectiveness on average.

This is a significant performance regarding the proposed scheme, which produces the best results. The proposed scheme has shown best efficiencies for sensor data aggregation depending on query settings that were applied in our experiments. Hence, with this information, best system decisions can be made to improve the overall network performance.

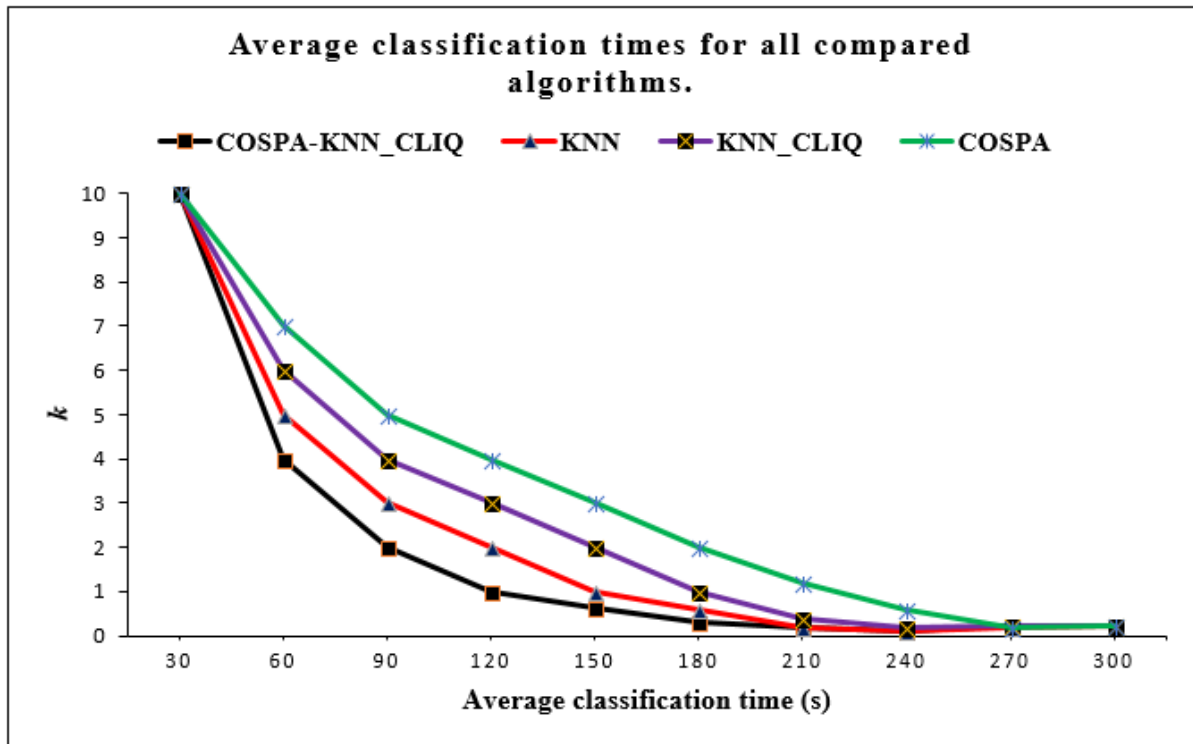
To get the overall performances of all the compared algorithms, an analysis of various performances based on the proposed and implemented algorithms is given in Figures 4.5, 4.6, 4.7 and 4.8. These performance plots provide an understanding as well as proof of the proposed schemes in terms of improving network operations.



**Figure 4.9:** A data aggregation plot for all the compared algorithms.

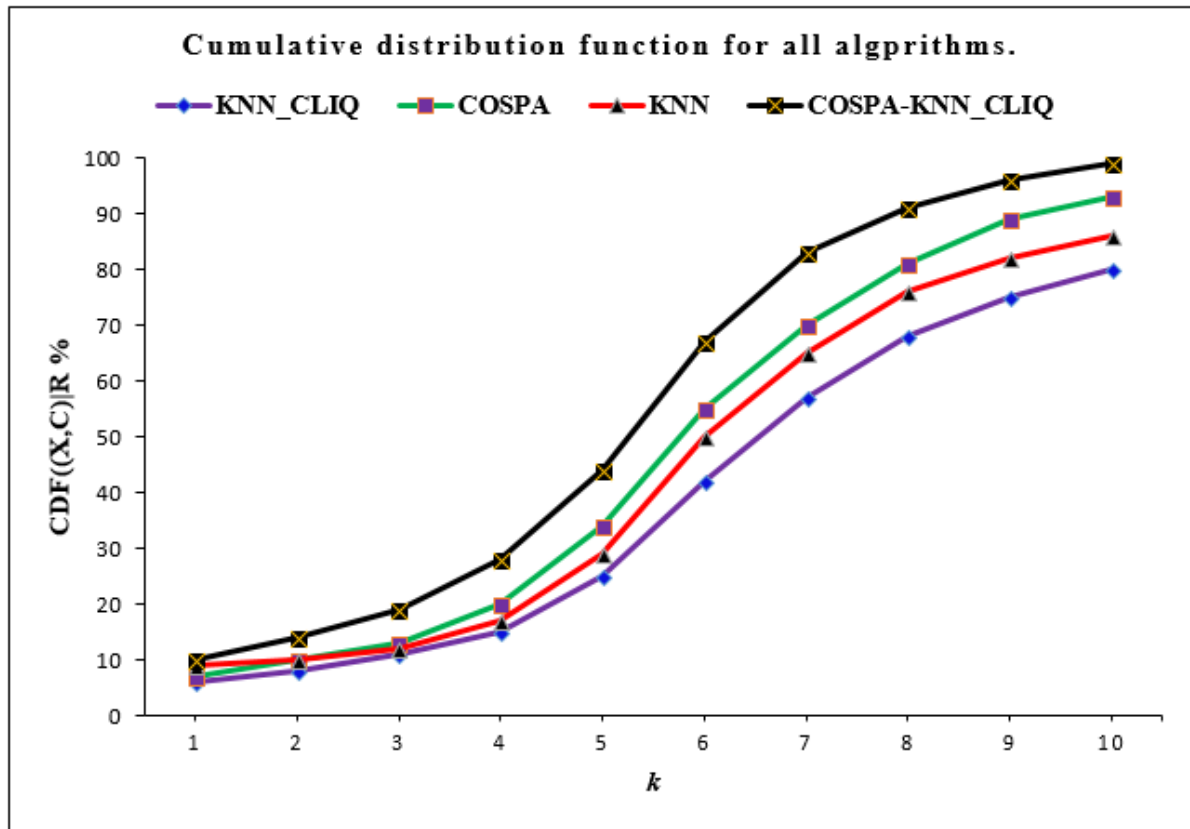
In Figure 4.9, all the proposed and implemented procedures are compared in terms of data aggregation efficiency. It can be seen from the figure above that, the proposed scheme (COSPA-KNN\_CLIQ) achieved the best performance against all the competing procedures. This part of the experiment was performed using hundred and twenty (120) sensor nodes, so as to establish the efficiency of each when the numbers of sensors is increased unlike

previously with hundred (100) sensor nodes. In this case KNN\_CLIQ indicated better performance than both KNN and COSPA. On average, COSPA-KNN\_CLIQ achieved 82% in terms of data aggregation efficiency. KNN\_CLIQ produced 72% of data aggregation efficiency, whereas the original KNN and COSPA algorithms achieved about 62% and 57% respectively.



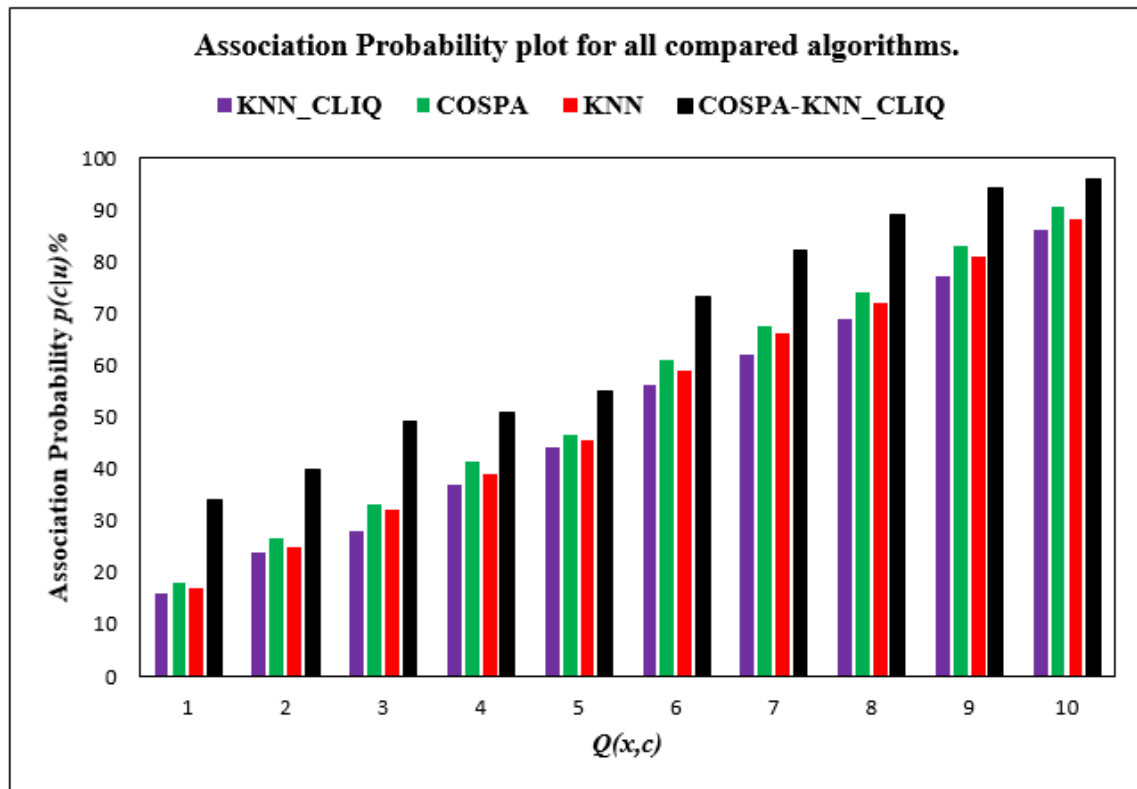
**Figure 4.10:** Average query classification times for all compared algorithms.

Figure 4.10, indicates a performance comparison in terms of the average query classification time (speed) for all the compared procedures. The proposed scheme indicates a fasted classification times on average for various values of  $k$ . For three hundred seconds (5 minutes) of query processing and between values of  $k$  ranging from 3 - 10, COSPA-KNN\_CLIQ takes just about 9.4 seconds to perform a full classification on average. For the same range of  $k$  values, KNN takes just over 11.3 seconds, whereas KNN\_CLIQ and COSPA performs this in just 15 and 18.8 seconds on average respectively. This is a significant achievement by the proposed scheme since it faster classification and as a result the system query processing time in reduced.



**Figure 4.11:** A CDF function plot for all compared algorithms.

As introduced earlier, the CDF provides an association class of a query point based on the observed region or a cluster in terms of sensor nodes. COSPA-KNN\_CLIQ achieved the best results with COSPA performing better in terms of the performed cluster query operation compared to KNN and KNN\_CLIQ. Just as before COSPA-KNN\_CLIQ produces above 95% class association success, with COSPA just above 86% and KNN at 74%. KNN\_CLIQ indicated the least performance at 70% which is within the same performance range with KNN algorithm.



**Figure 4.12:** Association probability of all compared algorithm.

A query probability is also performed to evaluate all these algorithms in terms of their capacity to classify query data points on targeted regions. Based on the conducted experiment, the proposed scheme, provides the highest probability estimation compared to all the other algorithms. In this case COSPA performed better than both KNN and KNN\_CLIQ in terms of the class association. This plot also verifies the performance recorded using the CDF function for categorizing query objects in a specific region. With the applied parameters based on the system model, the candidate is confident in saying that the proposed scheme produces efficient results with opportunities to achieve full control of the network infrastructure.

#### 4.5 CHAPTER CONCLUSION

In this work, a controller-based open status presentation algorithm (COSPA) was proposed with a focus to enhance network knowledge for SDWSN monitoring systems. To advance



its data aggregation capacity, another machine learning technique- KNN CLIQ, was also proposed. The coupling of these two algorithms form our proposed scheme. Performance comparisons of our system were performed under different network activities. The proposed scheme has exhibited best performance on all activity test that were performed in this work.

Greater aggregation efficiencies have been realized through the proposed scheme. It is also noteworthy to acknowledge that, based on statistical analysis and observations, the original KNN has produced better results compared to the proposed COSPA alone with regards to data aggregation efficiency and query classification time. Through this work, a better sensor aggregation information has been achieved through granularity operations performed on the network. This has significantly enhanced the capacity of our system in terms of knowledge. Machine learning techniques were successfully integrated to SDWSN programming strategies to achieve efficient sensor data aggregation. Thus, our system indicates rich capabilities for SDWSN monitoring application systems. To deal with transmission throughput of the changing cluster formations due to sensor data transmission or broadcasting, the implemented approach maintains networking status and sensitive cooperative data so that intended traffic packets reach their destinations. This is also a great contribution towards efficient QoS implementation to achieve flexible and effective network management.

Efficient data aggregation within the network is critical, especially between sink nodes and sensor clusters as this promotes data integrity and network reliability. Thus, flexible routing mechanisms can improve the entire network communication between the cluster-to-sink and that between the sink nodes and the global controller. This will also promote the capacity of the network to advance intelligent methods to know of the status of the network. SDN can enable different or new roles to be provided or installed to sensor nodes depending on their capacities and resources so as to make the network more intelligent and robust. WSN technologies operate efficiently based on their topologies, therefore, clustering techniques must be well maintained to avoid weakening the overall network performance.

## CHAPTER 5

## DISCUSSION

This work has produced commendable results, and from which, a lot of technical analysis can be made. To come to these discussions, it is noteworthy to provide a description of the proposed approach and setup as these have largely informed the type of achieved results. Afterwards, provide details of all the proposed and developed strategies, which forms the main contribution of this work, in terms of how they were applied to the implemented system. This work proposed an SDN strategy to WSNs in particular for wireless monitoring application systems, with OpenFlow as its main communication protocol. The use WSNs for monitoring has gained popularity given their simplicity to implement and deploy. Even though many of such systems benefit from the use of wireless sensors, their systems capacity is limited for high computing application. As a result, a lot of developmental approaches and proposals tries to implement strategies that will enhance the processing capacity of these technologies.

This work further proposed machine learning techniques as an effort to improve sensor data aggregation and network intelligence by means of efficient query processing and information presentation. This position has motivated this study, to propose, develop flexible SDN strategies coupled with machine learning techniques to improve application potential of WSN monitoring application systems. A machine learning algorithm- COSPA is then developed to enhance the implemented SDWSN system by means of improving its computing capacity in terms of resource management and knowledge presentation. COSPA, coupled with a variation of the original KNN algorithm- KNN-CLIQ whose functionality is to enhance cluster query processing and also improve sensor data aggregation.

To evaluate the proposed and implemented SDN controller, performance comparisons of three different controllers can be observed in Figure 3.12. The three SDN controllers- OpenDayLight, Ryu and NOX, were evaluated based on the capacity to support high sensor data throughput with efficient data aggregation time. The study makes a point that, efficient data aggregation time exhibits efficient data handling. In addition, a support for higher throughput implies better system knowledge basis and as a result, reliable data can be achieved at the control terminal. Hence, efficient network knowledge will enable administrators to make in time and sound decisions to maintain the system. The proposed and implemented SDN controller (ODL Controller) produced best results in terms of data aggregation time and sensor throughput, with Ryu performing better than the NOX controller. This performance achievement puts the ODL controller as an efficient SDN controller in terms of data handling and processing time. Based on the conducted experiments, the ODL controller also has indicated efficient support for implementing customized top-layer network applications than the other two competing SDN controllers implemented in this work. This capability by the ODL controller, suggested it as a better controller to support WSN application system innovation especially for developing top-layer applications that advances simplified management of network resources.

Efficient resource management is critical for wireless monitoring application systems, especially those that are deployed for life concerning purposes. To achieve efficient resource sharing and the implementation of flexible network applications or service requires competent communication or traffic channelling protocols. This work proposed and implemented OpenFlow communication protocol as the southbound interface between the controller and the underlying data plane.

Performance results of the proposed SDWSN-OF compared with both SDWSN-BGP and SDWSN-NETCONF can be seen from figures 3.12 to 3.17. Performance experiments were conducted in terms of packet error rate, sensor energy consumption, throughput computation and communication end-to-end delay. In all these performance metrics, the proposed approach produced best results, with SDWSN-NETCONF performing better than SDWSN-BGP. To further evaluate the efficiency of the proposed communication approach, enhanced variations of both SDWSN-NETCONF and SDWSN-BGP were implemented and compared

with the proposed one. These performance results can be seen particularly in figure 3.12. In this case, the three approaches were compared in terms of sensor energy consumption for different data packet sizes. The proposed approach produced commendable results with regard to sensor energy utilization. This achievement suggested the proposed approach as an optimum approach to be implemented for energy-constrained sensor application systems.

This work further proposed machine learning methods to introduce some level of intelligence on SDNWS systems, in particular for monitoring applications. The original KNN was adapted to develop KNN-CLIQ, which is a variation of the former. COSPA is proposed and developed for providing this intelligence in terms of building efficient network knowledge. KNN-CLIQ was developed to improve data aggregation, as it is critical for achieving enough sensor data at the control layer. COSPA is then coupled with KNN-CLIQ to form a proficient strategy for improving network intelligence and routing capacity. Performance results based on conducted experiments can be seen in Figures 4.5 to 4.8.

Conducted experiments were performed based in sensor cluster query point probability, classification times for different values of  $k$  and data aggregation efficiencies for the compared algorithms. Based on these experiments, the proposed strategy produced best results compared to other competing algorithms. In particular, Figure 4.8 illustrates aggregation competences of all the compared methods for a different numbers of sensor nodes. It can be seen from this plot, that, for a higher number of sensor node, the proposed strategy produced better data aggregation performances. This is a significant achievement by the proposed approach since it reflects the desired potential for the implemented application system.

To quantify the class estimation capability of all the compared algorithms for different numbers of query rounds based on specific query points range in a particular region, an associated probability is performed. This can be seen in figure 4.7, wherein the proposed strategy produces the best results. As with other experiments, it can also be observed in this figure that, COSPA alone also provides better results compared to the original KNN algorithm. This is also a significant achievement from the proposed strategies, as the exhibit commendable performances. Query classification time performance comparisons for these

strategies can also be seen in figure 4.6. For different values of  $k$ , COSPA-KNN CLIQ produced best results with KNN producing better results than COSPA.

It is also critical to know the status of different deployed sensor nodes since these devices contribute to the formation of the larger system or network. To achieve this, requires intelligent techniques to query the network efficiently. Based on the proposed approach; this was realized through the coupled processing functionality of the coupled KNN-CLIQ and COSPA. Together, these algorithms form an intelligent resource that performs efficient cluster querying and information retrieval. This allows the realization of efficient network knowledge, which is critical to system administrators for making informed network decisions. The COSPA's information coalition functionality with the KNN-CLIQ ensures that accurate information is presented to the ODL controller for efficient network management.

## CHAPTER 6

## CONCLUSION

SDN programmable strategies has a potential to improve how WSN application systems perform and are implemented. These strategies have the capacity to evolve monitoring capabilities of WSN technologies. WSN application systems can greatly benefit from SDN strategies in terms of; resource management, network service automation, data transmission, knowledge presentation, query processing, network security and network configuration flexibility. Even though their integration to WSNs has not gained some popularity, machine learning techniques also possess the capacity to improve WSN application systems. WSN systems can benefit from machine learning methods through automated network services, data querying and network statistics. Given their technicality, both SDN and machine learning methods hold the potential to improve the computing capacity of WSN application technologies. A carefully planned strategy is needed to integrate these computing methodologies and apply them to WSN systems to advance and improve their computing nature. However, for WSN application systems to realize these aspired capabilities, a lot of work still needs to be done.

This work has developed and implemented strategies to improve WSN application systems for the mention application area. The candidate can say in confidence that, this work has developed a foundation as well as a direction for efficient WSN monitoring application systems. This work draws technological visions from the fact that, efficient computing network policies and strategies must possess the ability to; support the growth of the network, flexibility and be easily manageable. Hence, the proposed approach was essentially motivated by these technological aspirations.

Technical achievements from the surveyed related work has also contributed to the developmental strategies implemented in this work. In Chapter 3, SDWSN programmable strategies have been devised to provide efficient network management in terms of improved communication, simplified resource management, efficient network querying, data aggregation and information presentation. Efficient system communication has been realized from the use of OpenFlow since this protocol fully supports SDN systems. This has assisted in developing network applications that supports service flexibility and service abstractions for resource-demanding application services. Through SDN programmable strategies, developed service abstractions can be initiated from the ODL controller for specific network demands. To verify the efficiency of OpenFlow, performance comparisons with other technologies as seen in Figures 3.8 to 3.13. It has been shown that for SDN applications, when compared to both SDN variations of BGP and NETCONF, OpenFlow produced lower packet error rate. This is a significant achievement as it implies that, OpenFlow promotes connection integrity and networking reliability. NETCONF performed better compared to BGP in terms of packet error rate. This performance by NETCONF indicates that it is also suitable for SDN applications.

Based on the conducted experiments it has also been observed that for different network operations NETCONF produces faster configuration speed compared to both OpenFlow and BGP. However, BGP indicated some efficiency in terms of traffic routing between distributed base stations. OpenFlow fully supports SDN programmability and controller communications to the underlying network devices. Given their deferent potential, these communication protocols can be implemented together to achieve different goals especially for large application networks. The ODL controller supports programmability and network services abstraction. This is a critical aspect of the OpenDayLight's computing functionality as it promotes SDN strategies. The ODL also indicated best performances in terms of network view and top-layer applications support compared to the other two SDN controllers. This is a significant achievement by the implemented SDN controller since it forms part of the proposed approach.

Applications of machine learning techniques to SDWSNs is fairly new and most work is on proposals and surveys. This work has proposed and developed machine-learning methods to enhance WSN systems. These methods have been successfully coupled to the developed SDN strategies to improve the computing capacity of the proposed SDWSN application system. Figures 4.5 to 4.8 indicate technical achievements based on machine learning methods coupled with SDN strategies. The coupled strategies forms COSPA-KNN CLIQ, which is a scheme that operates intelligently to increase network visibility and data aggregation. Experimental results indicate commendable systematic achievement in terms of the overall performance of the system. The proposed strategy has significantly evolved how WSN systems can be implemented and deployed for mission critical systems.

As with SDN, machine learning brings functional intelligence to the network by means of improving traffic routing and data processing. Hence, coupling SDN and machine learning computing methods, provides rich opportunities to innovate SDWSN networks. Developed strategies have commendably addressed most of the WSN application system challenges discussed in this work. Computing aspects such as; path computation for efficient routing, efficient data aggregation for improved throughput handling, fast and proficient query processing, accurate information presentation, increased network flexibility and visibility as well as intelligent resource management, were commendably improved. Through the improvement of these computing aspects, the overall system performance was enhanced. The candidate is confident to say that, the proposed approach holds systematic proficiencies to be adopted as an architecture for compute intensive and sensitive wireless monitoring application systems. The proposed approach displayed efficiency in terms of sensor energy consumption. This is a very significant achievement since this aspect of the system directly contribute towards the prolonging an SDWSN system lifetime. This study has produced research articles in terms of a review paper and technical papers. The carried out study with all systematic developments and implementations has led to this thesis document.



## 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]. 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.
- [3]. 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.
- [4]. 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.
- [5]. I. El 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.
- [6]. 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.

## REFERENCES

---

- [7]. A. Nanda, A. K. Rath and S. K. Rout. "Node Sensing & Dynamic Routes for Wireless Sensor Networks." *International Journal of Computer Science and Information Security (IJCSIS)*, Vol. 7, No.3, pp. 122-131 Mar. 2010.
- [8]. G.S. Vadivel, K. Mundada and P. Kanjalkar. "Failsafe Wireless Sensor Network Using Cooperative Communication," *In Proc. International Conference on Advances in Electrical Engineering (ICAEE)*, VIT University, Vellore, Tamilnadu, India, pp.1-4, Jan., 2014.
- [9]. 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.
- [10]. J. Dhananandhini, C. M. Niranjana, K. Rajeswari and R. Karthick. "Detecting Sensor Node Failure and Node Scheduling Scheme for WSNs". *International Journal of Engineering Research and Development*, Vol. 9, Issue 2, pp. 53-56, Nov. 2013.
- [11]. ITU-T (Telecommunication Standardization Sector of ITU), "Applications of Wireless Sensor Networks in Next Generation Networks," Series Y.2000: Next Generation Networks, pp. 2-94, 28 Feb. 2014.
- [12]. E. Nigussie, A. Hakkala, S. Virtanen and J. Isoaho, "Energy-aware Adaptive Security Management for Wireless Sensor Networks," *In Proc. A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on*. Sydney, NSW, pp. 1-4, Jun. 2014.
- [13]. M. Kabiri and J. Vahidi. "Analysis of Performance Improvement in Wireless Sensor Networks Based on Heuristic Algorithms Along with Soft Computing Approach". *Journal of mathematics and computer Science*, vol. 13, pp. 47-67, Sept. 2014.
- [14]. J. Ko, C. Lu, M. B. Srivastava, J. A. Stankovic, A. Terzis, and M. Welsh. "Wireless Sensor Networks for Healthcare". *In Proc. of the IEEE*, Vol. 98, No. 11, pp. 1947-1960, Nov. 2010.
- [15]. 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 7<sup>th</sup> International Symposium on Telecommunications (IST'2014)*. Tehran, Iran, pp. 1138-1143, Sept., 2014.

## REFERENCES

---

- [16]. 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.
- [17]. 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.
- [18]. G. Liu, X. Zhang and Y. Liu, "Distributed estimation based on game theory in energy harvesting wireless sensor networks", *In Proc. Control Conference (CCC), 2014 33rd Chinese*, Nanjing, pp. 401 – 404, 2014.
- [19]. J. Jessen, M. Venzke, and V. Turau, "Design considerations for a universal smart energy module for energy harvesting in wireless sensor networks", *In Proc. of the Ninth Workshop in Intelligent Solutions in Embedded Systems (WISES), 2011*, Regensburg, Germany, pp. 35 – 40, 2011.
- [20]. S. Jiaying and K.T. Yen, "Energy consumption analysis of ZigBee-based energy harvesting wireless sensor networks", *In Proc. Communication Systems (ICCS), 2012 IEEE International Conference*, Singapore, pp. 468 – 472, 2012.
- [21]. A. Cammarano, C. Petrioli, and D. Spenza, "Pro-Energy: A novel energy prediction model for solar and wind energy-harvesting wireless sensor networks", *In Proc. 2012 IEEE 9th International Conference on Mobile Adhoc and Sensor Systems (MASS)* , Las Vegas, NV, pp. 75 – 83, 2012.
- [22]. N. A. Cloete, R. Malekian and L. Nair, "Design of Smart Sensors for Real-Time Water Quality Monitoring," in *IEEE Access*, vol. 4, pp. 3975-3990, 2016. doi: 10.1109/ACCESS.2016.2592958
- [23]. A. C. Jose and R. Malekian, "Improving Smart Home Security: Integrating Logical Sensing Into Smart Home," in *IEEE Sensors Journal*, vol. 17, no. 13, pp. 4269-4286, July1, 1 2017. doi: 10.1109/JSEN.2017.2705045
- [24]. V. Shakhov, "Tradeoff between energy harvesting duration and reliability in Wireless Sensor Networks", *Telecommunications Forum Telfor (TELFOR), 2014 22<sup>nd</sup>*, Belgrade, pp. 292 – 295, 2014.

## REFERENCES

---

- [25]. S.M. Reda, M. Abdelhamid, O. Latifa and A. Amar, Efficient uncertainty-aware deployment algorithms for wireless sensor networks, *In Proc. Wireless Communications and Networking Conference (WCNC)*, Shanghai, China, pp. 2163-2167, Apr. 2012.
- [26]. Z. Bojkovic and B. Bakmaz, "A Survey on Wireless Sensor Networks Deployment," *WSEAS Transactions on Communications*, Vol. 7, Issue. 12, pp. 1172-1181, Dec. 2008.
- [27]. T. Stoyanova, F. Kerasiotis and G. Papadopoulos, "Deployment Considerations for Reliable Communication in Wireless Sensor Networks," *International Journal on Advances in Networks and Services (IARIA)*, Vol. 3 No. 1&2, pp. 158-169, 2010.
- [28]. Z. Ghaffari, T. Jafari and H. E. Shahraki, "Comparison and Analysis Data-Centric Routing Protocols in Wireless Sensor Networks," 2013 International Conference on Communication Systems and Network Technologies, Gwalior, 2013, pp. 351-355. doi: 10.1109/CSNT.2013.80
- [29]. J. Grover, Shikha and M. Sharma, "Location based protocols in Wireless Sensor Network — A review," Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT), Hefei, 2014, pp. 1-5. doi: 10.1109/ICCCNT.2014.6962990
- [30]. A. A. Anasane and R. A. Satao, "A Survey on Various Multipath Routing Protocols in Wireless Sensor Networks," *Procedia Computer Science*, Vol. 79, pp 610-615, 2016, <https://doi.org/10.1016/j.procs.2016.03.077>.
- [31]. K. K. Krishna and R. Augustine "A Survey on Mobility based Routing Protocols in Wireless Sensor Networks," *International Journal of Computer Applications*, Vol. 135, No.5, pp. 36-38, 2016.
- [32]. M. Jan and M. Khan, "A Survey of Cluster-based Hierarchical Routing Protocols," *International Journal of Computer Networks and Wireless Communications (IJCNWC)*, Vol. 3, no. 2, pp. 138-143, 2013.
- [33]. R. Sumathi and M. G. Srinivas, "A survey of QoS based routing protocols for wireless sensor networks," *Journal of Information Processing Systems*, Vol. 8, No. 4, pp. 589–602, 2012.

## REFERENCES

---

- [34]. S. Tanwar, N. Kumar and J.P.C. Rodrigues, "A systematic review on heterogeneous routing protocols for wireless sensor network," *Journal of Network and Computer Applications*, Vol. 53, pp 39-56, 2015.
- [35]. R. Anane, R. Bouallegue and K.Raouf (2016), "Medium Access Control (MAC) Protocols for Wireless Sensor Network: An Energy Aware Survey," In: El Oualkadi A., Choubani F., El Moussati A. (eds) *Proceedings of the Mediterranean Conference on Information & Communication Technologies 2015*. Lecture Notes in Electrical Engineering, Vol 380. Springer, Cham.
- [36]. A. Braman and G. R. Umapathi, "A comparative study on advances in leach routing protocol for wireless sensor networks: A survey," *International Journal of Advanced Research in Computer and Communication Engineering*, Vol. 3, No. 2, pp. 5683–5690, 2014.
- [37]. I. D. Chakeres and E. M. Belding-Royer, "AODV routing protocol implementation design," *24th International Conference on Distributed Computing Systems Workshops, 2004 Proceedings*, pp. 698-703, 2004. doi: 10.1109/ICDCSW.2004.1284108
- [38]. Geetu and S. Juneja, "Performance Analysis of SPIN and LEACH Routing Protocol in WSN," *International Journal of Computational Engineering Research* Vol. 2, No. 5, 2012.
- [39]. Rama Sundari Battula and O. S. Khanna, "Geographic Routing Protocols for Wireless Sensor Networks: A Review," *International Journal of Engineering and Innovative Technology (IJEIT)*, Volume 2, Issue 12, pp. 39-42, June 2013.
- [40]. G. Scheepers, R. Malekian, D. C. Bogatinoska and B. R. Stojkoska, "A low-power cost-effective flexible solar panel powered device for wireless livestock tracking," *2017 25th Telecommunication Forum (TELFOR)*, Belgrade, 2017, pp. 1-4. doi: 10.1109/TELFOR.2017.8249349
- [41]. V. Henriques, R. Malekian and D. Capeska Bogatinoska, "Mine safety system using wireless sensor networks," *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, 2017, pp. 515-520. doi: 10.23919/MIPRO.2017.7973480

## REFERENCES

---

- [42]. H. Huang, T. Gong, P. Chen, R. Malekian and T. Chen, "Secure two-party distance computation protocol based on privacy homomorphism and scalar product in wireless sensor networks," in *Tsinghua Science and Technology*, vol. 21, no. 4, pp. 385-396, Aug. 2016. doi: 10.1109/TST.2016.7536716
- [43]. K.M. Modieginyane et al., "Software Defined Wireless Sensor Networks Application Opportunities for Efficient Network Management: A Survey," *Computers and Electrical Engineering* (2017), <http://dx.doi.org/10.1016/j.compeleceng.2017.02.026>
- [44]. R. Jain, "introduction to Software Defined Networking (SDN)," available at: <http://www.cse.wustl.edu/~jain/cse570-13/>, (2013).
- [45]. 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
- [46]. N. McKeown, T. Anderson, H. Balakrishnan, G. M. Parulkar, L. L. Peterson, J. Rexford, S. Shenker, and J. S. Turner, "OpenFlow: Enabling Innovation In Campus Networks," *Computer Communication Review*, Vol. 38, No. 2, pp. 69–74, 2008.
- [47]. M. Elkhodr, S. Shahrestani and H. Cheung, "The Internet of Things: New Interoperability, Management and Security Challenges," *International Journal of Network Security & Its Applications (IJNSA)*, Vol. 8, No. 2, March 2016.
- [48]. D. C. Bogatinoska, R. Malekian, J. Trengoska and W. A. Nyako, "Advanced sensing and internet of things in smart cities," 2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, 2016, pp. 632-637. doi: 10.1109/MIPRO.2016.7522218
- [49]. J. Wannenburg and R. Malekian, "Physical Activity Recognition from Smartphone Accelerometer Data for User Context Awareness Sensing," in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 12, pp. 3142-3149, Dec. 2017. doi: 10.1109/TSMC.2016.2562509

## REFERENCES

---

- [50]. J. Wannenburg and R. Malekian, "Body Sensor Network for Mobile Health Monitoring, a Diagnosis and Anticipating System," in *IEEE Sensors Journal*, vol. 15, no. 12, pp. 6839-6852, Dec. 2015. doi: 10.1109/JSEN.2015.2464773
- [51]. J. Wannenburg, R. Malekian and G. P. Hancke, "Wireless Capacitive Based ECG Sensing for Feature Extraction and Mobile Health Monitoring," in *IEEE Sensors Journal*. doi: 10.1109/JSEN.2018.2844122
- [52]. I. Bedhief, M. Kassar and T. Aguilu, "SDN-Based Architecture Challenging The IoT Heterogeneity," *2016 3rd Smart Cloud Networks & Systems (SCNS)*, Dubai, 2016, pp. 1-3. doi: 10.1109/SCNS.2016.7870558
- [53]. R. Morabito, "A Performance Evaluation of Container Technologies on Internet of Things Devices," *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPs)*, San Francisco, CA, 2016, pp. 999-1000. Doi: 10.1109/INFCOMW.2016.7562228
- [54]. A. Hakiri and A. Gokhale, "Rethinking the Design of LR-WPAN IoT Systems with Software-Defined Networking." *2016 International Conference on Distributed Computing in Sensor Systems*, Washington, DC, pp. 238-243, 26-28 May 2016.
- [55]. 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
- [56]. Z. Qin, G. Denker, C. Giannelli, P. Bellavista and N. Venkatasubramanian, "A Software Defined Networking architecture for the Internet-of-Things," *2014 IEEE Network Operations and Management Symposium (NOMS)*, Krakow, 2014, pp. 1-9. Doi: 10.1109/NOMS.2014.6838365
- [57]. L. Yang *et al.*, "System-level design solutions: Enabling the IoT explosion," *2015 IEEE 11th International Conference on ASIC (ASICON)*, Chengdu, 2015, pp. 1-4. Doi: 10.1109/ASICON.2015.7517023
- [58]. A. M. I. Alkuhlani and S.B. Thorat, "Internet of Things (IOT) Standards, Protocols and Security Issues," *International Journal of Advanced Research in Computer and*

## REFERENCES

---

- Communication Engineering Vol. 4, No. 11, November 2015, pp. 491-495. Doi: 10.17148/IJARCCCE.2015.411109 491
- [59]. K. Xu, Y. Qu and K. Yang, "A Tutorial On The Internet Of Things: From A Heterogeneous Network Integration Perspective," in *IEEE Network*, vol. 30, no. 2, pp. 102-108, March-April 2016. Doi: 10.1109/MNET.2016.7437031
- [60]. M. I. Hussain, "Internet of Things: Challenges and Research Opportunities," *CSI Transactions on ICT (CSIT 2017)*, Vol. 5, No. 1, pp. 87-95. Doi.org/10.1007/s40012-016-0136-6
- [61]. S. K. Tayyaba, M. A. Shah, O. A. Khan, and A. W. Ahmed. 2017, "Software Defined Network (SDN) Based Internet of Things (IoT): A Road Ahead," In *Proceedings of the International Conference on Future Networks and Distributed Systems (ICFNDS '17)*. ACM, New York, USA, Article 15, pp. 1-8. doi: <https://doi.org/10.1145/3102304.3102319>
- [62]. D. Bendouda, A. Rachedi and H. Haffaf, "A Hybrid and Proactive Architecture Based on SDN For Internet of Things," *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, Valencia, 2017, pp. 951-956. doi: 10.1109/IWCMC.2017.7986414
- [63]. A. Kliem and O. Kao, "The Internet of Things Resource Management Challenge," *2015 IEEE International Conference on Data Science and Data Intensive Systems*, Sydney, NSW, 2015, pp. 483-490. Doi: 10.1109/DSDIS.2015.21
- [64]. Z. Zhang, Z. Zhang, R. Wang, Z. Jia, H. Lei and X. Cai, "ESD-WSN: An Efficient SDN-Based Wireless Sensor Network Architecture for IoT Applications," In: Ibrahim S., Choo KK., Yan Z., Pedrycz W. (eds) *Algorithms and Architectures for Parallel Processing. ICA3PP 2017. Lecture Notes in Computer Science*, Vol. 10393. Springer, Cham.
- [65]. 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>.
- [66]. K.M. Modieginyane, R. Malekian and B.B. Letswamotse, "Flexible Network Management and Application Service Adaptability in Software Defined Wireless



## REFERENCES

---

- 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>
- [67]. D. Zeng, T. Miyazaki, S. Guo, T. Tsukahara, J. Kitamichi and T. Hayashi, “Evolution of Software-Defined Sensor Networks,” In Proc. IEEE 9th International Conference on Mobile Ad-hoc and Sensor Networks (MSN), Dalian, pp.410-413, Dec., 2013.
- [68]. A.S. Yuan, H. Fang and Q. Wu, “OpenFlow based hybrid routing in Wireless Sensor Networks,” In Proc. IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Singapore, pp. 1-5, Apr., 2014.
- [69]. Y. Mao, L. Yong, J. Depeng, S. Li and Z. Lieguang, “Opportunistic spectrum sharing in software defined wireless network”, IEEE, *Journal of Systems Engineering and Electronics*, Vol. 25, No. 6, pp. 934 – 941, Dec . 2014.
- [70]. 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, November 26, 2012.
- [71]. R. Sayyed, S. Kundu, C. Warty and S. Nema, "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, 2014.
- [72]. A. De Gante, M. Aslan and A. Matrawy, “Smart Wireless Sensor Network Management Based on Software-Defined Networking”, 27th Biennial symposium on Communications (QBSC), Kingston, ON, pg. 71-75, 1-4 June 2014.
- [73]. 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.
- [74]. 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.
- [75]. P. Jayashreel and F. Infant Princy, "Leveraging SDN to Conserve Energy in WSN- An Analysis." In: 2015 3rd International Conference on Signal Processing,

## REFERENCES

---

- Communication and Networking (ICSCN), Chennai, India, pp.1-6 , 26 - 28 Mar 2015.
- [76]. R. Huang, X. Chu, J. Zhang and Yu Hen Hu, "Energy-Efficient Monitoring in Software Defined Wireless Sensor Networks Using Reinforcement Learning: A Prototype", International Journal of Distributed Sensor Networks, Volume 2015, Article ID 360428, <http://dx.doi.org/10.1155/2015/360428>.
- [77]. 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
- [78]. 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.
- [79]. 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
- [80]. 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
- [81]. 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.
- [82]. 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

## REFERENCES

---

- Personal, Indoor, and Mobile Radio Communications (PIMRC), Valencia, 2016, pp. 1-6. doi: 10.1109/PIMRC.2016.7794902
- [83]. 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.
- [84]. 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>
- [85]. T. Miyazaki, "Dynamic Function Alternation to Realize Robust Wireless Sensor Network," *International Journal of Handheld Computing Research (IJHCR)*, Vol. 3, No. 3, pp. 17–34, Jul. 2012.
- [86]. 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
- [87]. S. Misra, S. Bera, M. P. Achuthananda, S. K. Pal and M. S. Obaidat, "Situation-Aware Protocol Switching in Software-Defined Wireless Sensor Network Systems." *IEEE Systems Journal*, vol. PP, no. 99, pp. 1-8. doi: 10.1109/JSYST.2017.2774284
- [88]. L. Galluccio, S. Milardo, G. Morabito and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIRELESS SENSOR networks." In: 2015 IEEE Conference on Computer Communications (INFOCOM), Hong Kong, pp. 513-521, 26 Apr. - 1 May, 2015.
- [89]. M. Li, J. Li and L. Shen, "Joint Routing and Resource Control in Software-Defined Sensor Networks." 2015 International Conference on Wireless Communications & Signal Processing (WCSP), Nanjing, pp. 1-5, 15-17 Oct. 2015.
- [90]. 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

## REFERENCES

---

- [91]. J. Wang, P. Zhai, Y. Zhang, L. Shi, G. Wu, X. Shi and P. Zhou (2018) "Software Defined Network Routing in Wireless Sensor Network". In: Wan J. et al. (Eds) Cloud Computing, Security, Privacy in New Computing Environments. CloudComp 2016, SPNCE 2016. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 197. Springer, Cham
- [92]. 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
- [93]. 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.
- [94]. B. T. de Oliveira, C. B. Margi, "TinySDN: Enabling TinyOS to Software-Defined Wireless Sensor Networks", [http://sbrc2016.ufba.br/downloads/Salao\\_Ferramentas/154318.pdf](http://sbrc2016.ufba.br/downloads/Salao_Ferramentas/154318.pdf)
- [95]. 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, 1 – 12, 11-14 April 2016.
- [96]. B. T. de Oliveira, L. B. Gabriel and C. B. Margi, "TinySDN: Enabling Multiple Controllers for Software-Defined Wireless Software Networks." IEEE Latin America Transactions, Vol. 13, No. 11, pp. 3690 – 3696, Nov. 2015.
- [97]. 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, Aug. 2016.
- [98]. 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

## REFERENCES

---

- [99]. 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
- [100]. P. Di Dio, S. Faraci, L. Galluccioz, S. Milardoy, G. Morabitoz, S. Palazzoz, and P. Livrieriy. "Exploiting State Information to Support QoS in Software-Defined WSNs." MedHocNet 2016, 2016.
- [101]. L. Huang, G. Li, J. Wu, L. Li, J. Li and R. Morello, "Software-Defined QoS Provisioning for Fog Computing Advanced Wireless Sensor Networks," 2016 IEEE SENSORS, Orlando, FL, 2016, pp. 1-3. Doi: 10.1109/ICSENS.2016.7808814
- [102]. L. Han, S. Sun, B. Joo, X. Jin and S. Han, Jul. 2016, "QoS-Aware Routing Mechanism in OpenFlow-enabled Wireless Multimedia Sensor Networks." International Journal of Distributed Sensor Networks, Vol. 12, No. 7, pp. 1-11.
- [103]. A. Hakiri and A. Gokhale, "Rethinking the Design of LR-WPAN IoT Systems with Software-Defined Networking." 2016 International Conference on Distributed Computing in Sensor Systems, Washington, DC, pp. 238-243, 26-28 May 2016.
- [104]. O. Flauzac, C. Gonzalez and F. Nolot, "SDN Based Architecture for Clustered WSN." 2015 9th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, Blumenau, pp. 342 – 347, 8-10 July 2015.
- [105]. M. L. F. Miguel, M. C. Penna, E. Jamhour, and M. E. Pellenz, "A CoAP Based Control Plane for Software Defined Wireless Sensor Networks" Journal of Communications and Networks, Vol. 19, No. 6, pp. 555-562, 2017.
- [106]. S. W. Pritchard, R. Malekian, G. P. Hancke and A. M. Abu-Mahfouz, "Improving northbound interface communication in SDWSN," IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society, Beijing, 2017, pp. 8361-8366. doi: 10.1109/IECON.2017.8217468
- [107]. 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

## REFERENCES

---

- [108]. R. A. D. Esguerra, A. D. P. Truelen, C. A. M. Festin and W. M. Tan, "SDN-enabled loss mitigation of wireless sensor network data traffic," TENCON 2017 - 2017 IEEE Region 10 Conference, Penang, 2017, pp. 387-392. doi: 10.1109/TENCON.2017.8227895
- [109]. 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
- [110]. R. Kadel, K. Ahmed and A. Nepal, "Adaptive error control code implementation framework for software defined wireless sensor network (SDWSN)," 2017 27th International Telecommunication Networks and Applications Conference (ITNAC), Melbourne, VIC, 2017, pp. 1-6. doi: 10.1109/ATNAC.2017.8215413
- [111]. 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.
- [112]. T. Miyazaki, "Dynamic Function Alternation to Realize Robust Wireless Sensor Network," International Journal of Handheld Computing Research (IJHCR), Vol. 3, No. 3, pp. 17-34, Jul. 2012.
- [113]. M. Mukherjee, L. Shu, T. Zhao, D. Wang and H. Wang, "Lightweight flow management for software-defined wireless sensor networks with link fault in data plane," 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Atlanta, GA, 2017, pp. 998-999. doi: 10.1109/INFCOMW.2017.8116529
- [114]. S. Zidi, T. Moulahi and B. Alaya, "Fault Detection in Wireless Sensor Networks Through SVM Classifier," in IEEE Sensors Journal, vol. 18, no. 1, pp. 340-347, January 2018. doi: 10.1109/JSEN.2017.2771226
- [115]. T. Y. Fu, W. C. Peng and W. C. Lee, "Parallelizing Itinerary-Based KNN Query Processing in Wireless Sensor Networks," in IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 5, pp. 711-729, May 2010. doi: 10.1109/TKDE.2009.146
- [116]. A. Naseem, O. Salem, Y. Liu and A. Mehaoua, "Reliable Vital Sign Collection in Medical Wireless Sensor Networks," 2013 IEEE 15th International Conference on e-Health Networking, Applications and Services (Healthcom 2013), Lisbon, 2013, pp. 289-293. doi: 10.1109/HealthCom.2013.6720687

## REFERENCES

---

- [117]. İ. Aydın, M. Karaköse and E. Akın, "Combined Intelligent Methods Based on Wireless Sensor Networks for Condition Monitoring and Fault Diagnosis", *Journal of Intelligent Manufacturing*, vol. 26, no. 4, pp. 717-729, August 2015, <https://doi.org/10.1007/s10845-013-0829-8>
- [118]. M. Keally, G. Zhou, G. Xing, D. T. Nguyen and X. Qi, "A Learning-Based Approach to Confident Event Detection in Heterogeneous Sensor Networks", *ACM, Journal of Transactions on Sensor Networks (TOSN)*, vol. 11, no. 1, article no.10, pp 10:1-10:28, November 2014. doi:10.1145/2575788
- [119]. P. Vallabh, R. Malekian, N. Ye and D. C. Bogatinoska, "Fall detection using machine learning algorithms," 2016 24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, 2016, pp. 1-9. doi: 10.1109/SOFTCOM.2016.7772142
- [120]. P. Kosmides, E. Adamopoulou, K. Demestichas, M. Anagnostou and A. Rouskas, "On Intelligent Base Station Activation for Next Generation Wireless Networks", *Procedia Computer Science*, vol. 63, pp. 82-88, September 2015, <https://doi.org/10.1016/j.procs.2015.08.315>
- [121]. X. Yue, Y. Liu, J. Wang, H. Song and H. Cao, "Software Defined Radio and Wireless Acoustic Networking for Amateur Drone Surveillance," in *IEEE Communications Magazine*, vol. 56, no. 4, pp. 90-97, APRIL 2018. doi: 10.1109/MCOM.2018.1700423
- [122]. W. Xiang, N. Wang and Y. Zhou, "An Energy Efficient Routing Algorithm for Software-Defined Wireless Sensor Networks," in *IEEE Sensors Journal*, vol. 16, no. 20, pp. 7393-7400, Oct.15, 2016. doi: 10.1109/JSEN.2016.2585019
- [123]. M. M. Hassan and A. Alsanad, "Resource Provisioning for Cloud-Assisted Software Defined Wireless Sensor Network," in *IEEE Sensors Journal*, vol. 16, no. 20, pp. 7401-7408, Oct.15, 2016. doi: 10.1109/JSEN.2016.2582339
- [124]. OpenDayLight, "OpenDayLight: Open Source SDN Platform," 2017. Available at: <https://www.opendaylight.org/>
- [125]. R. Klöti, V. Kotronis and P. Smith, "OpenFlow: A security analysis," 2013 21st IEEE International Conference on Network Protocols (ICNP), Goettingen, 2013, pp. 1-6. doi: 10.1109/ICNP.2013.6733671

## REFERENCES

---

- [126]. A. Lara, A. Kolasani and B. Ramamurthy, "Network Innovation using OpenFlow: A Survey," in *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 493-512, First Quarter 2014. doi: 10.1109/SURV.2013.081313.00105
- [127]. G. Sun, F. Liu, N. Liu and G. Liu, "Extended monitoring with group localization in software defined hybrid Wi-Fi/zigbee networks: An initial prototype," 2014 International Symposium on Wireless Personal Multimedia Communications (WPMC), Sydney, NSW, 2014, pp. 457-461. doi: 10.1109/WPMC.2014.7014862
- [128]. T. Lin, H. Bannazadeh and A. Leon-Garcia, "Introducing wireless access programmability using software-defined infrastructure," 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, 2015, pp. 585-591. doi: 10.1109/INM.2015.7140341
- [129]. S. Kaur, K. Kaur and V. Gupta, "Implementing openflow based firewall," 3rd International Conference on Electrical, Electronics, Engineering Trends, Communication, Optimization and Sciences (EEECOS 2016), Tadepalligudem, 2016, pp. 1-3. doi: 10.1049/cp.2016.1516
- [130]. A. S. Yuan, H. T. Fang and Q. Wu, "OpenFlow based hybrid routing in Wireless Sensor Networks," 2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Singapore, 2014, pp. 1-5. doi: 10.1109/ISSNIP.2014.6827650
- [131]. H. Liaoruo, S. Qingguo, S. Wenjuan and C. Xiaoyu, "Optimizing Segment Routing with the Maximum SLD Constraint using Openflow," in *IEEE Access*. doi: 10.1109/ACCESS.2018.2826925
- [132]. P. Dely, A. Kassler, and N. Bayer, "OpenFlow for Wireless Mesh Networks", Proc. of 20th International Conference on Computer Communications and Networks (ICCCN), pp. 1-6, 2011.
- [133]. M. H. R. Jany, N. Islam, R. Khondoker and M. A. Habib, "Performance analysis of OpenFlow based software defined wired and wireless network," 2017 20th International Conference of Computer and Information Technology (ICCIT), Dhaka, 2017, pp. 1-6. doi: 10.1109/ICCITECHN.2017.8281814
- [134]. Open Networking Foundation, "OpenFlow Switch Specification," *ONF White Paper*, Version 1.5.1 (*Protocol Version 0x06*), ONF TS-025, 26 March 2015.



## REFERENCES

---

- [Online]. Available: <https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf>
- [135]. A. Mahmud and R. Rahmani, "Exploitation of OpenFlow in wireless sensor networks," Proceedings of 2011 International Conference on Computer Science and Network Technology, Harbin, 2011, pp. 594-600. doi: 10.1109/ICCSNT.2011.6182029
- [136]. G. Xiao, W. Wenjun, Z. Jiaming, F. Chao and Z. Yanhua, "An OpenFlow based Dynamic Traffic Scheduling strategy for load balancing," 2017 3rd IEEE International Conference on Computer and Communications (ICCC), Chengdu, 2017, pp. 531-535. doi: 10.1109/CompComm.2017.8322602
- [137]. Z. Can and M. Demirbas, "Querying on Federated Sensor Networks," Journal of Sensor and Actuator Networks, vol. 5, no. 4, p. 14, Sep. 2016.
- [138]. M. Dallaglio, N. Sambo, F. Cugini and P. Castoldi, "Control and management of transponders with NETCONF and YANG," in IEEE/OSA Journal of Optical Communications and Networking, vol. 9, no. 3, pp. B43-B52, March 2017. doi: 10.1364/JOCN.9.000B43

# ADDENDUM A SOFTWARE AND HARDWARE COMPONENTS

The following table provides a description of software and hardware components implemented and used in this work.

Table #: Software, distributions and hardware used.

Simulator / Software Platform	OS Distributions	Hardware
<p><b>NS-3</b> - Version: ns-3.28 Used for network modelling. Models the behavior of sensor nodes.</p> <p><b>OpenDaylight</b> - ODL Release: Oxygen The implanted SDN controller. Provides the central control and intelligence of the system.</p> <p><b>VMware</b> - Type: Workstation Player Provides virtualization platform functionality.</p> <p><b>Developmental Language</b> - C++ For network events coding and programming definitions.</p>	<p><b>Linux Environment</b> - Distribution ID: Ubuntu - Desc: 16.04.4 LTS - Codename: Xenial</p>	<p><b>Desktop Computer</b> - Processor: Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz - RAM: 16.0 GB - System Type: 64-bit OS</p>

# ADDENDUM B      OPENFLOW SOFTWARE SPECIFICATION

Code Procedure	Description
<pre>struct ofp_switch_config {     struct ofp_header header;     uint16_t flags;      uint16_t miss_send_len;</pre>	<p><b><i>OpenFlow Switch configuration</i></b></p>
<pre>struct ofp_table_mod {     struct ofp_header header;     uint8_t table_id;     uint8_t pad[value];     uint32_t config;     struct ofp_table_mod_prop_header properties[0]; };</pre>	<p><b><i>Configuring the behaviour of the table</i></b></p>
<pre>struct ofp_switch_features {     struct ofp_header header;     uint64_t datapath_id;      uint32_t n_buffers;     uint8_t n_tables;     uint8_t auxiliary_id;     uint8_t pad[value];    /* Align to 64-bits. */      /* Features. */     uint32_t capabilities;     uint32_t reserved; }; OFP_ASSERT(sizeof(struct ofp_switch_features) == 32);</pre>	<p><b><i>OpenFlow switch features</i></b></p>

<pre> struct ofp_instruction_goto_table {     uint16_t type;     uint16_t len;     uint8_t table_id;      /* Set next table in the lookup pipeline */     uint8_t pad[value]; }; OFP_ASSERT(sizeof(struct ofp_instruction_goto_table) == 8); </pre>	<p><b>OpenFlow procedure to traverse to a certain tale</b></p>
<pre> struct ofp_instruction_write_metadata {     uint16_t type;          uint16_t len  uint8_t pad[4];     uint64_t metadata;     uint64_t metadata_mask; }; OFP_ASSERT(sizeof(struct ofp_instruction_write_metadata) == 24);  struct ofp_instruction_actions {     uint16_t type;     uint16_t len;     uint8_t pad[4];     struct ofp_action_header actions[0]; }; OFP_ASSERT(sizeof(struct ofp_instruction_actions) == 8); </pre>	<p><b>OpenFlow instruction set to write meta data</b></p> <p><b>OpenFlow instruction set to write, apply and clear actions</b></p>
<pre> struct ofp_packet_in {     struct ofp_header header;     uint32_t buffer_id;     uint16_t total_len;     uint8_t reason;     uint8_t table_id;      uint64_t cookie;     struct ofp_match match; }; OFP_ASSERT(sizeof(struct ofp_packet_in) == 32); </pre>	<p><b>OpenFlow packet receive confirmation from controller datapath</b></p>

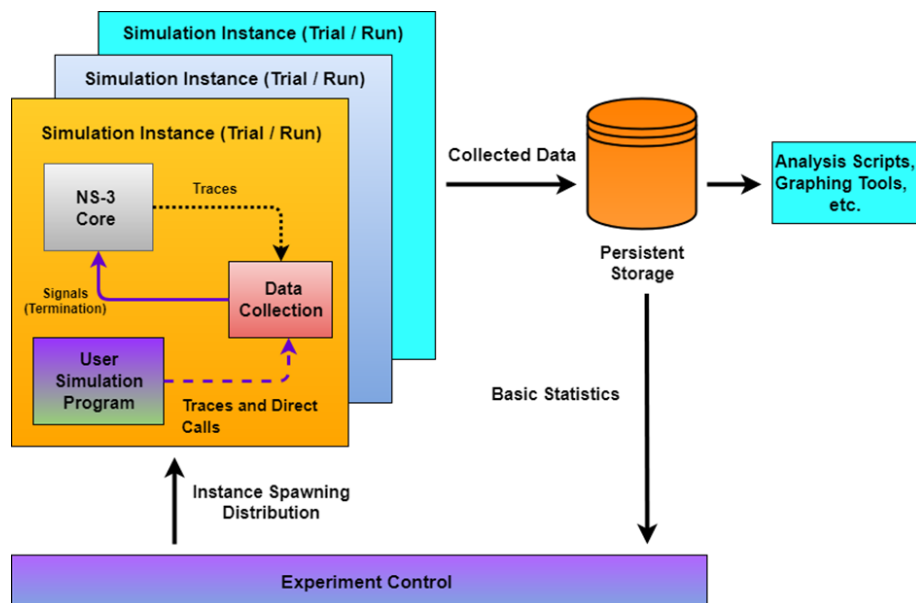
# ADDENDUM C ODL OXYGEN RELEASE FRAMEWORK



## ADDENDUM D      NS-3 STATISTICAL FRAMEWORK

The NS-3 Statistical framework is based on the following principles:

- Each experimental trial is conducted by a one simulation program instance, regardless of the code's procedural type.
- Instances of the simulation are executed by a control script and parameters are varied accordingly.
- Collected data is stored for visual plots and data analysis using other third-party scripts and existing analysis tools. Measurements are collected through the statistics framework. Custom simulation codes may be instrumented by manipulating the framework of using trace signals.



## ADDENDUM E      SDN BASED WSN SIMULATION CODE SNIPPETS

<pre>#include &lt;iostream&gt; #include "ns3/netanim-module.h" #include "ns3/core-module.h" #include "ns3/network-module.h" #include "ns3/internet-module.h" #include "ns3/wifi-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 &lt;cmath&gt; #include "ns3/gnuplot.h" #include &lt;string&gt; #include &lt;cassert&gt; #include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #include "ns3/sixlowpan-module.h" #include "ns3/basic-energy-source.h" #include "ns3/simple-device-energy-model.h"</pre>	Header Files
<pre>using namespace ns3;  NS_LOG_COMPONENT_DEFINE("SDWSN");  int main (int argc, char *argv[]) {     uint16_t NumberOfNodes = 100;     uint16_t NumberOfBaseStationNodes = 3;     double SimTime = 60;     double distance = 150.0;     double InterPacketInterval = 150;</pre>	Definitions and declarations

<pre>uint32_t ResourceID = 0; uint32_t Protocol = 1;</pre>	
<pre>CommandLine cmd; cmd.Addvalue("sensorNode", "NumberOfNodes", sensorNode); cmd.Addvalue("BSNode", "NumberOfBaseStationNodes", BSNode); 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);</pre>	Configuring command line parameters
<pre>NodeContainer sensorNodes; NodeContainer BSnodes; BSnodes.Create (NumberOfBaseStationNodes); sensorNodes.Create (NumberOfNodes);  for (uint32_t u = 0; u &lt; sensorNodes.GetN (); ++u) {     Ptr &lt;Node&gt; ipv6Node;     Ipv6Node = sensorNodes.Get(u);     Ptr &lt;Ipv6StaticRouting&gt; ipv6StaticRouting;     Ipv6StaticRouting = Ipv6RoutingHelper.GetStaticRouting (ipv6Node) -&gt; GetObject &lt;Ipv6&gt; (); } }</pre>	Configuring nodes as IPv6 devices
<pre>stack.Install (allNodes); Ipv6AddressHelper     SetBase (Ipv6Address ("2001:1::"), Ipv6Prefix (64)); Ipv6InterfaceContainer = ipv6.Assign (sensorNodes); Ipv6InterfaceContainer staInterfaces; staInterfaces = address.Assign (staNodes); Ipv6InterfaceContainer apInterfaces; apInterfaces = address.Assign (apNodes);</pre>	Enable IPv6 addressing
<pre>Ptr &lt;BasicEnergySource&gt; energySource; energySource = CreateObject &lt;BasicEnergySource&gt; (); Ptr &lt;SimpleDeviceEnergyModel&gt; energyModel; energyModel = CreateObject &lt;SimpleDeviceEnergyModel&gt; (); energySource -&gt; SetInitialEnergy (300); energyModel -&gt; SetEnergySource (energySource); energySource -&gt; AppendDeviceEnergyModel (energyModel); energyModel -&gt; SetCurrentA Ipv6StaticRouting -&gt; SetDefaultRoute (ipv6Helper -&gt; GetIpv6DefaultGatewayAddress (), 1); (20);</pre>	Configuring the energy of the sensors
<pre>for (uint32_t u = 0; u &lt; sensorNodes.GetN (); ++u) {     Ptr &lt;Node&gt; ipv6Node;     Ipv6Node = sensorNodes.Get (u);     Ptr &lt;Ipv6StaticRouting&gt; ipv6StaticRouting;     Ipv6StaticRouting = Ipv6RoutingHelper.GetStaticRouting (ipv6Node -&gt; GetObject &lt;Ipv6&gt; ());</pre>	Object route processing



<pre> } for (uint32_t i = 0; i &lt; sensorNodes.GetN (); ++i) {     anim.UpdateNodeDescription (sensorNodes.Get(i), "sensorNodes");     anim.UpdateNodeColor (sensorNodes.Get (i),0, 255, 0); }  for (uint32_t i = 0; i &lt; BSnodes.GetN (); ++i) {     anim.UpdateNodeDescription (BSNodes.Get(i), "BS");     anim.UpdateNodeColor (BSNodes.Get(i),0, 0,255); } </pre>	<p>Configuring the animation of the simulation</p>
<pre> TapBridgeHelper tapBridge; tapBridge.SetAttribute ("Mode", StringValue ("ConfigureLocal")); tapBridge.SetAttribute ("DeviceName", StringValue ("ctrl")); tapBridge.Install (controllerNode, ctrlDev); </pre>	<p>Configuring the network to communicate with external controller via port 6653</p>
<pre> NodeContainer OpenDayLight; OpenDayLight.Create (1); SpectrumAnalyzerHelper spectrumAnalyzerHelper;  if (generateSpectrumTrace) {     ptr&lt;ListPositionAllocator&gt; positionAlloc = CreateObject&lt;ListPositionAllocator&gt; ();     positionAlloc-&gt;Add (vector (distance * 0.5, distance * 0.866, 0.0));     spectrumTraceHelper spectrumAlloc;     spectrumAlloc.SetLocationModel ("ns3::ConstantPositionMobilityModel");     spectrumAlloc.SetPositionAllocator (external_Controller);     spectrumAlloc.Install (OpenDayLight);      ptr&lt;LteSpectrumPhy&gt; enbDISpectrumPhy =BSNodes.Get (0)-&gt; GetObject&lt;LteEnbDevice&gt; ()-&gt;GetPhy ()-&gt; GetDownLinkSpectrumPhy ()-&gt; GetObject &lt;LteSpectrumPhy&gt; ();     ptr&lt;SpectrumChannel&gt; dlChannel = enbDISpectrumPhy-&gt;GetChannel ());      SpectrumAnalyzerHelper.SetChannel(dlChannel);     ptr&lt;SpectrumModel&gt; sm = LteSpectrumValueHelper::GetSpectrumModel(120, bandwidth);     SpectrumAnalyzerHelper.SetRxSpectrumModel (sm);     SpectrumAnalyzerHelper.SetPhyAttribute "Resolution", TimeValue(microSeconds (20));     SpectrumAnalyzerHelper.SetPhyAttribute("NoisePowerSpectralDensit y", DoubleValue (1e+15)); </pre>	<p>NS3 OpenDayLight controller implementation and controller spectrum allocation</p>

<pre>SpectrumAnalyzerHelper.EnableAsciiAll("spectrum.analyser.output"); SpectrumAnalyzerHelper.Install(OpenDayLight); }</pre>	
<pre>if (trace) {     openflowHelper-&gt;EnableOpenFlowPcap ("openflow");     openflowHelper-&gt;EnableDatapathStats ("AllNodes-stats");     csmaHelper.EnablePcap "BSNodes";     csmaHelper.EnablePcap "sensorNodes"; }</pre>	Enable datapath stats and pcap traces at network elements and controller