LEVERAGING SOFTWARE-DEFINED NETWORKING FOR MODULAR MANAGEMENT IN WIRELESS SENSOR NETWORKS

by

Musa Ndiaye

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

in the

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

UNIVERSITY OF PRETORIA

August 2019

SUMMARY

LEVERAGING SOFTWARE-DEFINED NETWORKING FOR MODULAR MANAGEMENT IN WIRELESS SENSOR NETWORKS

by

Musa Ndiaye

Supervisor:	Dr. G.P. Hancke
Co-Supervisor:	Dr. A.M. Abu-Mahfouz
Department:	Electrical, Electronic and Computer Engineering
University:	University of Pretoria
Degree:	Philosophiae Doctor (Electronics)
Keywords:	Wireless sensor networks, SDN-based wireless sensor networks, network management architecture, management service interface, modularity, control message quenching, 6LoWPAN, neighbour reports, control traffic

Wireless sensor networks (WSNs) are becoming increasingly popular with the advent of the internet of things (IoT). Various real-world applications of WSNs such as in smart grids, smart farming, and smart health would require a potential deployment of thousands or maybe hundreds of thousands of sensor nodes/actuators. To ensure the proper working order and network efficiency of such a network of sensor nodes, an effective WSN management system has to be integrated. However, the inherent challenges of WSNs such as sensor/actuator heterogeneity, application dependency, and resource constraints have led to challenges in implementing effective traditional WSN management. This difficulty in management increases as the WSN becomes larger. Software-defined networking (SDN) provides a promising solution for flexible management of WSNs by allowing the separation of the control logic from the sensor nodes/actuators. The advantage with this SDN-based management in WSNs is that it enables centralized control of the entire WSN making it simpler to deploy network-wide management protocols and applications on demand. Therefore in a comprehensive literature review, this study highlights some of the recent work on traditional WSN management in brief and reviews SDN-based management techniques for WSNs in greater detail. All this while drawing attention

towards the advantages that SDN brings to traditional WSN management. This study also investigates open research challenges in coming up with mechanisms for flexible and easier SDN-based WSN configuration and management.

A profound research challenge uncovered in the literature review is the need for an SDN-based system that would provide an opportunity for rapid testing and implementation of management modules. Therefore, this study proposes SDNMM, a generic and modular WSN management system based on SDN. SDNMM introduces the concept of management modularity using a management service interface (MSI) that enables management entities to be added as modules. The system leverages the use of SDN in WSNs and by being modular it also allows for rapid development and implementation of IoT applications. The system has been built on an open-source platform to support its generic aspect and a sample resource management module implemented and evaluated to support the proposed modular management approach. Results showed how adding a resource management module via the MSI improved packet delivery, delay, control traffic and energy consumption over comparable frameworks.

However, SDN-based implementation comes at a cost of control overhead traffic which is a performance bottleneck in WSNs due to the limited in-band traffic channel bandwidth associated with WSNs. This has driven the research community to look into methods of effectively reducing the overhead control traffic in a process known as control message quenching (CMQ). In this study, a state of the art overview of control traffic reduction techniques available and being implemented for SDN-based WSNs is also presented. It provides an insight on benefits, challenges and open research areas available in the field of control message quenching for SDN-based WSNs. This study opens the door to this widely unexplored research area in its current form.

Additionally, this study introduces a neighbour discovery control message quenching (ND-CMQ) algorithm to aid the reduction of neighbour reports in an SDN-based 6LoWPAN framework. The algorithm produces a significant decrease in control traffic and as a result shows improvements in packet delivery rate, packet delay, and energy efficiency compared to not implementing any CMQ algorithm and also compared to an alternative FR-CMQ algorithm based on flow setup requests.

DEDICATIONS

This thesis is dedicated to my late father Papa Hamet Ndiaye and my late guardian Papa Moussa Dia. Thank you for instilling the value of strict discipline and prayer in me. Thank you for providing an opportunity for a great foundation for my education. I will forever be grateful for the treasures you have provided. May Allah grant you both Janaatul Firdaus.

ACKNOWLEDGEMENTS

I thank the Almighty Allah for giving me the opportunity to pursue my PhD study and also for providing the necessary strength, wisdom and intellectual capacity to complete this work.

I would like to thank my supervisors, Dr. Gerhard Hancke and Dr. Adnan Abu-Mahfouz for their tireless efforts in ensuring that this work is completed accordingly. I am grateful for being constantly available for consultation and guidance even on weekends. I would further like to thank Dr. Abu-Mahfouz for his dedication towards following up the work progress in this study, his efforts in taking the time to set up progress meetings every two weeks are greatly appreciated. Special thanks to Prof. Gerhard Hancke senior, for overlooking all my affairs from the beginning of my study to the end. I owe him abundant gratitude for his outstanding efforts and support. I also thank the Copperbelt University under the ministry of higher education in Zambia and the Council for Scientific and Industrial Research (CSIR) for financial support. I would also like to thank my friend and colleague Jaco Marais for his support and efforts in helping me improve my computer engineering skills. He contributed invaluably towards changing my view of efficient skills and personal development. Many thanks also go to my colleagues in the ASN research group for their various inputs towards perfecting my study.

I also thank my family especially my mother for her prayers and guidance. I thank my friends and relatives for the special support and patience as I had been away studying. May Allah bless you all.

LIST OF ABBREVIATIONS

6LoWPAN	IPv6 over Low-power Wireless Personal Area Networks
ANMS	Active Network Management System
API	Application Programming Interface
BLIP	Berkley Low power IP
BOSS	Bridge of the SenSors
CMQ	Control Message Quenching
CRLB	Cramer-Rao Lower Bound
CSIR	Council for Scientific and Industrial Research
DoS	Denial of Service
EASA	Energy Aware Adaptive Sampling Algorithm
EPMOSt	Energy-efficient Passive MOnitoring SysTem
ETX	Estimated Transmission Count
FPGA	Field Programmable Gate Array
FR-CMQ	Flow Request Control Message Quenching
GUI	Graphical User Interface
IoT	Internet of Things
IP	Internet Protocol
ISP	Internet Service Providers
KiBaM	Kinetic Battery Model
LB	Lower Bound
LNMP	LowPAN Network Management Protocol
LoRa	Long Range
LoWPAN	Low-power Wireless Personal Area Networks
LPWAN	Low Power Wide Area Networks
LQI	Link Quality Indicator
LR-WPAN	Low Rate Wireless Personal Area Network
M2M	Machine to Machine
MbD	Management by Delegation
MILP	Mixed Integer Linear Program
MSI	Management Service Interface

NAVS	Network Assisted Video Streaming
ND	Neighbor Discovery
ND-CMQ	Neighbor Discovery Control Message Quenching
NMS	Network Management System
NOS	Network Operating System
OS	Operating System
OTAP	Over the Air Programming
PDR	Packet Delivery Rate
PPBR	Polynomial Pool and Basic Random
PPI	Proactive Probing Interval
QoS	Quality of Service
QoSen	Quality of Sensing
RSSI	Received Signal Strength Indicator
SDCSN	Software-Defined Clustered Sensor Networks
SDN	Software-Defined Networking
SDN-WISE	SDN-WIreless SEnsor network
SDNMM	Software-Defined Networking based Modular Management
SDWN	Software-Defined Wireless Network
SDWSN	Software-Defined Wireless Sensor Network
SI	Sampling Interval
SMC	Secure Multiparty Computation
SNMP	Simple Network Management Protocol
SNMS	Sensor Network Management System
SoF	Sensor OpenFlow
STEM	Sparse Topology and Energy Management
TCP	Transmission Control Protocol
UB	Upper Bound
UP	University of Pretoria
UPnP	Universal Plug and Play
WBAN	Wireless Body Area Network
WCN	Wireless Cellular Network
WinMS	Wireless sensor network Management System
WRAN	Wireless Regional Area Network
WSN	Wireless Sensor Network

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 BACKGROUND

The Internet of Things (IoT) has attracted a lot of attention from developers and researchers with various applications requiring the deployment of Wireless Sensor Networks (WSNs). A good example is an implementation of a Smart Water Distribution Network (SWDN) [1, 2] which would involve a network of hundreds or even thousands of sensor/actuator nodes required for measurement and analysis of data. Other potential applications of IoT on a large scale include smart city [3, 4], and smart healthcare systems [5] both of which would also result in big data generation. Consequently, we will experience multiple sensor system implementations with varying vendor sources, each with application-specific requirements. This directly results in a major bottleneck in terms of vertical integration of applications. Modifications, updates and troubleshooting of the sensor network will increasingly become difficult for network administrators and managers [6] especially as the application sizes scale up to meet modern industry.

Network management is essential in ensuring the continuous operation of communication networks whether computer or sensor-based. The challenges highlighted above show that it be critical for a WSN management system to be put in place for the proper and healthy operation of the numerous sensor/actuator nodes. In the absence of an active and efficient network management system, it would be increasingly difficult to manage and operate the thousands of sensors/actuators deployed over a significantly large area. WSN management systems such as BOSS [7] and MANNA [8] exist, however, the efficiency of such systems is limited due to the heterogeneous, application-specific and resource-constrained nature of WSNs. Managing and configuring such networks especially at a large scale is a challenging task for the responsible network operators who are also expected to respond to a wide range of events that may result during operation.

To solve this problem in the management of large scale WSNs, several techniques have been researched and developed and increasingly some of these methods revolve around the concept of Software-Defined Networking (SDN). SDN has evolved into a leading architecture for networking. It is based on the principle of dividing and separating the network into two separate planes; the control plane which determines the traffic routes and the data plane which forwards the traffic packets [9, 10]. Implementing SDN in wireless sensor networks results in a modern architecture commonly referred to as Software-Defined Wireless Sensor Network (SDWSN) which promises greater efficiency in network management. This vision that SDN promises is mostly based on the introduction of logically centralized control of network operations backed by the global view of the network that the SDN controller maintains.

1.2 PROBLEM STATEMENT

SDN was designed for wired networking and therefore using it for WSNs would pose a challenge due to limited node resources and challenges inherent in the WSN architecture. The promising SDWSN model would allow for ease and flexibility in network management to a level where building a Network Management System (NMS) would be no different from adding another application to the control plane. This has generated rapid interest from the research community to quickly adopt SDN for pre-existing and future WSN implementations. As interest grows towards large scale application of IoT technology in various industry sectors, we can only expect rapid expansion in the application and adoption of the SDWSN architecture.

1.2.1 Context of the problem

The interest that SDWSN has generated has resulted in efforts to allow SDN to be implemented as part of the WSN architecture. Proposals of methods to integrate SDN are increasingly being made to allow more applications to adopt this paradigm. Consequently, there are now multiple available platforms for implementing SDWSNs such as that presented in [11–13] with great emphasis being placed on the network management flexibility that SDN promises. This calls for a greater interest in the resulting implementation strategy considering that fewer contributions are being made towards network management in specific. Upon application of SDN in respective sensor network deployments, network managers would still need generic guidance towards issuance of management policies. With the current state of adoption, questions such as: how easily can management techniques and policies

be integrated or removed? how will different vendors provide management applications to meet a wide range of implementations? to mention a few, would still arise. These are questions that an SDN-based management framework or system for WSNs would provide solutions to.

Some real-world scenarios that the SDN-based management system proposed in this thesis aims to address include applications resulting in difficulty with managing wireless sensor nodes post deployment such as in harsh or hard to reach areas and applications that require large-scale WSN implementations. Typical examples include management of sensor nodes for military operations deployed in war zones, maritime operations where thousands of nodes are deployed on the sea or ocean bed, management of nodes deployed in chemically or thermally harsh environments and similar scenarios. Applications such as smart water grids and smart power grids require large scale implementation of sensor nodes that need consistent, flexible and effective management. In both scenarios above, an SDN-based modular management system would allow for rapid deployment and/or removal of management protocols including the effective monitoring and control of large-scale WSNs.

Traditional wireless sensor networks have had contributions towards the development of Active Network Management Systems (ANMS) to intentionally guide managers on network management [7, 8, 14]. Aspects of how to implement management policies and controls can be included within the management system and there must be an aspect of being generic to allow use and rapid prototyping by different vendors. As regards SDWSN, there is still more work required in developing generic SDN-based management frameworks to meet the said requirements for the desired management efficiency and flexibility.

1.2.2 Research gap

An SDN-based modular management system for WSNs would provide some of the management solutions that are currently being demanded by SDWSN implementations. Such a system would enable network administrators to rapidly implement or remove management applications. This is possible based on the management modularity concept which looks at management applications such as energy, security, resource or topology management as components or modules. When such a system is integrated as part of the controller, this would greatly leverage SDN in improving management

flexibility and efficiency across several SDWSN applications. This study addresses the need for such an efficient management system for SDN-based WSN applications. Furthermore, as part of the study, the proposed system is designed and implemented on a selected opensource SDWSN platform to bring in the generic aspect. The baseline platform was selected based on resource availability and developer support to ensure widespread use in the research community. To prove the feasibility of the proposed modular management system, a sample resource allocation management module is built and the performance is evaluated against baseline SDWSN platforms without any management system in place. The results show various performance improvements due to the implemented management module hence proving the efficacy of the modular management concept leveraging SDN.

1.3 RESEARCH OBJECTIVE AND QUESTIONS

The following are the objectives of the research study:

- To develop a clear understanding of WSN management based on SDN and clearly identify existing open challenges.
- To propose and implement a modular management system in an SDWSN environment.
- To develop a sample resource allocation management module suitable for use in the modular system.
- To demonstrate through evaluation that a modular management system is feasible and can be used to improve network efficiency.

The research questions that this study aims to answer include:

1. How can we leverage SDN in developing an efficient and comprehensive management framework for flexible and versatile functionality in WSNs?
2. How can a modular management system based on SDN for WSNs be developed and evaluated for improved management efficiency?
3. Can a resource allocation management module developed for a modular system improve management of the network Quality of Service (QoS) metrics?

1.4 HYPOTHESIS AND APPROACH

A generic SDN-based modular management framework is proposed to envision efficient and flexible network management of wireless sensor networks. The proposed architecture based on SDN also integrates a two-tier control plane for implementation of a global controller and distributed controllers. This aspect has been included in the framework to provide future support for extended network scalability when the management system is implemented. This study seeks to provide a modular solution to implementing management policies and techniques across the network. The end result would be the rapid prototyping of management techniques even over varying vendor applications. To enable this kind of management implementation, this study proposes a form of a common interface for the management modules to easily be integrated into the framework. The interface referred to as a Management Service Interface (MSI) should use context information which can be collected according to the needs of the management module to be implemented.

A context-aware management service interface can rely on a context data knowledge base built and populated in the controller to provide input requirements into modules. Therefore, each module can be designed with a context-based input requirement of the data plane network and output can be a management decision or policy based on the input context data. A security module, for example, can require context data based on the encryption status of data in the data plane or data values which are essential for threat detection computation in the controller. The module can then output and send down policies based on the network context. Another example can be a topology module which can perform routing decisions based on node state such as congestion levels as intended by Dio *et al*. [15]. Node states can be part of the context data knowledge base making the implementation of such a system easier. To evaluate this modular operation, implemented as part of this study is a resource allocation module. This module can take in as context data input the node resource values such as battery level, number of neighbours, pending tasks and Received Signal Strength Indicator (RSSI). The performance in terms of quality of service metrics such as packet delivery rate, packet delay, control overhead and energy efficiency of the resource module plays an important role in proving the possibility and effectiveness of such a modular management system. The approach to ensuring that this framework is generic is also based on implementing the framework on an open and readily available SDWSN framework. ITSDN [13] proves a suitable candidate based on the literature study. This will enable the research community to easily access the platform and implement the modular framework according to required specifications.

This study aims to prove the following two hypotheses:

- A generic and modular management system is possible for improvement of SDN-based management of wireless sensor networks.
- A resource allocation module integrated as part of the controller can be used to improve the SDWSN performance metrics.

1.5 RESEARCH GOALS

The research goals of this study are as follows:

1. to leverage software-defined networking for improved and efficient management of wireless sensor networks.
2. to develop a generic and modular management system for wireless sensor networks based on SDN and build an accompanying sample resource allocation module to demonstrate performance efficacy.

1.6 RESEARCH CONTRIBUTION

This study makes the following contributions to research:

- The management aspect of SDWSNs although important has not been vastly investigated unlike the rapidly expanding proposals on SDN-based implementation technologies. In this regard, this study makes a comprehensive literature survey on SDN for improved wireless sensor network management. The survey takes a look at various aspects of SDN-based management including application opportunities and identifies the associated research gaps. This survey has been published in the MDPI Sensors Journal.
- In line with a critical research gap uncovered in the literature review, a generic and modular management framework for WSNs leveraging SDN has been proposed. This study carries out research to design and implement the novel framework to generate interest in the research community towards flexible modular management. As part of the framework, a WSN resource allocation module is developed and implemented. The performance results based on the included

resource allocation module is evaluated to demonstrate the feasibility of a modular management system in increasing management efficiency. This system and associated performance evaluation of the sample module were submitted and accepted for publication in the IEEE Systems Journal.

- The modular management mechanism is based on the population of context data in the controller for the builtin management service interface. This requires the generation of context data which is sent to the controller using the limited in-band channel that exists for both control and data traffic. There is, therefore, a potential bottleneck due to overhead control traffic. As a result, this study embarks on a further research contribution investigating and discussing techniques to quench and/or minimize general control traffic in SDN-based WSNs. The first part involved carrying out a state of the art overview of Control Message Quenching (CMQ) techniques for SDN-based WSN with a background given on CMQ for computer network based SDN. Findings here were compiled and accepted as part of the proceedings for the 2019 SATNAC conference. Further research was carried out on implementing CMQ based on flow setup requests for SDN-based WSN. The goal here was to implement an algorithm to effectively reduce duplicate flow setup requests which result from multiple mismatches at SDN-enabled nodes. This contribution was submitted and accepted as part of the INDIN 2019 conference proceedings. The study in particular study also resulted in another novel research contribution based on neighbour discovery CMQ for SDN-based 6LoWPAN (IPV6 Low-power Wireless Personal Area Networks). A two-step algorithm was proposed and implemented to optimize neighbour report generation for CMQ purposes. The goal of this contribution was to minimize neighbour report generation and duplication while keeping the desired QoS in terms of packet delivery, delay and energy efficiency. This work was submitted for publication in the IEEE Wireless Communications Letters and is currently under review.

1.7 RESEARCH OUTPUTS

This research study has the following outputs:

1. M. Ndiaye, G. P. Hancke, and A. M. Abu-Mahfouz, “Software-defined networking for improved wireless sensor network management: A survey,” *Sensors*, vol. 17, no. 5, pp. 1-32, 2017.
2. M. Ndiaye, A. M. Abu-Mahfouz, and G. P. Hancke, “SDNMM - A generic SDN-based modular management system for wireless sensor networks,” *IEEE Systems Journal*, pp. 1-11, 2019, to be

published.

3. M. Ndiaye, G. P. Hancke, and A. M. Abu-Mahfouz, "Towards control message quenching for SDWSN: A state of the art overview," in *Proceedings of the 22nd Annual SATNAC Conference*, September 2019, in press.
4. M. Ndiaye, A. M. Abu-Mahfouz, and G. P. Hancke, "Exploring control message quenching in SDN-based management of 6LoWPANs," in *Proceedings of the 17th IEEE International Conference on Industrial Informatics (INDIN)*, July 2019, pp. 890-983.

1.8 DELINEATION AND LIMITATIONS

The field of software-defined wireless sensor networks is still fairly new and this study experienced unavailable SDWSN testbeds for experimentation at the time of research. In this regard, all evaluations performed in this research are simulation based on using the IT-SDN platform and Contiki Cooja simulation tool. The study assumed that the SDN-enabled node and controller firmwares compiled for simulation were working as expected.

1.9 OVERVIEW OF STUDY

The rest of this thesis is organised as follows: Chapter 2 reviews comprehensively literature on management of WSNs. It keeps abreast of traditional techniques in managing WSNs and delves further into SDN-based management techniques presenting application opportunities and research gaps in this area. In Chapter 3, a novel SDN-based modular management system for wireless sensor networks is presented. All aspects of system design including the architectural framework, functionality concept, system component design algorithms and mode of implementation are discussed here. Evaluation of the proposed management framework is also presented in Chapter 3. This includes the simulation setup for the various categories of evaluation, result presentation, and discussion. Challenges of the system are also discussed based on the result findings. In Chapter 4, the study investigates opportunities in minimizing control traffic; an essential aspect in optimizing the proposed system operation. A novel two-step algorithm to quench control messages based on neighbour discovery is presented and evaluated in this chapter. Finally, the thesis is concluded in Chapter 5 and future research opportunities presented therein.

CHAPTER 2 LITERATURE STUDY

2.1 CHAPTER OVERVIEW

Software-defined networking introduces inherent flexibility in network management as a whole. This property has resulted in a welcome approach to the management of wireless sensor networks in general. Needless to say, there have been several attempts to better improve the management efficiency and flexibility of WSNs in literature before the advent of SDN technology and also further techniques have been proposed integrating SDN. This chapter provides an updated overview of management in WSNs by reviewing various contributions in management of WSNs and also discusses the adoption of SDN-based management. In general, this work identifies essential components of WSN management, its associated challenges and how SDN can be leveraged to improve the efficacy of this form of network management including the research gaps associated with leveraging SDN for effectively managing WSNs. This work has been published in "Software-defined networking for improved wireless sensor network management: A survey" [16], a journal article that sets into light the various advantages that SDN brings to the management of WSNs.

The rest of this chapter is outlined as follows. In Section 2.2 an overview of wireless sensor network management is provided and followed by a discussion on network management based on the SDN-paradigm in Section 2.3. Section 2.4 gives a general outlook on real-world applications of WSNs and the management potential that SDN can provide to these applications. An overview of SDN-based management of WSNs specifically highlighting SDN techniques and proposed architectures for managing various aspects of WSNs based on SDN is then given in Section 2.5. Section 2.6 narrows down the discussion to specific contributions in literature towards SDN-based management of WSNs. Future research challenges associated with the management of WSNs based on SDN are then identified in Section 2.7 of this chapter. The chapter summary is provided in Section 2.8.

2.2 OVERVIEW OF WIRELESS SENSOR NETWORK MANAGEMENT

Management of WSNs should allow for the definition of a set of functions that promote productivity and integration in an organized manner of the configuration, maintenance, operation and administration of the components and services of the WSN [8]. Several management methods have been proposed to manage functionality in the architecture of WSNs [17]. These methods take into account WSN metrics such as energy consumption, system lifespan, data latency, system tolerance to faults, accuracy in data acquisition or the quality of service and security.

WSN management should be simple and adhere to network dynamic behaviour, as well as provide efficiency in the use of resources as proposed by Ruiz *et al*. [8] in the MANNA (a management architecture for wireless sensor networks) architecture. The MANNA architecture considers management policies for WSN services, functions and models by looking at the management of WSNs in three dimensions defining functionality abstractions:

- WSN functionalities which include maintenance, configuration, sensor node operation (sensing, processing, communication).
- Management levels which include application services and management of network elements (node clusters, data aggregation, network connectivity).
- Management of functional areas such as security, fault monitoring, performance, and configuration.

In this chapter, the structure of WSN management schemes presented is based on the above protocols [17] and abstraction entities [8] however they will be re-categorised according to functional relationships as follows:

a) Network configuration management: All issues associated with network configuration and operation are categorized here. This includes protocol implementation, configuration of data acquisition, network service, network level programming issues.

b) Topology management: This management category handles issues related to the layout of the WSN. Management of sensor node location and distribution, network activity distribution, node to node communication including gateway elements fall under topology management.

c) QoS management: Data latency, system performance, fault tolerance, data acquisition accuracy are among the issues managed under this category to ensure an optimum WSN quality of service.

d) Energy management: All parameters regarding energy consumption in the network are categorized here including energy sources, energy consumption minimization and system lifespan in relation to available energy resources.

e) Security management: Considering the popularity of WSNs, more sensitive information is being transmitted over such networks thus it is necessary that the network is protected from malicious attacks. This would mostly involve management of network security functionalities such as encryption (key distribution techniques), threat detection and recovery which are categorized here.

f) Maintenance management: Wireless sensor network aspects related to maintaining correct operation of the network are classified under this management category. Monitoring of network performance, energy levels, and faults define some of these aspects.

As shall be discussed depending on the availability of the management techniques for the categories above, WSNs can be tailored to meet at least one or more of the following design criteria:

a) Energy Efficiency- This takes into account the ability of a system to conserve energy or allow operation on limited power for long periods resulting in improved network lifetime.

b) Robustness- This criterion ensures that a system performs as expected regardless of varying environmental conditions or design requirements [9]. A robust system should produce desirable performance despite network variations such as node failure, power outages and instabilities resulting from node mobility. An important characteristic of a robust management system is network reconfiguration [18].

c) Scalability- WSN nodes are expected to scale up to very large numbers, therefore, a scalable management system should function efficiently at any network scale. Distributed management plays an important role here while reducing traffic overhead which may otherwise be all directed to a centralized activity manager.

d) Adaptability- This criterion refers to a system's ability to meet network variations and task demands. The system should be able to work efficiently in varying network conditions such as energy fluctuations, topology changes, and task variation. The ability to reconfigure and re-task also plays an important role in meeting this criterion.

2.2.1 Network configuration management

Wireless sensor network configuration management integrates network configuration, performance management and maintenance management of the sensor nodes. To ensure correct network operation, the network has to be configured taking into account events and problems likely to occur in the field. Ruiz *et al*. [8] define some requirements for configuration management at network level such as environmental variation data, operational environment specification, topology discovery, synchronization, node programming and node deployment. When the network is correctly configured and managed, information on the node location, administration state, usage state, and energy level can easily be acquired.

Apart from MANNA, Bridge of the Sensors (BOSS) has been presented as a network management framework creating a bridge based on the resource demanding UPnP (Universal Plug and Play) protocol [7]. This enables the resource constrained sensor network to deploy its services to devices based on the UPnP protocol with zero configuration. The architecture is composed of three parts: BOSS which forms the base node, UPnP control point and the non-UPnP nodes in the WSN. BOSS also provides the user with sensor network management services based on UPnP. It is worth noting that both MANNA and BOSS are based on the central management approach. A distributed management scheme allows for management tasks to be shared across nodes in the WSN also called Management by Delegation (MbD). This kind of setup reduces task loading on the central manager however the delegation process requires the use of intelligent nodes or mobile agents. Mobile agents are simply code sections required to distribute management tasks for execution locally on nodes while returning the resulting data to the central management node [8]. Lee *et al*. [18] discuss protocol functions based on MbD such as Agilla, sectoral sweepers, intelligent based power management and mobile agent-based policy management.

2.2.2 QoS management

Coupled with configuration management is performance management. The two main objectives of performance management in a WSN are the quality of service and the quality of the information obtained [8]. However, there is usually a trade-off between QoS, energy consumption and network lifetime [8, 19]. Depending on the environmental conditions, resource constraints, service demand;

the management protocols can permit lower QoS when energy is scarce and thus extending network lifetime. Several network management protocols have been proposed in QoS management of the WSN architecture including the hybrid data dissemination framework protocol RRP, Sensor Network Management System (SNMS), Simple Network Management Protocol (SNMP) and Wireless Sensor Network Management System (WinMS) [18]. SNMP is a widely used management protocol that provides good management for TCP/IP based networks and is supported by several vendors in managing both wired and wireless networks. It allows management of network performance and enables fault identification. WSNManagement [20] based on SNMP protocols has been proposed as an efficient network configuration, fault and performance management system. The performance of WSNManagement showed efficiency in WSN management with a 5 percent reduced packet loss and a 0.2 second reduced delay time while improving the lifetime of the overall sensor node system. Delay-related QoS is critical for WSNs, especially for real-time applications. Apart from WSNManagement, Zhao *et al*. [21] propose an optimized resource allocation solution for delay constrained Wireless Regional Area Networks (WRANs) to improve delay-related QoS management.

2.2.3 Security management

Management of security is difficult to provide as WSNs are mainly made up of ad-hoc wireless networks with intermittent connectivity and resource limitations [8, 22]. This leads to vulnerability to threats that could be internal, external, malicious or even accidental. Data and resources can thus be stolen or modified and a denial of service attack is also possible. Security management, for example, has been approached through the use of key management schemes [23]. Reegan *et al*. [23] mention various requirements in managing security such as authentication, confidentiality, integrity, availability and authorisation. A network with high security features should be developed to meet these requirements however, the challenge here is the limited resource in bandwidth and energy inherent to WSNs [24, 25]. Management of security, therefore, has to be based on effective key distribution [23] taking into account features such as bandwidth, sensor memory, transmission range and the necessary know-how [26]. Several key management schemes have also been discussed by Reegan *et al*. [23]. Key management based on the dynamic clustering and optimal routing choice of the Mobile Sink has been proposed [27]. The scheme extends static key management to dynamic key management with an improved flexibility while satisfying storage efficiency and connectivity. A further contribution has been made by proposing a hybrid key management scheme which takes advantage of the Polynomial

Pool and Basic Random (PPBR) key pre-distribution techniques to improve the difficulty in cracking the key system [28]. To tackle issues related with link connectivity in key management, the tree base path key establishment method is used.

2.2.4 Maintenance management

Maintenance management as part of network management would enable higher QoS and an extended network lifetime. Faults in WSNs occur frequently due to interruptions in connectivity, energy shortages, changes in the environmental conditions and other network events expected and unexpected. The network, therefore, should be configured, managed and maintained in a manner that allows for self-healing and fault tolerance. This results in a robust characteristic enabling the WSN to provide useful information even when some nodes fall out of the network. Dayal and Kumar [29] have presented various fault mitigation techniques through the management of the network topology.

An important aspect of maintenance management and fault mitigation is network monitoring required to collect information on the state of the network (e.g., node energy levels, bandwidth and link states). Thus, interfaces and tools have been developed to provide a visual representation of the network state such as Mote-view [30], SNAMP [31], spy-glass [32], tinyviz [33], nanomon [34] among others. Lee *et al*. [18] mention that systems such as the mote-view and others that have a central management scheme with all analysis data being sent collectively to a central server have the disadvantage of being passive and lack flexibility in the configuration in an event of fault occurrence. This can be seen as leverage for SDN in solving this lack of flexibility, an aspect that becomes visible as management of WSNs based on SDN is explored in this chapter. An Energy-efficient Passive Monitoring System (EPMOSt) has been proposed taking into account the need for energy efficiency in WSNs [35]. Similar to WSNManagement [20] EPMOSt uses an SNMP agent to provide monitoring information with reduced energy consumption. This energy efficiency ensures longer network monitoring lifetime.

2.2.5 Comparison of WSN management systems

Notable WSN management systems discussed above are summarised in Table 2.1 and evaluated against the WSN design criteria. From the table we see that BOSS, mobile agent based power management,

WinMS, AppSleep and EpMOSt are promising management schemes meeting atleast three of the criteria required for an efficient management system with WinMs being most efficient.

2.3 NETWORK MANAGEMENT BASED ON SDN

The principle concept behind Software-Defined Networking (SDN) is based on the separation of the control plane from the data plane in the architecture of WSNs. This paradigm allows for a logically centralized controller which is a central program acting as the Network Operating System (NOS) thus controlling and managing the overall behaviour of the network. The network devices such as sensor nodes and switches exist in the data plane and simply forward data based on flow instructions provided by the controller. This approach to network management allows network configuration to be performed globally as opposed to a distributed management scheme which may require individual configuration of network devices to change network behaviour. The overall basic architecture is shown in Figure 2.1.

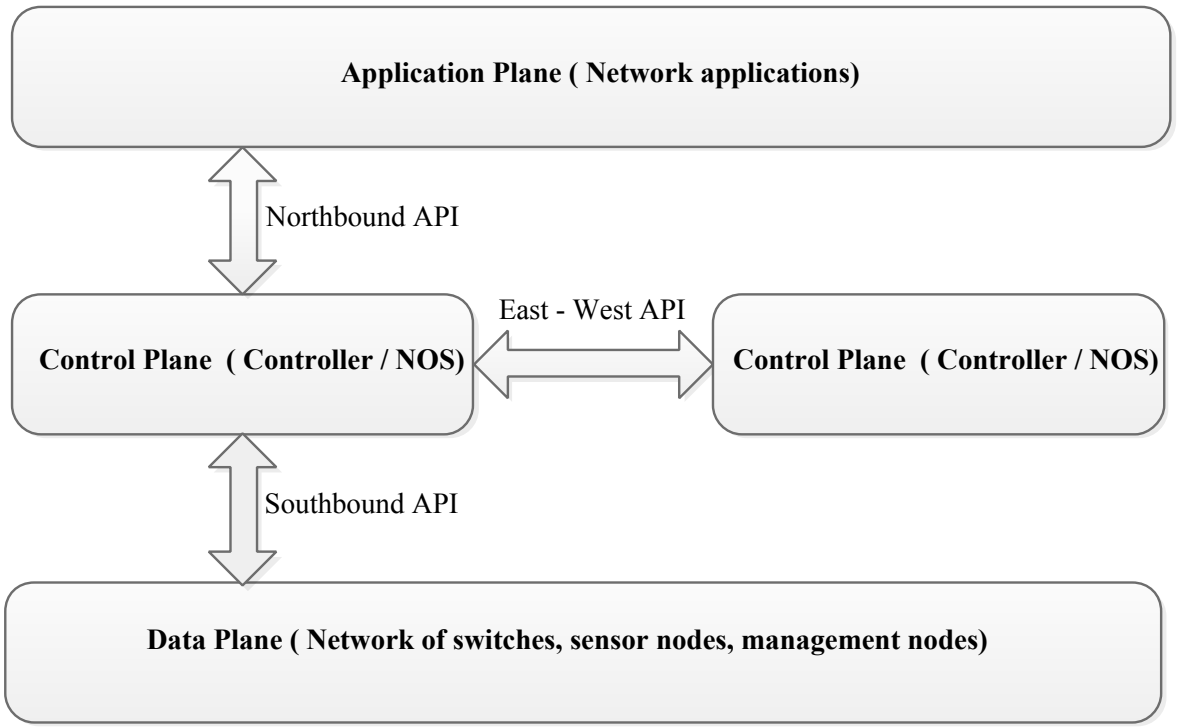

Figure 2.1. SDN overall architecture, adapted from [16].

Table 2.1. Available WSN management systems evaluation against the design criteria, taken from [16].

Management Scheme	Functionality	Energy Efficiency	Robustness	Scalability	Adaptability
MANNA [8]	Policy based framework	NA	NA	NA	NA
BOSS [7]	State retrieval, power management	Yes	Yes	No	Yes
Agilla [18]	Event detection	Yes	No	No	Yes
Sectoral Sweeper [18]	Switching node on/off	Yes	No	No	No
RRP [18]	Data aggregation	Yes	No	No	No
SNMS [18]	Health and event data collection	Yes	Yes	No	No
SNMP [18]	Network definition and monitoring	No	Yes	No	Yes
WSNManagement [20]	Performance and fault management	No	Yes	No	Yes
WinMS [18]	local repair and state retrieval	Yes	Yes	Yes	Yes
SenOS [18]	Triggering node on/off	Yes	No	No	No
AppSleep [18]	Power Management	Yes	Yes	Yes	No
EASA [36]	Self-sustaining energy management	Yes	Yes	No	No
MOTE-VIEW [30]	Network state and visualisation	Yes	No	Yes	No
EPMOSt [35]	Passive network monitoring	Yes	Yes	No	Yes

The SDN architecture integrates Application Programming Interfaces (APIs) which provide a working interface between the application, control and data planes. Between the application plane and the control plane are the northbound APIs and between the controller and the data plane exist the southbound APIs. The southbound APIs facilitate the flow of control traffic from the controller to the devices in the data plane. OpenFlow [37–39] is a standard commonly used as a southbound interface [40] and most vendors are producing more OpenFlow capable network infrastructure. East-West APIs enable controllers that are in the same domain or neighbouring domains to communicate with each other [41]. It is important to mention here that SDN is not about improving the performance of the network but rather SDN has been seen to have the potential to simplify network management and allow for innovation through network programmability. Updates to network control and management are made as easy as installing new applications on a computer [42]. SDN changes a network administration problem into a programming problem increasing flexibility, other potential benefits of SDN in WSNs [43] include:

a. Energy management: WSN nodes are energy constrained and thus need energy-efficient protocols to be employed in sensor networks. SDN can provide an energy efficient way for network management of WSNs. This is possible as having a logically centralized controller maintains a complete view of the entire WSN and thus can reduce the power consumed by nodes in maintaining that view locally. Control plane functions can manage all the routing protocols saving nodes from following application-specific protocols which might drain energy depending on traffic. Alternative energy management techniques can also be employed in the control plane.
b. Configuration management: The complexity of network management can be simplified due to the increased flexibility of SDN. With SDN new routing protocols can be employed easily without reconfiguring the nodes and also it reduces the need for compiling different versions for the same network applications for different sensor nodes. Thus, if a new management and control method becomes available, it can easily be deployed resulting in an efficiency of operation.

Network management with different vendor-specific network hardware can be challenging [40]. Network operators need to adjust to low-level vendor specific hardware configurations to implement complex high-level network management policies. Network management schemes such as those discussed in the previous section encounter difficulties in changing the policies embedded in the underlying network infrastructure. The direct result is the existence of application-specific networks with few possibilities for improvement and innovation. New possibilities in the management of a

network and configuration therein can be introduced by SDN. SDN helps solve some of the problems of network management by enabling easy and frequent changes to the network function and state. It allows configuration of the network in high-level languages that provides visibility and control over the entire network. Such a feature makes it easier to troubleshoot problems in the network.

In traditional wired/wireless networks, network operators have to implement and configure networks using a set of low-level device commands in the command line interface [40]. This is especially a problem with constantly varying network conditions. Operators have to manually adjust configurations frequently to meet these changes leaving room for misconfiguration and mistakes in implementing changes [44]. With SDN it is much easier to integrate new protocols and ideas into the network through the use of a centralized control. SDN introduces a global approach to the management of network configuration rather than local management which would require configuration for each of the network devices to implement changes or integrate new services. Several works have been researched and proposed on the potential benefits of implementing SDN in both wired and wireless networks [9, 37, 43, 45, 46]. Kim and Feamster [40] have designed and implemented Procera which is an event-driven control framework for a network based on SDN. Procera focuses on network management by allowing operators to implement high-level policies which are translated into a set of forwarding rules. These rules are then used to enforce policies on the network hardware using OpenFlow [37–39, 47]. This management process takes place at the policy layer of SDN which exits as part of the northbound API. There has been numerous prior research on applying SDN in the management of computer networks leaving room for study on how SDN can facilitate and enhance management of IoT. An SDN-based management architecture for IoT with a focus on Machine to Machine (M2M) infrastructure has been proposed by Huang *et al*. [48]. The proposed framework consists of a centralized controller, M2M nodes, a gateway linking non-M2M protocol general nodes to the controller. The framework was tested successfully for routing reconfiguration without the need to deploy new infrastructure.

2.4 WSN APPLICATIONS AND THE NEED FOR SDN-BASED MANAGEMENT

Wireless sensor networks are able to integrate various sensing capabilities thus providing support for various real-world applications. This flexibility is accompanied by several research challenges in providing effective management for the application considering the resource-constrained nature of

sensor nodes. The various benefits that SDN introduces to these management challenges allow it to be utilized in several applications such as in environmental applications, health care, military and home networks. This section will briefly discuss these application scenarios and bring to light the benefits of SDN-based management in these areas.

2.4.1 Environmental applications

A popular environmental application of WSNs is based on the measurement of physical parameters such as temperature, vibration, sound, chemicals and gases. This has led to application in scenarios such as the agriculture industry for irrigation or animal monitoring, water industry for the development of smart water grids to enable efficient distribution and control of potable water [2] and in disaster relief applications for monitoring and fighting forest fires and also in earthquake detection and rescue operations. Other applications include environmental control of pollutants and marine monitoring requiring long term unmanned WSN operations. However, such applications may require the deployment of sensors over very large areas, harsh environments or even hard to reach regions. This may result in dependence on sensor energy supply which affects network lifetime and the ability to alternate tasks upon change of sensing requirement or upon node failure [49]. There is, therefore, a need for effective topology, energy and maintenance management of which integration of SDN introduces to the application scenarios. SDN-based management would allow for a centralized network view of essential parameters suitable for large-scale applications. In situations where energy is crucial, SDN techniques can provide effective resource allocation through software duty-cycling and by enabling easier use of traffic engineering, congestion control, load balancing [50] and mobility management. SDN also provides an effective interface for sensor re-programmability necessary for function alternation without requiring additional hardware such as Field Programmable Gate Arrays (FPGAs) [51].

2.4.2 Medicine and health care

WSN application in health care has been considered to have the potential of being beneficial to the quality of health, especially for the chronically ill and elderly patients. WSNs can also be used for tracking of patients and doctors in hospitals which can prove useful in saving lives. A controversial application of WSNs in health care is the use of implantable sensor systems to monitor patient activities resulting in Wireless Body Area Networks (WBANs). Darwish *et al*. [52] identify challenges to

implementing WBANs among which are sensor maintenance once body-worn, energy management for sensor batteries considering that charging would pose a challenge especially for the elderly as they have to remember to charge multiple sensors, insufficient bandwidth resource to allow for transmission of large amounts of medical diagnostic data, meeting the QoS requirements of health monitoring hindered by WSN resource constraints, reliability of critical medical data packets being delivered, security and privacy of medical data, scalability and sensor mobility challenges as patients move about in their daily life. SDN-based management would provide a solution to several of these highlighted challenges based on the ability to provide energy and maintenance management from a central location which could be a task of hospital administration for instance. Furthermore, SDN techniques are being developed to reduce traffic overhead data to mitigate bandwidth problems [53, 54] and also to ensure the accuracy of data and fault monitoring more efficiently. Considering the possibility of implanting multiple body sensors in close proximity which may result in interference affecting the QoS and application performance, SDN provides a solution based on the separation of destructive interfering nodes. Security and privacy is a key issue in health applications and SDN provides several encryption algorithms that can be delivered on demand, the encryption techniques available are discussed later in this chapter.

2.4.3 Military applications

In military applications, WSNs can be used to detect the presence of the enemy or friendly forces, assessment of terrain and chemical sensing of weapons. Fraga-Lamas *et al*. [55] reviewed the possibility of applying IoT to warfare to increase efficiency. Uses of WSNs for defence IoT included application in fire control, inventory tracking, fleet monitoring, energy management and management of the health and safety of troops. However, to be suitable for use in defence and security WSNs need to meet operational requirements such as robustness, ease of deployment, interoperability/adaptability and most importantly security. SDN can be leveraged to better achieve these requirements as it maintains a global control of the network infrastructure in the field which better manages system security and issuance of new task protocols during operation. SDN-based management also provides an opportunity to easier deploy mobility algorithms to handle the constant movement of troops as these algorithms can be configured and managed in a more resource capable global controller.

2.4.4 Wireless home networks

WSNs for home use have seen increasing demands with arising smart homes and buildings in general. Homes are now composed of a network of light, motion and temperature sensor nodes that can be integrated into appliances to introduce smart capabilities in a home. This network can also be linked to the internet to allow users to remotely control these home appliances. However, in addition to a network of smart devices is a network of other communication technologies such as WiFi, ethernet, cellular and power-line communications. The challenge with such a heterogeneous network is the lack of an automated system to enable network optimization by selecting the best communication technology to use, this decision is usually made by the user manually. Soetens *et al*. [56] have proposed an SDN-based management technique to improve the management of such heterogeneous home networks by using OpenFlow-enabled link switches. SDN techniques have also been proposed to enhance the development and management of smart homes [4]. Smart device use is increasing in homes and this creates difficulty in management especially with the advent of video streaming services such as Youtube and Netflix. User demand and control becomes an important management factor leading to the need for smart home network management based on SDN. In addition, these user applications such as voice, video, gaming and others have to compete for bandwidth resulting in a poor quality of service. Kumar *et al*. [57] have proposed and experimented on an SDN-based solution that allows users some control over the service quality for their devices enabled by Internet Service Providers (ISPs). Other bandwidth allocation techniques for managing home applications like video and voice have been proposed based on SDN to improve quality of service including home slice [58], Network Assisted Video Streaming (NAVS) [59] and SDN@home [60].

2.5 MANAGEMENT OF WSNS BASED ON SDN

WSNs are composed of a network of sensors that are deployed to measure parameters such as temperature, pressure and air quality. The sensor network may exist as hundreds or even thousands of resource constrained nodes. Other inherent challenges of WSNs include energy requirement which is directly linked to network lifetime, the application-specific nature with vertical integration [40] where manufacturers and vendors control the end products of the devices as well as the software and hardware of their infrastructure. Another challenge introduced as the nodes are deployed in the field is the management of the network topology [61] where we have to consider the coverage of the

network and quality of service in an environment with varying conditions at a limited energy supply. Integration of SDN in WSNs has been proposed [9, 37, 43] to mitigate these challenges. Integration of SDN would require consideration of how the network will be managed to ensure efficient performance and resource allocation. The WSN architecture based on SDN can be divided into managerial entities similar to the management functionality abstractions as mentioned by Ruiz *et al*. [8]. Figure 2.2 shows this architecture [46] adapted with a further focus on the management entities. The architecture shown depicts an SDN-based WSN with a centralized controller thus providing a global view of the entire network resulting in efficient management. Management of the network topology, energy, configuration, QoS and security are done centrally in the control plane while management of enabling technologies can be implemented at all levels of the architecture. In the data plane enabling hardware such as SDN-enabled sensor nodes with multiple sensing capabilities and hardware capabilities can be installed while programming of the nodes for various applications can be done and managed centrally in the control plane.

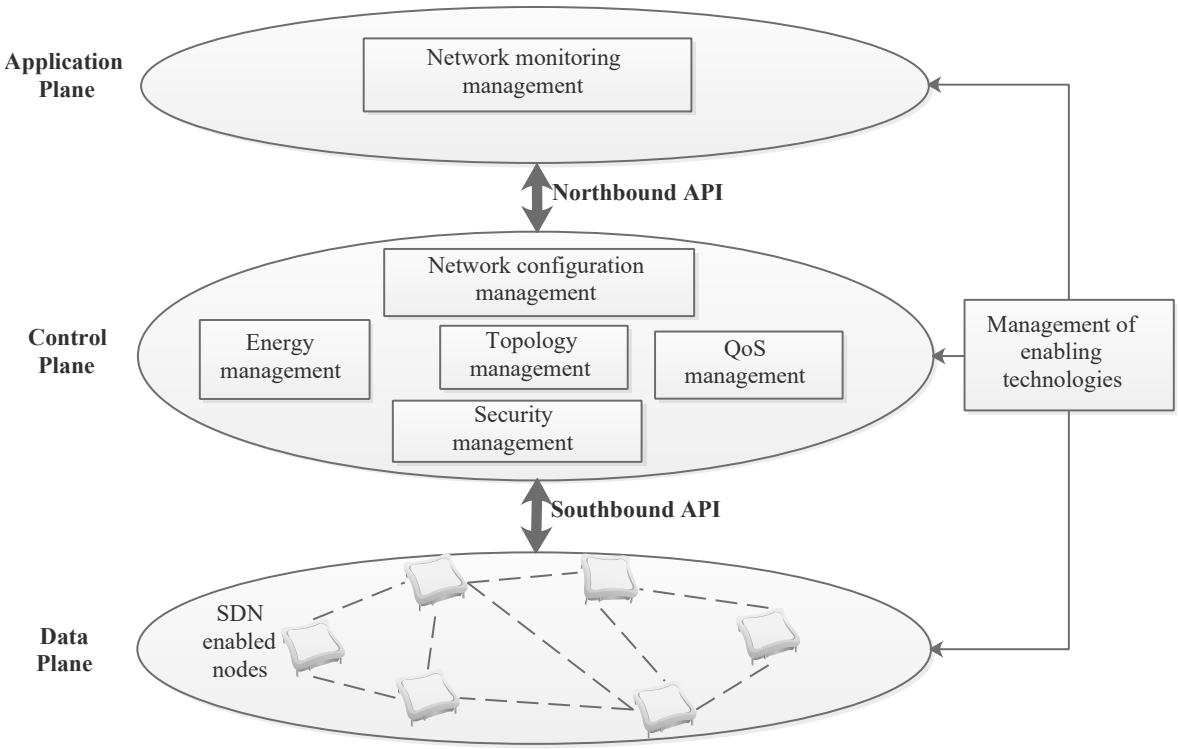

Figure 2.2. Overview of an SDN-based wireless sensor network management architecture, adapted from [46], © 2016 IEEE.

2.5.1 Network configuration management

Network configuration management is a functional area that will allow the sensor network to meet its required specification. In an SDN-based WSN architecture, the first challenge with meeting the requirements has to do with SDN support for the WSN architecture [46]. SDN was designed for conventional networks which are different from WSNs and therefore it is necessary to configure the network to enable SDN support. This is particularly important in harnessing existing network management protocols and architectures which can provide the SDN-based WSN with a proper network management model [10]. Several proposals have been made towards configuring and extending SDN functionality for wireless sensor applications. Sensor OpenFlow (SOF) [37, 38] is one such proposal which serves as a standard communication protocol between the data plane and control plane and is a core component of WSNs based on SDN. The idea is based on the data plane that is composed of sensors performing flow-based packet forwarding and the control plane consisting of a controller handling all the network intelligence at a central place (centralized control). This allows the data plane to be fully programmable by customizing the flow table of each sensor. The challenge with this concept is that it is built on the fundamental OpenFlow assumption that the underlying network is made of ethernet switches and IP routers which are wired protocols useful in cellular and enterprise backbones and data centres; and not so much in WSNs. Moreover, using the centralized controller would raise questions on the management of security in an event of attacks directed towards the controller [43]. Thus, adequate network security management has to be in place.

Software-Defined Wireless Network (SDWN) is yet another solution that has been prototyped to extend SDN functionality in wireless and mobile communications specifically providing support for SDN in Low Rate Wireless Personal Area Networks (LR-WPANs) [42]. SDWN uses energy management techniques such as duty cycling, in-network data aggregation and cross-layer optimization to reduce energy consumption. However, adapting the SDWN architecture to integrate and configure SDN for WSNs for efficient energy use needs further investigation. Mainly because SDWN requires messages to be passed between control and data layers which may strain the already resource-constrained WSN.

Galluccio *et al*. [12] have proposed SDN-WISE (SDN-WIreless SEnsor network), a technique which makes sensor nodes programmable as finite state machines so as to enable them to run applications that cannot be supported by stateless solutions. It provides an API which allows developers to program the

SDN controller in a language of their choice enabling simplicity and flexibility in the management of network programming. This is especially useful when managing a large-scale and long-range wireless sensor network.

Configuration of SDN-based WSNs should allow the SDN protocols which are address-centric to address the inherently data-centric WSN. A solution [37] to this would be integrating the use of IP stacks such as uIP, uIPv6 [62, 63] based on Contiki OS [64], Berkley Low Power IP (BLIP) based on Tiny OS [65, 66] for low power networks. As a result of this integration, IPv6 based Low-power Wireless Personal Area Networks (6LoWPAN) have been realised. Management architectures based on the 6LoWPAN protocol have thus been proposed including the LoWPAN network management of the existing SNMP [18] protocol and a further network management architecture based on SNMP for 6LoWPANs proposed by Feng *et al*. [67]. These management architectures although used for various wireless networks provide a model for developing a proper network management architecture for the WSNs based on SDN [10].

Another aspect of network configuration includes management of the flow of control and data traffic which uses the same network path (in-band) in WSNs unlike in wired SDN applications where a dedicated control channel exists for communication between the controller and the data plane. This would pose a challenge for WSN control data if not managed due to variations in connectivity from link and node failures, mobility, routing and other similar network parameter variations. In line with this, proposals have been made to reduce control traffic as a management solution, Luo *et al*. [68] for instance propose a method of only sending a single packet meant for a flow setup request to the controller when a table mismatch occurs. This is enabled by suppressing the proceeding packet requests from the associated packet having the same destination address as the first packet until a corresponding request is received from the controller or when a set expiry time is reached.

2.5.2 Topology management

WSNs are different from the conventional networks that SDN was initially designed for. WSNs have limited resources and use the same channel for the node to node communication and data processing (in-network processing). SDN allows for decoupling of control plane and data plane however, for WSNs the control and data packets would still use the already resource-constrained and dynamic WSN

topology. In the management of the network topology, the technique employed should be able to support node mobility which results in topology changes including dealing with unreliable wireless links. These functions should be in tandem with the management of QoS which ensures that the network is robust. WSN topology management based on SDN can be divided into the management of the following:

(a) Scalability and localization management: The architectures reviewed so far are based on a centralized controller in a flat topology which is easier to deploy and manage. Gante *et al*. [43] for instance introduce smart management of SDWSNs to improve efficiency and overcome the inherent difficulties of ordinary WSNs. The management scheme is based on a proposed base station architecture for WSNs with an integrated controller. The controller creates forwarding rules that are placed in flow tables from location data obtained through localization techniques processed in the application layer of the architecture. However, flat topologies like this are limited to short range and small scale applications and as networks scale up to very large numbers of nodes need arises to integrate topologies that are hierarchical or provide a virtual overlay of the physical network [69] to support efficient scalability and localization. A few architectures are available for network management designed towards improving SDN-based WSN management efficiency by using distributed multiple controllers rather than a single centralized controller. This also enables efficient scalability and a reduction in overhead control data as it is not necessary to communicate all control data centrally. Distributed controllers also allow for effective security and QoS management as the WSN is not solely dependent on one controller, making the system more reliable during attacks or failures in the link to the central controller.

A hierarchical architecture called Software-Defined Clustered Sensor Networks (SDCSN) has been proposed [70] and uses multiple base stations as controllers that also play the role of cluster heads. A large-scale group of nodes is divided into clusters and there is a cluster head for each. The cluster head controls and coordinates the sensor nodes in each cluster and all the information processed in each cluster is routed to the cluster head. Management of such an architecture requires enabling the sensor nodes to function as controllers effectively enabling multiple controllers in the network. Oliveira *et al*. [54] design and implement an architecture based on multiple controllers within a WSN in a framework called TinySDN based on Tiny-OS with a design structure that consists of SDN-enabled sensor nodes and an SDN controller node. This addresses issues such as in-band control, higher communication latency, and limited energy supply. The downside to this management approach is that the cluster head can also become vulnerable to attack posing a network security risk [69] however self-

stabilisation techniques for re-selecting a new cluster head upon such an event have been proposed [71]. To manage the network with more flexibility for integration of various functions an architecture based on network virtualisation has been proposed [41, 69, 72]. Basically, a virtual overlay topology can be built to certain specifications based on an actual physical WSN integrating SDN [69] as shown in Figure 2.3.

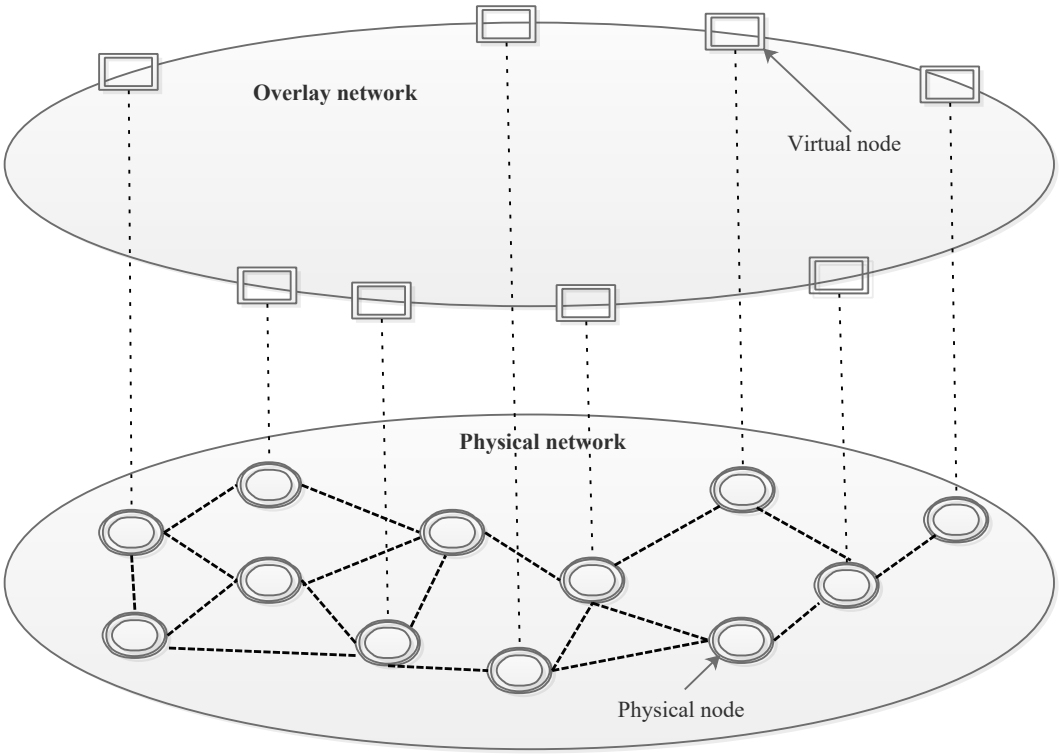

Figure 2.3. Topology based on virtual overlay, taken from [69], © 2016 IEEE.

Another hierarchical architecture proposed to manage scalability is context based logical clustering [73, 74]. This is based on logically clustering sensors according to context (gathered data type) regardless of their physical distribution based on HyperFlow [75] unlike other clustering methods proposed which allow clustering of adjacent nodes. Clustering sensors according to context would allow for data and resource sharing regardless of network expansion. Each cluster has a local controller which may be physically distributed but logically synchronised hence referred to as logical sink in the virtual network overlay. The performance study shows that this technique improves network stability effectively [74].

Sensor data without location information is not useful especially in large-scale applications with hundreds to thousands of nodes. SDN in WSNs provides the possibility of obtaining location in-

formation with a good level of accuracy by implementing proposed localization algorithms [76–81]. The localization algorithm to be implemented and managed can either be centralized or distributed depending on the mapping information and level of accuracy provided by the algorithm. More recently a localization node selection algorithm based on the SDN and the Cramer-Rao Lower Bound (CRLB) metric has been investigated and proposed [82]. The algorithm makes use of the global network view that the SDN controller has while satisfying the need for energy conservation and the simulation results presented a significant improvement in the localization performance.

(b) Mobility management: Sensor nodes in an SDN-based WSN are susceptible to movement and this can cause variation in packet transmission and execution of tasks making it necessary to monitor and manage the movement of nodes in the network. The challenges to consider when integrating a mobility management scheme includes handling the effect of nodes entering the network and the nodes leaving the network on QoS, execution of functions and other network attributes. A few solutions and processing steps that the SDN controller can provide to prevent problems associated with nodes entering and leaving the WSN have been outlined by Zhou *et al*. [69]. Considering IP support for nodes in SDN-based WSNs [62, 65] research has been conducted on mobility management in IP based WSNs based on TinyOS implemented and tested on a 6LoWPAN [83]. A mechanism based on IEEE 802.15.4 [84] standard has been investigated and implemented to manage mobility in a distributed network architecture with a promising implementation capacity in a real system [85]. There have been several contributions on mobility management in Wireless Cellular Networks (WCNs) based on SDN [86–89], whether or not these management mechanisms can be applied and integrated into SDN-based WSNs is a worthwhile research area.

(c) Communication management: Integrating SDN in the WSN topology enables the management and operation of a network with sensors of different architectures and manufacturers with little or no challenge with compatibility issues. These nodes can be densely located with neighbouring sensor nodes appearing very close to each other and hence the need for an efficient communication scheme to be deployed. The communication scheme should take into account energy consumption and signal quality however, the primary focus should be on conservation of energy to improve network lifetime. Multi-hop techniques based on short-range radio communication such as Meshlogic and ZigBee [24] have been proposed [90] for dense WSNs enabling consumption of less energy compared to direct communication. In SDN applications employing clustered hierarchical topologies, long-range radio can be used for node to cluster head communication. This would allow for continuity in network operation even when a few nodes in a cluster fall out due to the direct communication between the cluster head (SDN local controller) and each node. The proposed radio technology to be implemented

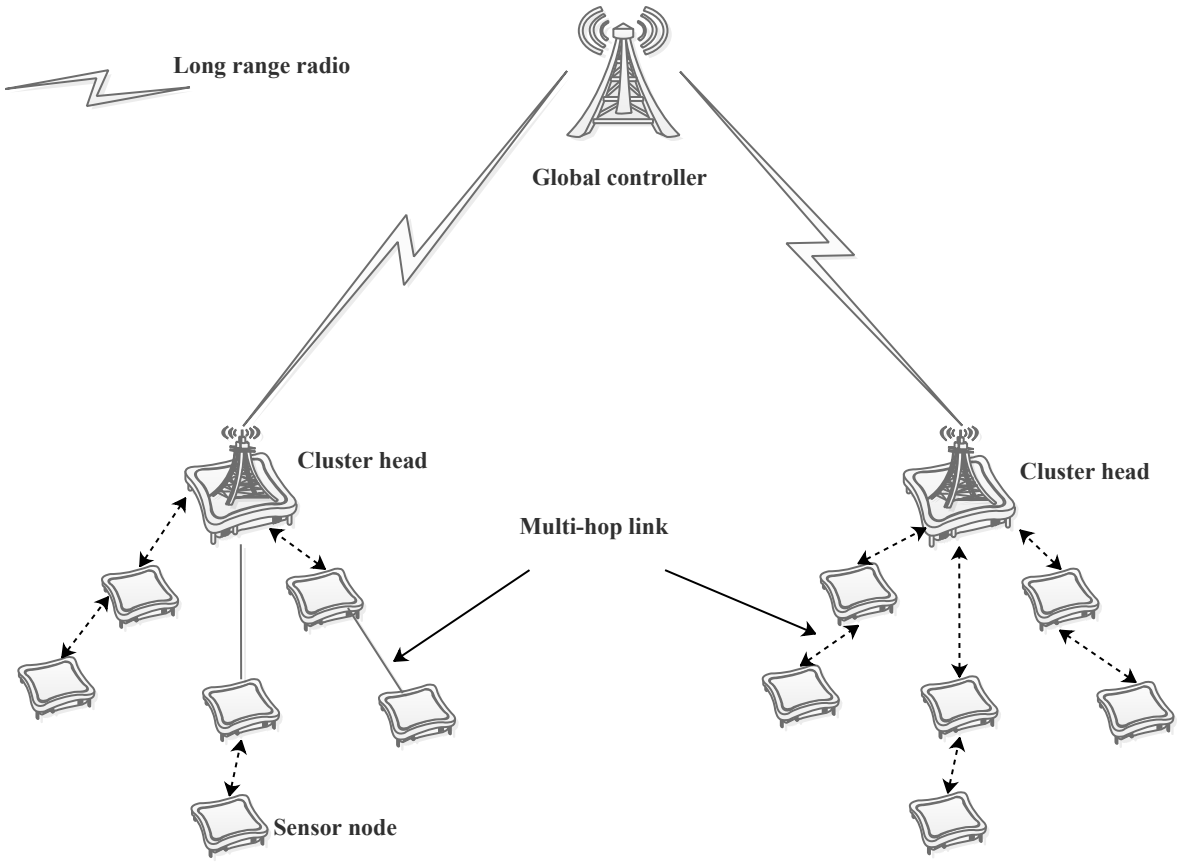

Figure 2.4. Hybrid radio management, taken from [16].

in such scenarios should also require less power such as the LoRa technology intended for Low Power Wide Area Networks (LPWAN) [91, 92]. A hybrid of multi-hop techniques and long-range radio can be used to manage a large low power WSN based on SDN where multi-hop communication is implemented between nodes in a cluster and long-range radio is used between cluster heads and the global controller as shown in Figure 2.4.

(d) Network monitoring: To improve manageability and ease of implementing a WSN based on SDN, network monitoring is essential as it enables the display of all necessary sensor information such as details of the sensor node hardware, software, and battery level at the management station [93]. In SDN, network monitoring applications have been developed based on the OpenFlow capability to provide a mechanism for the collection of flow information and statistics. These applications through the controller are able to query nodes from time to time. The accuracy of the monitored information is dependent on the frequency of these queries. Techniques such as Payless [94] have been proposed to improve the efficiency of this query process on monitoring accuracy and network overhead. Other proposed network monitoring tools based on this concept for traditional software-defined IP networks

include OpenSketch [95], OpenTM [96], FlowSense [97] and OpenNetMon [98]. There has been little effort in investigating these techniques or new techniques for monitoring SDN-based WSNs and it is still an open research area. However, work has been presented on a network measurement architecture based on SDN for monitoring of WSN information such as routing path per-packet, the loss ratio for each link and the delay resulting from each hop in a multi-hop configuration. TinySDM [99] a software-defined measurement architecture built on the TinyOS [66] platform enables easy customisation and execution of various measurement metrics and allows for quick and efficient implementation of measurement tasks.

(e) SDN-based network and topology management classification for WSNs: Classification criteria is an important factor to consider when integrating a management system into a network. Choice of the management system would depend on the management function, controller configuration to which the architecture is suited and control traffic channel used. Table 2.2 summarises and compares the architectures available in network and topology management of WSNs based on SDN against classification criteria.

2.5.3 Energy management

Energy management focuses on the several techniques available to leverage SDN in ensuring that energy utilization is minimized in an already energy resource-constrained WSN. Several proposals for adopting energy management in the SDN controller for the global balance of energy utilized in the WSN have been made. Integration SDN in WSNs allows for energy consuming tasks such as data traffic routing decisions to be handled in the controller leaving the energy constrained sensors only with the task of forwarding data traffic inherently reducing the amount of energy consumed. Part of the topology management of WSNs based on SDN is focused on using SDN for resource allocation control. To enhance resource allocation, architectures and topologies likewise have been proposed such as the SDWN [42] and Smart [43] both executing controller functions on the sink node freeing remaining sensor nodes from functions like localization, QoS and mobility management resulting in energy efficient operation. Similarly, a general framework for has been proposed that places the controller at the base station with centre nodes in each cluster integrating switching and communication functions with the controller based on OpenFlow [100].

Management of energy has also been demonstrated by establishing control through software in a

Table 2.2. SDN-based network and topology management architectures, taken from [16].

Management architecture	Management feature	Controller configuration	Control traffic channel	Configuration & Monitoring	Scalability & Localization
Sensor OpenFlow [37, 38]	SDN support protocol	Distributed and Centralized	in-band and out-band	✓	
SDWN [42]	Duty cycling, aggregation and routing	Distributed	in-band	✓	
SDN-WISE [12]	programming simplicity, aggregation	Distributed	in-band	✓	
Smart [43]	Efficiency in resource allocation	Centralized	in-band		✓
SDCSN [70]	Network reliability and QoS	Distributed	in-band		✓
TinySDN [54]	in-band traffic control	Distributed	in-band		✓
Virtual Overlay [41, 69, 72]	network flexibility	Distributed	in-band		✓
CRLB [82]	node localization	Centralized	in-band		✓
TinySDM [99]	network task measurement	-	in-band	✓	

method referred to as duty cycling, focusing on reducing the amount of energy consumed by the sensor nodes through control of the sleep and wake up states. This technique has been proposed severally for managing WSNs [101–103] however SDN-based architectures rely on cross-layer communication

between the decoupled control and data planes. Such a setup may require a robust link for the control traffic flow to be optimized in addition to the need for energy saving through data aggregation and thus using techniques that switch nodes on and off may result in unreliability in the communication links. Multi-tasking of WSNs has been proposed as an architecture based on SDN to minimize energy utilization in [104]. Multi-tasking is possible through the ability of the sensor node to carry out multiple tasks at the same time achieving different sensing outputs and to achieve a desired Quality of Sensing (QoSen) a task is carried out collectively by multiple nodes. Zeng *et al*. [104] present an architecture allowing for energy efficiency in sensor scheduling and management while producing the required QoSen. A Mixed Integer Linear Program (MILP) is formulated to take into account three objectives: sensor activation, task mapping and sensing scheduling. The authors have also developed an on-line algorithm to handle the dynamics associated with nodes entering the network (mobility) or leaving the network (mobility or node failure) with simulation results showing equal energy efficiency as a global optimized network with the advantage of having a lower rescheduling time and overhead control data.

Inherent in the SDN-based management is a high control traffic overhead to maintain the global view of the logically centralized controller; a process that requires more energy for operation and thus affecting the energy efficiency of the network. Huang *et al*. [105] propose a prototype based on reinforcement learning to improve the energy efficiency of SDN-based WSNs. The technique reduces overhead control traffic by filtering redundancy and performing a load-balancing routing mechanism in accordance with data flow distribution with the required QoS. Experiments on the proposed SDN-based WSN prototype showed an improved energy efficiency performance.

Energy management techniques such as data aggregation and information fusion [106] coupled with multi-hop communication used in WSNs [107] can be extended and applied to within each cluster. Multi-hop links can take advantage of the densely located sensor nodes to transmit data at low power [90]. More recently, an efficient energy harvesting management technique based on wireless power transfer for SDN-based WSNs has been proposed [108]. The mechanism uses dedicated wireless power transmitters to transmit energy to sensor nodes in the network thus increasing the network lifetime. A utility function is defined to manage the energy distribution and the maximum energy required to charge the nodes to enable maximum optimization while meeting the minimum energy required by each sensor node. Ejaz *et al*. [108] also propose an accompanying energy efficient scheduling system for the energy transmitters during each charge process. Table 2.3 compares the

energy management schemes available for SDN-based WSNs.

Table 2.3. SDN-based energy management schemes for WSNs, taken from [16].

Scheme	Features	Controller configuration	Enabling technology
SDWN [42]	Duty cycling, data aggregation	Distributed	Software and hardware
Smart [43]	Resource allocation	Centralized	Software and hardware
Multi-task [104]	Resource allocation, QoSen, scheduling	Centralized	Software
SDWSN-RL [105]	Load balancing, traffic control	Distributed	Software
Wireless power transfer [108]	Energy harvest, Optimization, efficiency	Centralized	Software and hardware

2.5.4 QoS management

The quality of service management covers all aspects of ensuring an acceptable delivery of service in a network. This includes management of system tolerance and reliability, packet latency, event detection, robustness and maintenance. The SDN architecture allows for the QoS management tasks to take place in the controller by virtue of being logically centralized. The SDN-based WSN environment varies due to factors such as topology changes, node failure, bandwidth availability and energy fluctuations therefore, there is a need to integrate a mechanism for the reliability of the network. Lu *et al.* [109] study the reliability of the WSN architecture for SDN through the use of models based on continuous time Markov chain and continuous stochastic logic techniques. The results show that network reliability is affected by the number of controllers and sensors and their respective failure rates. From the results, a proposal to integrate more than one reliable controller as a reliability management strategy is made.

Management of reliability has been classified into the reliability of communication systems and the reliability of tasks [69]. Management of communication reliability can be done through methods such as providing back up nodes and redundant links [69]. Function alternation has been proposed to manage the reliability of sensor tasks by enabling neighbouring sensor nodes to take over damaged sensor node tasks automatically providing a self-healing mechanism [49]. The SDN controller can also provide back-up for sensing tasks and information to improve the reliability of tasks [69]. The back-up information can be used for re-tasking neighbouring sensor nodes to take up the task of damaged or unreliable nodes in SDN-based WSNs. The SDN OpenFlow concept has been demonstrated and exploited to address issues of performance reliability in WSNs [110]. The idea presented is based on the inherent characteristic of OpenFlow to control and monitor sensor traffic enabling a reliable routing and load balancing mechanism.

In ordinary WSNs the reliability of multi-hop and end to end communications is still an unsolved and difficult issue to manage and hence flow-sensor has been proposed to produce better reliability as compared to typical sensors by generating a lower amount of packets reducing the traffic overhead in constrained networks [110]. Another QoS parameter is a robustness which is a characteristic that defines a system's ability to meet its expected performance [9] under unreliable environmental conditions. To manage system robustness it is necessary to consider factors affecting and interfering with the system and monitor the interference patterns in a proposed statistical machine learning technique [9]. This technique would allow for the prevention of possible interference based on previous behavioural patterns of interference. In managing latency TinySDN [54] has been proposed however, there is a need for further research on managing latency and event detection.

2.5.5 Security management

The architecture of WSNs based on SDN is such that a logically centralized controller poses a security risk as any compromise in the controller, cluster-head in clustered topologies or even the link to the controller can affect the system QoS. A malicious attacker can introduce false data and parameters in the network or a Denial of Service (DoS) resulting in system unreliability. A few contributions have been made in managing security for SDN-based WSNs, Flauzac *et al*. [111] for example describe the need for security management in both wired and wireless networks based on SDN for the internet of things and propose a security model for IoT SDN architecture. An extension of the proposed architecture

for IoT is made to include sensors integrating the aspect of guaranteeing the security of the network based on the grid of security concept embedded in each controller. The grid of security concept has been presented previously as an approach to securing networks by ensuring that communication takes place between trustworthy devices regardless of system policy control supported by a communication model to help structure the service distribution of security among nodes [112]. The overall concept is to decentralize the management of the security of the network ensuring continuity of security during failure.

Another security management contribution has been the integration of Secure Multiparty Computation (SMC) to secure private sensory data in SDN-based WSNs [113]. The focus is the application of SMC in securely processing sensory data between the SDN-based WSN and the web server. Sun *et al*. [113] present a security model based on the SMC protocol allowing clients in the SDN-based WSN to connect and disconnect to the web server arbitrarily. A lottery SMC protocol is constructed for the performing selection in SDN-based WSNs with encryption based on the layered homophobic function.

2.5.6 Management of enabling technologies

2.5.6.1 Enabling software

A major part of WSNs based on SDN is software, which enables the management and control of sensor nodes and also forms the basis for sensor operations. The SDN-based architecture allows for the management and control of enabling software taking place at the application, control and data planes. The manner of operation enables application programmers to adjust the sensor functionalities by invoking different programs making the sensor node behave like a tiny computer. This also calls for the sensor nodes in the data plane to have an operating system however, a less complex one compared to personal computers because sensor nodes have limited processing power and memory [45]. There are several sensor operating systems that enable the integration of sensors into the architecture of an SDN-based WSN. Choice of the operating system affects the memory management and protection of sensor nodes with memory management being classified into either static or dynamic memory management. Static memory management is fixed and simple hence useful for constrained memory nodes but is not flexible due to the unavailability of allocating memory at run-time. Dynamic memory

management, on the other hand, is flexible as it provides for allocation and de-allocation of memory at run-time [114].

SDN-based WSNs provide for multiple executions of functions thus the need for memory protection to prevent the mix up of process address spaces. TinyOS [66, 115] written in nesC has been structured for low-power wireless sensors and embedded systems that have limited resources. TinyOS provides for static memory management with memory protection. The use of TinyOS in WSNs based on SDN has been demonstrated in [54, 83, 99]. Contiki [64] has been presented as a lightweight operating system with support for dynamic network loading and redefinition of node programs. Feasibility of dynamic loading and offloading of programs in a resource-constrained network is demonstrated [64] based on Contiki. Contiki supports dynamic memory management and memory block functions but does not provide a management mechanism for memory protection between function executions [114]. Other available operating systems include multimodal system for networks of in-situ wireless sensors (MANTIS), Nano-RK and LiteOS [114] however there is little or no implementation of these operating systems in SDN-based management of WSNs but rather for ordinary WSNs. Table 2.4 compares the available sensor node operating systems for integration in SDN-based management of WSNs.

Table 2.4. Enabling node operating systems: A comparison, taken from [16].

OS	Language	Memory management	Implementation
TinyOS	NesC	Static	TinySDN [54], TinySDM [99], mobility [83]
Contiki	C	Dynamic	SDN-WISE [12]
MANTIS	C	Dynamic	-
Nano-RK	C	Static	-
LiteOS	Lite C++	Dynamic	-

An important feature of SDN in WSNs is to provide a global view of sensor nodes allowing for efficient program delivery when forwarding control data or updating node software for a new task. The architecture hence allows for this program delivery through Over The Air Programming (OTAP) techniques which contribute to the efficient management of enabling the software. Use of OTAP in downloading of roles to nodes via a multi-hop wireless link in a reconfigurable SDWSN has been

demonstrated in [116]. Efficiency in OTAP has been further demonstrated by the use of SDN-WISE, an SDN solution for WSNs. A hands-on demonstration has been done using SDN-WISE to reprogramme WSNs allowing for various logical sensor networks to exist together while using the same node hardware [117].

2.5.6.2 Enabling hardware

Enabling hardware forms part of the infrastructure of WSNs based on SDN and an important component for managing and enabling the SDN integration into WSNs is the SDN-enabled sensor node. The design and management of the nodes need to meet the requirements of SDN and a major part of the design is that the node should be reconfigurable even after deployment via wireless communications. Examples of reconfigurable nodes presented for WSNs include a sensor node based on modularity [118] using the Field Programmable Gate Array (FPGA) as the node control unit [119] due to the re-configurable property of FPGA hardware, unlike micro-controllers. This enables the remote reconfiguration of sensor hardware over wireless networks which is essential for SDN-based management of WSNs. To reduce the energy consumption of FPGAs, state of the art designs have been developed to operate in ultra-low power modes [45]. Another implementation of sensor reconfigurability is a node design with a soft core processor and the capacity to be reconfigured remotely via OTAP techniques post-deployment based on the FPGA as the control unit [120]. The systems proposed in [119, 120] use a micro-controller to re-configure the FPGA as a minimisation technique for power consumption.

Further efforts in proposing implementations of reconfigurable nodes (SDN-enabled nodes) in SDN-based WSNs have been made in recent years. TinySDN [54] uses an SDN-enabled node structured into three parts namely the TinySdnP which performs a flow table update upon receiving a flow setup response, TinyOS application which creates data packets and places them on the network and ActiveMessageC which is a component of TinyOS managing and providing a programming interface with radio hardware on the sensor. Miyazaki *et al*. [116] propose an SDWSN based on role generation and delivery to and from a reconfigurable node. The node makes use of a modular combination of FPGA and micro-controller for an energy efficient node that can be reprogrammed via OTAP to perform various roles. The micro-controller is used to alter the network behaviour while the FPGA handles heavier data processing tasks. Other enabling hardware such as computer servers for role or task generation and communication modules for the multi-hop wireless link have been presented for

implementation in SDN-based management for WSNs [49, 116].

2.6 OVERVIEW DISCUSSION OF SDN-BASED MANAGEMENT OF WSNS

In this chapter, several contributions to the management of WSNs based on SDN have been made. It can generally be concluded that the overall SDN-based management of WSNs is looked at as being developed around the North bound, East-West bound and South bound abstractions of the SDN architecture. These are classified based on the location of the management control referred to as the manager. Figure 2.5 illustrates this management view.

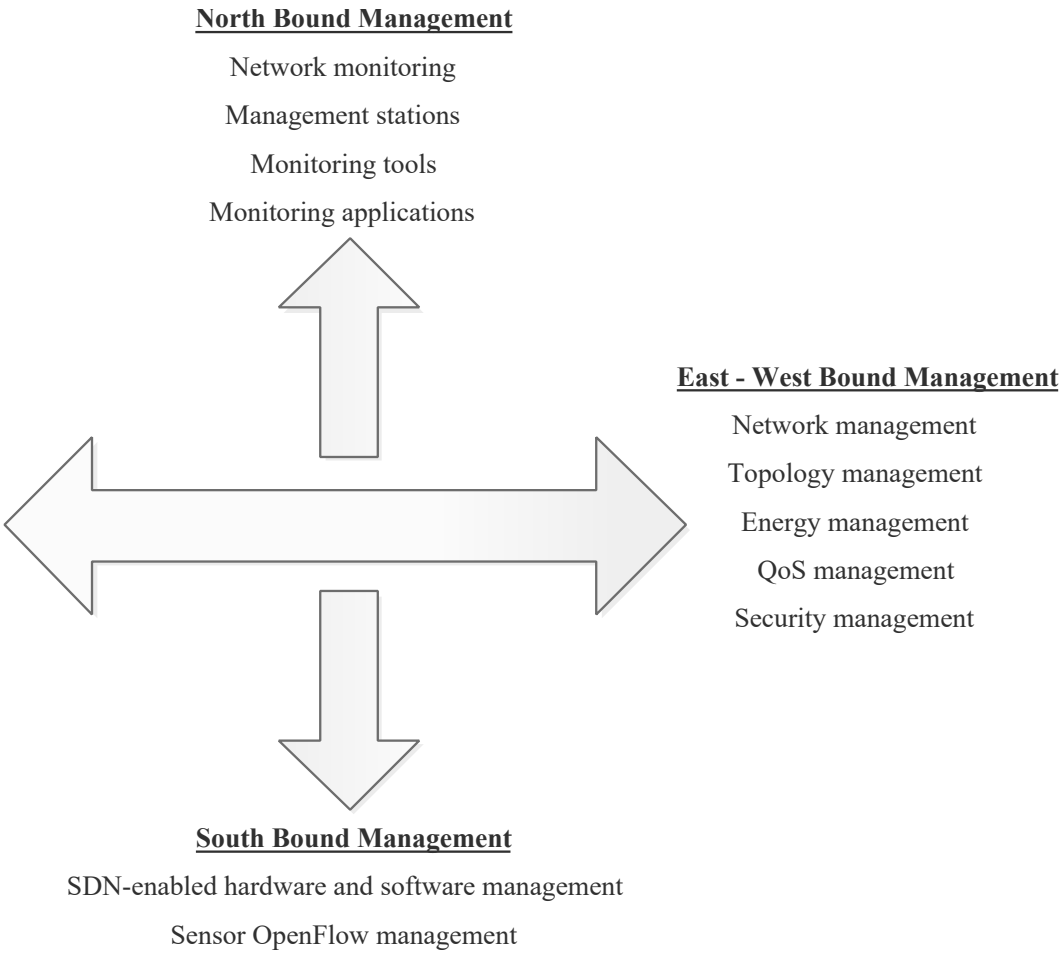

Figure 2.5. SDN-based management abstractions for WSNs, taken from [16].

East-West bound management can further be looked at as central management upon which several proposals have been made towards improving management of control plane functions. Core to the function of SDN-based management of WSNs is the network configuration and topology management.

Managing how the network operates efficiently and how nodes in the data plane communicate with the controller nodes and peripherals is the goal of developing a network configuration and topology management architecture. Soft-WSN an architecture based on East-West management has been proposed and tested by Bera *et al*. [51]. Soft-WSN is a software-defined WSN management system for IoT providing an application-aware service consisting of an application, control and data layers. The architecture is based on a centralized device (enabling hardware) and network configuration policies provisioned on the controller design. Experimental results on Soft-WSN show an enhanced energy efficiency, traffic overhead and data delivery ratio in a network [51].

To ensure SDN integration in WSNs, proposals such as Sensor OpenFlow, SDWN and SDN-WISE techniques which provide solutions to designing and managing WSNs based on SDN in terms of flow setup, in-network data processing and traffic management and general management of flow rules have been made. In managing the topology, Smart, SDCSN and TinySDN provide an efficient high-level architecture resulting in efficient resource allocation. However, Smart does not provide for distributed controllers compared to SDCSN and TinySDN posing a greater risk in system security and reliability once the base station controller in Smart comes under attack. The network virtualisation demonstrated in the virtual overlay topology introduces greater flexibility in managing and monitoring networks. In managing the localization of nodes, localization algorithms have been proposed to improve the accuracy of locating nodes. Zhu *et al*. [82], for example, present a localization algorithm that does not only improve the performance of the localization but also conserves energy which is a vital component. In tandem with the management of node localization is mobility management which takes into account the movement of nodes into and out of the network. An attempt at managing mobility has been made by Zhou *et al*. [69] outlining the steps for handling node mobility in and out of an SDWSN however, there is room for research in SDN-based management of node mobility.

In terms of communication management, the importance of using multi-hop techniques in an energy constrained network has been emphasised. As data is carried and passed on from node to node, less energy is required and it also allows for data aggregation which reduces redundancy in the network. The challenge with this for enabling SDN-based management is the requirement for communication between the global controller and the nodes on the network as it may happen that a link may break once any of the nodes acting as repeaters falls out of the network. A more reliable communication scheme would integrate long-range radio between each of the nodes and the controller although it may be argued that such a system would be less energy efficient. One solution would be investigating the

possibility of using low power long range radio schemes such as LoRa [91].

In addition, to reduce rigidity in choice of design or communication scheme due to energy constraints for WSNs based on SDN, there is a need for integration of energy harvesting mechanisms in the network. Methods to reduce the overhead on control data such as the proposed reinforcement learning technique [105] or a promising energy harvest method that uses wireless power transfer from a transmitting generator to the nodes in the field [108] can be further developed and investigated to reduce the energy bottleneck that exists in WSNs. In SDN-based energy management of WSNs, implementing a mix of proposed schemes to reduce and monitor energy consumed in the network such as duty cycling, data aggregation, smart [43] and multi-task methods [104] with energy harvesting techniques would result in the higher energy efficiency of the network. It is also worthwhile to mention here that sensor node design is also focused on minimizing the energy consumed by the use of low power devices in the electronic design while meeting the required QoS. In Table 2.5 various SDN-based management schemes and techniques for WSNs discussed in this chapter are evaluated against the design criteria required for improved management.

2.7 RESEARCH CHALLENGES

Generally, there is significant and ongoing contributions in investigating SDN-based management of topologies, energy, QoS and enabling technologies for WSNs. However, a number of open challenges still exist:

2.7.1 East-West bound management

With the increase in sensor node deployment, the use of a single controller becomes inefficient in terms of maintaining a global network view therefore, proposals have been made towards using multiple controllers to solve this issue [46]. Efficient SDN-based management of communication protocols between distributed controllers in the control plane remains an open challenge. This includes the development of East-West bound APIs for efficient control of traffic flow in the data plane.

Table 2.5. SDWSN management scheme evaluation against design criteria, taken from [16].

Management scheme	Energy efficiency	Robustness	Scalability	Adaptability
Sensor OpenFlow [37, 38]	-	Yes	Yes	Yes
SDWN [42]	Yes	Yes	-	Yes
Smart [43]	Yes	No	No	Yes
SDN-WISE [12]	Yes	Yes	Yes	Yes
SDCSN [70]	Yes	Yes	Yes	Yes
TinySDN [54]	Yes	Yes	Yes	Yes
Virtual Overlay [41, 69, 72]	-	Yes	Yes	Yes
Multi-task [104]	Yes	-	-	Yes
SDWSN-RL [105]	Yes	Yes	Yes	Yes
Wireless power transfer [108]	Yes	Yes	Yes	-
Function alternation [49]	Yes	Yes	Yes	Yes
Statistical machine learning [9]	-	Yes	-	-
Context-based [73, 74]	-	Yes	Yes	Yes
Soft-WSN [51]	Yes	Yes	-	Yes

2.7.2 Network and topology management

Most resource management techniques available such as Smart and SDWN have the problem of not addressing overhead in control traffic which is costly for resource constrained WSNs. Therefore, further investigation on how to reduce in-band control traffic between the control and data planes is required. Efficient techniques for in-band communication of control and data traffic can be proposed and implemented.

Context-based clustering can be studied further using other techniques other than HyperFlow to efficiently implement the logical controller concept while making use of its improved performance metrics such as improved scalability and reduced latency. Aspects of network decomposition hierarchies can be looked into to minimize control traffic and improve energy efficiency. Further investigation is also required to determine the number of controllers needed and their placement given an SDN-based WSN topology similar to the analysis done in [121] for WANs.

SDN-based network monitoring for WSNs is an essential part of network management that still remains largely unexplored. There is a need for more quick and efficient network monitoring techniques for WSNs based on SDN. Node localization is another issue that has the potential to be improved by SDN, conventional WSNs may require nodes to determine their geographical location based on information from other nodes [19] thus expending more energy. This task can be moved to the controller by virtue of its global position reducing the energy consumption on the nodes. In addition, novel localization techniques can be developed harnessing the potential of SDN.

2.7.3 Security management

The existence of a logically centralized controller and cluster heads in the SDN-based architecture for WSNs calls for efficient management of security. This includes investigation and development of support for securing the controller in both distributed and centralized architectures, in addition, to countermeasures whenever a cluster head or controller is attacked. There is also an open challenge in the development of SDN-based key management systems and other encryption techniques for WSNs.

2.7.4 QoS and mobility management

Performance metrics such as latency, reachability and mobility remain open areas of research in SDN-based management of WSN QoS. There is a need for more investigations in the controller placement problems to improve these performance metrics. Wang *et al*. [122] for example, have proposed a K-means algorithm to optimize controller placement thus minimizing network latency. Statistical analysis and artificial intelligence to predict fault occurrence in the network are also being investigated [9] for management of a robust SDN-based WSN. Investigation on the feasibility of mobility management

techniques implemented in cellular networks being used in node mobility management of WSNs based on SDN is also a worthwhile area of research. Back up techniques upon node failure such as function alternation [49] can be investigated further based on leveraging SDN to enable efficient and simpler function alternation. To ensure QoS control and monitoring, benchmarks and Key Performance Indicators (KPIs) need to be integrated into the management framework being developed.

2.7.5 Energy management

Energy is a scarce resource in WSNs and constant research in this area is necessary to ensure efficient energy use in WSNs. A common technique to minimize the energy use is by control of sleep and wake states of sensor nodes. However, SDN-based management of WSNs requires a robust link between the control and data planes for effective management and thus should be investigated further to reduce unreliability in communication between the planes. A paradigm shift is required to move away from data aggregation and further develop efficient long range low power radio techniques. Low power single hop communication techniques can also save energy by shifting in-network processing to the controller leaving the energy deprived sensor nodes with the task of only forwarding data to the controller and going back to sleep. Other improvements of energy efficiency based on the reduction of overhead control traffic and redundancy in routing tasks are also a promising area of research. While new methods of extending network lifetime through wireless power transfer [108] have been proposed, the feasibility of implementing such a technique in a large-scale WSN based on SDN has to be addressed in terms of the scalability-cost ratio.

2.7.6 Enabling technologies

Advances in microelectronic design and manufacture have allowed for miniaturisation of node hardware and at a lower production cost a factor which is useful in large-scale applications and allows for placement of redundant nodes in a network. However, there is a need for continuous research in developing increased energy efficiency while keeping these nodes small and at a reduced cost. The energy harvest mechanisms are one such area of development, there is room for developing batteries that support quick and efficient charging at lower currents. A characteristic like this would be useful in harvesting solar energy for example. Integration of these nodes into the SDN architecture will also require further development of software and hardware to enable the SDN functionality and also provide

support for multiple re-tasking, inter-plane data exchange via APIs and communication protocols (Flow-Sensor [110], FlowVisor [123], HyperFlow [75] etc.). There is also a need to develop some form of modularity in both software and hardware to allow for use in multiple applications and enable easy system maintenance. In terms of choice of using FPGA, micro-controller or a hybrid of both in SDN sensor node design, it can be argued that it vastly depends on energy efficiency, cost and nature of the application. There are still possibilities of integrating low power FPGAs however the cost impact for application in large-scale networks also needs to be accounted for. Generally, novel hardware and software need to be developed to improve support for the measurement, analysis and decision making the process for IoT applications such as smart water grids while allowing compatibility with the integrated management framework.

2.7.7 WSN management framework

In the future, a development trend of more than just concepts and ideas of proposed management techniques is required; more work is needed for proof of concept and actual deployments to allow for a deeper understanding of the necessary architectures and tools for SDN-based management of WSNs. There have been attempts for actual deployment of proposed management mechanisms such as the case of SDN-WISE, the attempt by Miyazaki *et al*. [116] to build a reprogrammable WSN based on SDN and more recently Soft-WSN but there is still room for many such implementations. Areas such as distributed management based on distributed controllers, low power and long-range SDN-based management of WSNs need investigation. As regards to network management, there is a need for the development of a WSN management framework leveraging SDN to ensure sensors/actuators are operating correctly and the heterogeneous SDN-based WSN is properly configured and managed. The management framework can be broken down to manage various aspects of the WSN and it should also be dynamic enough to be used for various applications. A platform such as IT-SDN [13] provides the potential open-sourcesness required for such versatile functionality. Just like MANNA and BOSS network management systems were presented for ordinary WSNs exploring how SDN can improve the performance of these NMSs would prove useful.

2.8 CHAPTER SUMMARY

This chapter reviewed the various contributions to managing WSNs and techniques available for SDN-based management of WSNs. The SDN paradigm has introduced flexibility and simplicity in managing wireless sensor networks, despite having different vendor specific hardware in the network. In Section 2.2 a general outlook on WSN management was provided. However, the inherent properties of WSNs do not permit the easy integration of SDN as it was initially meant for traditional wired/wireless address-centric networks which are different from the data-centric WSNs. Therefore, Section 2.3 focused on the generic architecture of SDN-based WSNs and reviewed the management schemes available to ensure efficient functionality of the network. While a highlight of the main real-world WSN applications and how SDN would improve the management of the applications was presented in Section 2.4. A review of the management classifications namely management of the network configuration, topology, QoS, energy, security, network monitoring and enabling technologies with a further focus on SDN-enabled node hardware and software was made in Section 2.5.

In Section 2.6 a discussion of various proposals and works done necessary for the management of WSNs based on SDN was provided. Furthermore, the section included an attempt to define the overall management of WSNs based on SDN as abstractions of the North, South and East-West bound architecture of SDN. However, there still exists a mix of open challenges available for effective SDN-based management of WSNs and they have been discussed in Section 2.7 with an emphasis on the need for actual implementations and test beds for effective evaluation of proposals and concepts. For a future leading to improved management of WSNs based on the SDN paradigm; novel SDN-enabled sensor hardware, development of efficient SDN techniques for WSN implementation, novel contributions to SDN-based power management and improved performance assessment of SDN-based management architectures for WSNs are expected.

CHAPTER 3 AN SDN-BASED MODULAR MANAGEMENT (SDNMM) SYSTEM

3.1 CHAPTER OVERVIEW

SDN brings greater advantages to management flexibility in wireless sensor networks. However, as has been drawn in Chapter 2 there is a need for a generic framework to better take advantage of the management flexibility that SDN introduces. In line with this research gap, Chapter 3 investigates and fully describes the framework and implementation structure of a generic SDN-based Modular Management System for wireless sensor networks (SDNMM). This chapter also evaluates and discusses the network performance of implementing such a modular system on an SDWSN framework, in this case, IT-SDN [124]. The performance of SDNMM is compared to that of baseline IT-SDN and SDN-WISE [12]. The basis of evaluation takes into consideration a resource allocation module use-case scenario that carries out task sampling and action allocation based on available node resources. The goal of this evaluation is to derive results on the feasibility of the proposed modular management system for SDN-based WSNs. Within the scope of this study the evaluation of the SDNMM system is based on simulation only and actual experimentation is included as part of future work due to the unavailability of access to a suitable SDN-based WSN test bed during the period of study. The works presented in this chapter have been published in the IEEE Systems Journal titled "SDNMM - A generic SDN-based modular management system for wireless sensor networks" [125][1].

The outline of the rest of this chapter is as follows. Section 3.2 discusses the background relating to SDN-based management frameworks for WSNs in general. The architecture for the modular

[1] © 2019 IEEE. Republished with permission, from M. Ndiaye, A. M. Abu-Mahfouz, and G. P. Hancke “SDNMM - A generic SDN-based modular management system for wireless sensor networks”, IEEE Systems Journal, 2019.

management system being proposed is then described in Section 3.3 while Section 3.4 highlights functions and features of the Management Service Interface (MSI) which is the core component of the management framework. Details on the implementation strategy of such a modular management system are provided in Section 3.5. Section 3.6 presents SDNMM performance results of several quality of service metrics from a series of simulations while Section 3.7 discusses the presented results in detail. In Section 3.8 the system challenges observed based on the performance results are discussed and Section 3.9 summarises this chapter.

3.2 BACKGROUND ON SDWSN MANAGEMENT FRAMEWORKS

Several techniques and frameworks have been proposed in enabling the management of wireless sensor networks based on SDN. Notable contributions to this paradigm include Sensor OpenFlow [37], TinySDN [11], SDN-WISE [12] , Coral-SDN [126] and IT-SDN [13]. These techniques play an important role in providing an operating environment for SDN-based management systems. However, the development of generic SDN-based management system for WSNs with high-level abstractions still remains largely open for research. Smart [43], Soft-WSN [51] and more recently a 6LoWPAN SDN-based IoT framework [127] have been proposed to provide a more generic management abstraction.

3.2.1 Smart

Smart offers some form of topology management through localization and tracking algorithms that work in tandem with the controller. In Smart, a general framework is proposed with the SDN controller at a base station. Gante *et al*. [43] argue that SDN would greatly ease the complexity of sensor network management and raise important questions regarding the integration of a distributed controller in terms of general network performance and energy consumption of the WSN. They also discuss the effect of the OpenFlow protocol in constrained sensor networks and also synchronization issues regarding maintaining the controller global view. However, the authors in Smart mention that their proposal was a preliminary stage with no implementation or framework evaluation presented. Their proposed Smart framework did not address the management of application-specific requirements or the integration of specific management entities. Furthermore, to allow for a global network view that SDN provides, Smart uses a single base station controller which has reliability and security challenges upon failure

or compromise of the centralized controller respectively. Another challenge associated with such centralized management is the accumulation of overhead traffic data to the controller in an already resource-constrained WSN.

3.2.2 Soft-WSN

Soft-WSN addresses the challenge of meeting application-specific needs for IoT by use of SDN. Two management techniques are developed to meet device and network management. Bera *et al*. [51] investigate sensing tasks and delay including active-sleep scheduling to implement device management. They also modify network policies at run time to meet topology requirements. Results from a hardware test-bed show improved packet delivery rate and energy consumption compared to an ordinary WSN. However, the discussion and implementation in Soft-WSN are limited to providing only device and topology management services. Like Smart, it is also based on a single global controller resulting in a centralized management scheme and thus prone to the associated challenges. Furthermore, both Smart and Soft-WSN also do not address issues of modularity to allow for widespread WSN applications or upgrades of management components.

3.2.3 6LoWPAN focus for SDN-based management

Lasso *et al*. [127] in an SDN-based IoT framework for 6LoWPAN focus on SDN-based energy management using a transmit power control mechanism from the controller specifically for a 6LoWPAN. The authors implement the framework on the open source Contiki [64] Cooja platform and results show improved energy performance in the SDN-enabled nodes. However, most of the work focuses on improving the energy consumption, issues related to the management framework operations including the modular aspect or the effects of the framework on traffic overhead and packet delay are not addressed in detail.

Considering the above-related work, it can be noticed that there is still a need for a more open source and general management framework that allows some form of modularity for easy implementation of management methods. Another bottleneck that could be observed is the increased control traffic coupled with the use of a centralized controller, an aspect that can be mitigated by using a distributed controller mechanism [128, 129]. There is a need for a framework that would allow the management

methods such as that those implemented in [51] and [127] to be packaged as a module for quick testing and implementation. The novel SDN-based modular management framework proposed in this chapter provides a promising solution to addressing the above-named issues.

3.3 MODULAR MANAGEMENT SYSTEM OVERVIEW

Figure 3.1 shows the architecture of the modular management system based on a hierarchical SDN distributed controller architecture efficient in handling scalability and traffic overhead [128, 129]. The overall framework can be broken down into the following SDN-based management abstraction planes.

3.3.1 Application plane

The application plane is composed of multiple third-party user applications that communicate with the network using APIs provided by the controller. The applications can be used to handle the following tasks:

- Monitoring of network state including fault detection or isolation and energy level monitoring.
- Task configuration of nodes to meet application demand.
- Policy issuance for network and device-specific tasks based on context.

3.3.2 Control plane

The control plane in the SDNMM framework is a two-tier configuration of a global controller and physically distributed local controllers (cluster managers). Both the global and local controllers are of a higher resource capacity compared to sensor nodes. The Management Service Interface (MSI) and the various management modules are contained in the global controller while cluster managers handling sub-management tasks are in the local controller tier of the control plane. Each cluster manager is responsible for events and tasks occurring in the respectively assigned clusters. Cluster managers are able to communicate with each other via East-West APIs. Generally, the following functions should be executed in the control plane:

- Integration of core WSN management modules such as configuration, topology, energy, security and QoS.
- Provision of APIs to the application and data planes.
- Collection of context information and handling of policies including issuing policies based on context and flow commands for application requests.

3.3.3 Data plane

The data plane is made up of a network of sensor nodes configured in clusters to allow for improved scalability. In each cluster, SDN-enabled nodes which are a hardware combination of an SDN-enabled switch and an end-device [11], generate data packets which are sent to the sink node. The sink node allows for a data link to the neighbouring cluster manager (controller) and cluster. Network APIs are made available by the controller to allow relaying of sensor data including context/policy information to and from the control/application planes.

3.3.4 System features and functions

Generally, the proposed SDN-based modular management system introduces the following features and functions into wireless sensor network management:

- Modularity: Allows each management entity to be a well defined functional component with APIs available to communicate commands, contexts, and policies via the management service interface (MSI). The management components/modules provide a higher level management abstraction thus introducing flexibility in the use of existing or new management techniques.
- Generic: Modular components allow the use of the system in various applications and heterogeneous environments as components can be designed and added to the framework depending on application need.
- Scalability: The distributed hierarchical architecture on which the framework is based allows for cluster managers to be assigned to each node cluster limiting the frequency of overhead traffic that needs to be sent to the global controller. This configuration gives room for efficient network expansion.

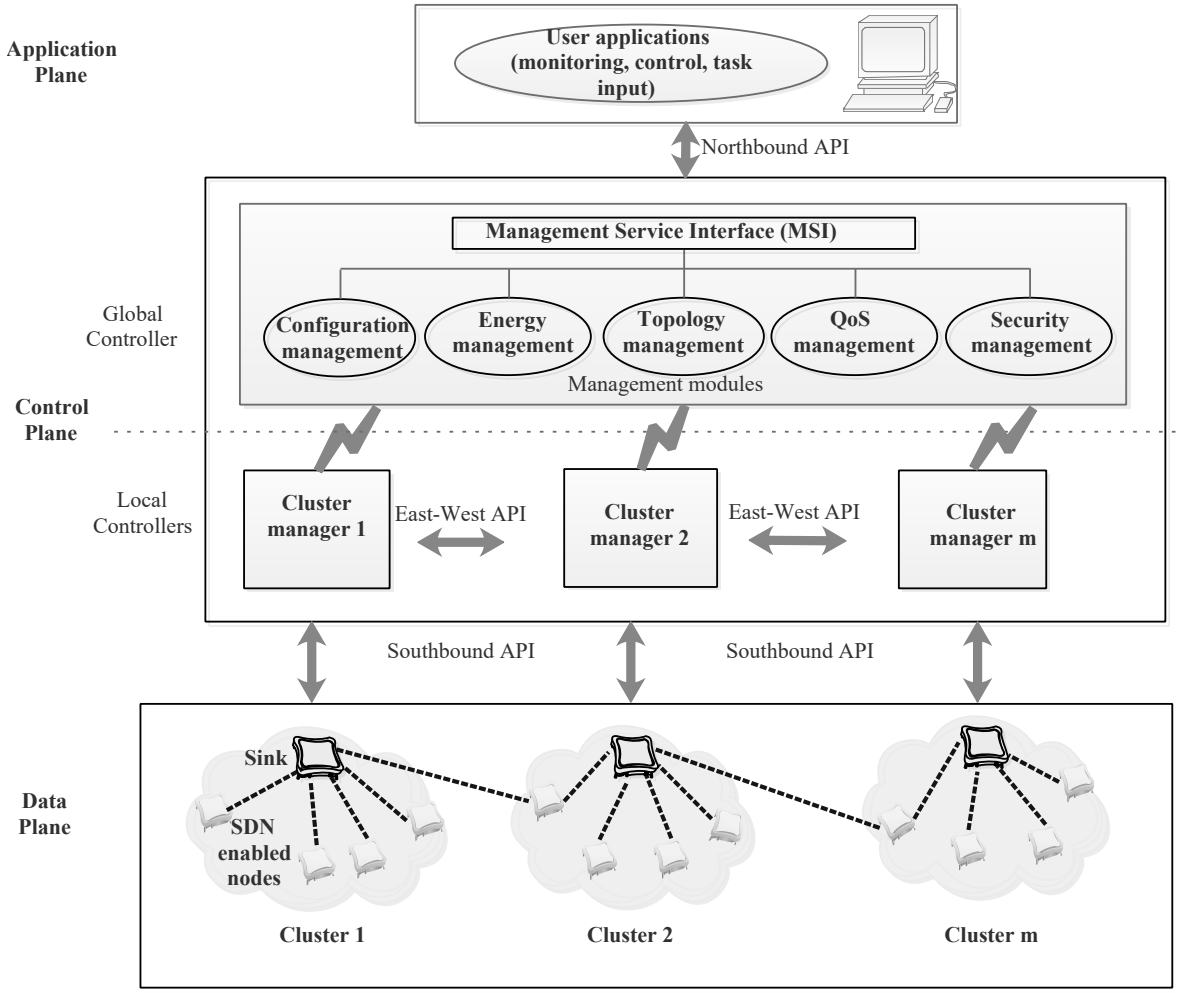

Figure 3.1. SDNMM system architectural framework, taken from [125], © 2019 IEEE.

- Adaptability: Sensor tasks can be issued on demand to meet application-specific needs. The possibility of function alternation upon neighbouring node failure is introduced.
- Robustness: Multiple context-aware cluster managers working in tandem with the global controller ensures effective detection and isolation of faults. Functionality can be introduced for a cluster manager to temporary manage a neighbouring cluster that has a cluster manager that is down.
- Energy efficiency: Apart from the ability to use energy management modules for active-sleep management, SDNMM leverages SDN to move resource-heavy tasks to more resource capable hardware in the network such as the cluster managers and the global controller. This leaves nodes with the simple task of forwarding event and context data.

3.4 THE MANAGEMENT SERVICE INTERFACE

The management service interface (MSI) provides an interface to enable this novel SDN-based approach of adding management services as modules or components. The interaction between the MSI and the modules can be through coupling APIs or through classification criteria based on management task type as has been implemented in this proposed system. The MSI is generally middle-ware between management services and user applications/network infrastructure. Core features and functions of the MSI include:

- Enabling a modular approach to SDN-based management for easy and rapid addition or removal of management services to the WSN necessary for future upgrades or expansion.
- Provision of loose coupling through APIs or event classifiers to management modules, application and data planes enabling the integration and use of third-party management services in an orderly manner.
- Provide an interface for the exchange of context information between management modules, for example, the energy management module can give insights on the status of energy resources in the network to the QoS management module which can then make decisions based on available resources.

APIs, contexts, and policies play a crucial role in ensuring this flexible coupling and feature function between management components in the SDNMM framework. In this study, the MSI has been built and implemented as a function that can be called at any stage in the processing of a task in the SDN architecture. Consider the sequence of processing a task in a typical software-defined wireless sensor network shown in Figure 3.2. The MSI can be engaged during the flow setup process in the controller as part of the data management process which may include altering task action based on destination node resource capability or addition of management data to the flow setup packet. The resource allocation management module is implemented and evaluated based on this approach.

3.4.1 Context and policy handling

Context awareness and the issuance of the corresponding policies are an important aspect of WSN management. Context is basically information or an event that often results in an action policy being

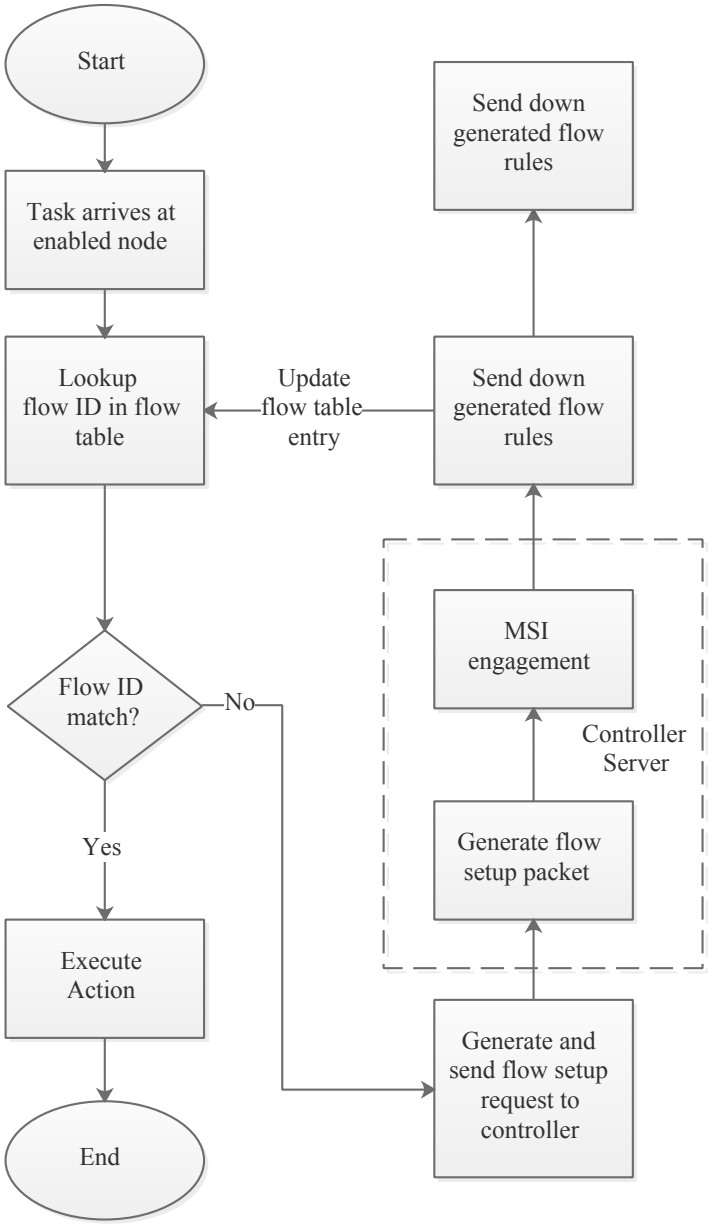

Figure 3.2. MSI engagement in SDWSN task sequence, taken from [125], © 2019 IEEE.

issued for reconfiguration purposes. WSN context can be classified into three categories [14]:

- Node resources: Information here mainly includes the status of node residual energy, memory usage, and sensing capabilities. In the SDNMM framework, node resource context information is obtained from the data plane. Resulting policies can be made from the application and control planes.
- Network state: Context in this category provides information on the running status of the network

in terms of communication link status, availability of bandwidth and other network topology related aspects. The source of this information in SDNMM is both from the data plane and aggregate data from cluster managers. Aggregate cluster manager data provides network-wide context information. User applications and management modules can input policies to respond to this kind of context.

- Application requirements: Users can set parameters for a task with certain requirements which provide context data for the MSI and management modules upon which policies can be generated. Examples include requests to change sensing task, add or drop packets and requests to encrypt data. In the proposed SDNMM system, this context category originates from the application plane.

Figure 3.3 shows the flow of contexts and policies in the SDNMM system.

3.4.2 API functions

APIs allow the interaction and coupling of the components in the framework shown in Figure 3.1. Northbound APIs allow the MSI in the control plane to interact with user applications in the application plane. Information such as task definition parameters, context, and policy data are transferred via this API. In the southbound direction, cluster managers interact with sensor nodes using southbound APIs. Cluster managers can request for sensor and network running statuses, provide flow commands, issue required task parameters and other context-related policies while being coupled by these APIs. East-West APIs facilitate the interaction of cluster managers. Information related to scheduling of network-wide tasks or policies can be exchanged over these APIs. APIs can be used to allow interaction of the MSI with management modules.

3.5 SYSTEM OPERATION AND IMPLEMENTATION STRATEGY

Notable platforms that can be used to implement an SDN-based management system framework include SDN-WISE [12], Coral-SDN [126] and IT-SDN [13]. All three platforms provide an environment controller and neighbour discovery suitable for SDWSNs. However, both SDN-WISE and Coral-SDN are limited in terms of developer support and source code availability making it a difficult choice for a generic framework such as SDNMM to be implemented on. On the other hand, IT-SDN provides

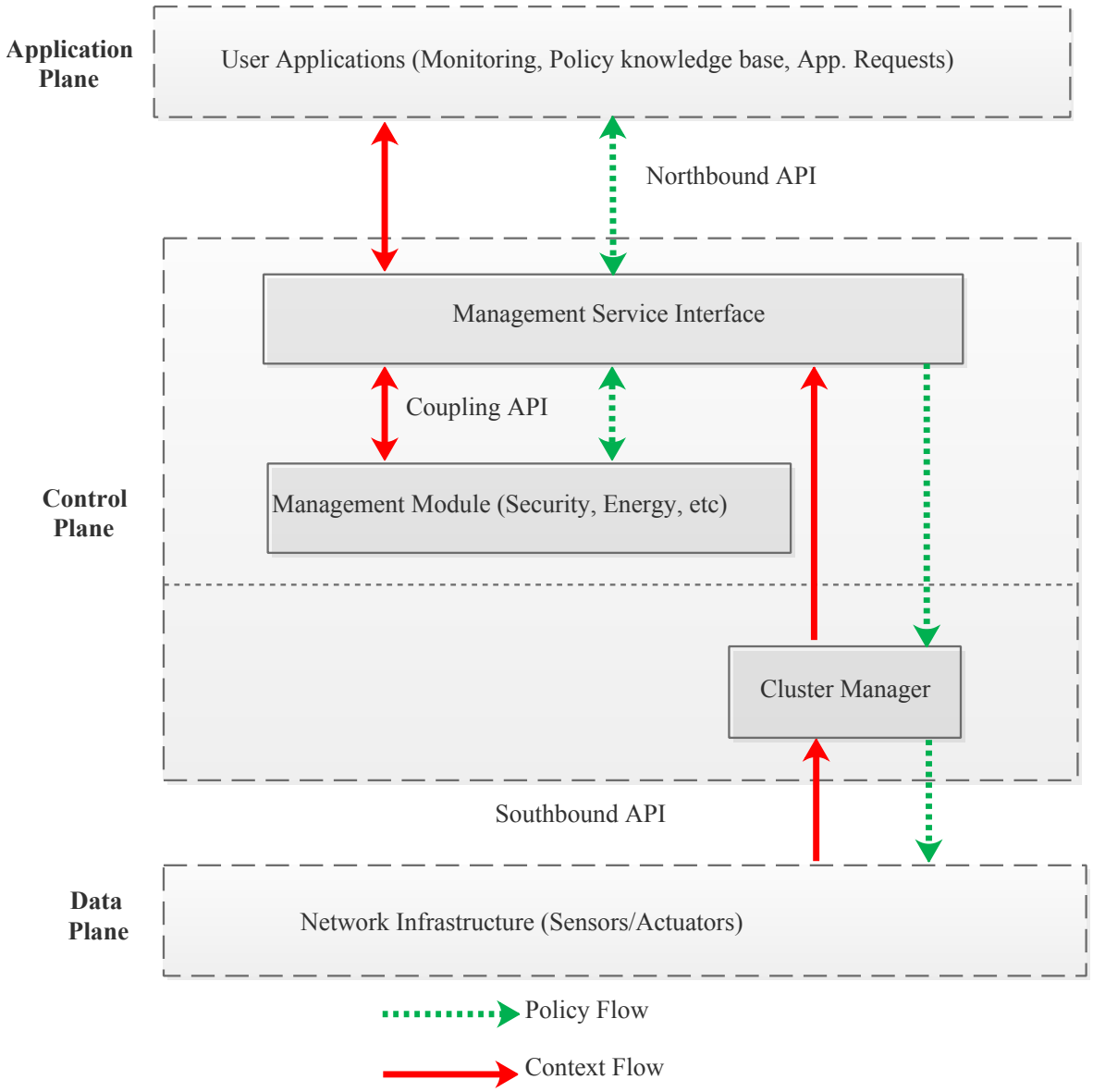

Figure 3.3. SDNMM context and policy flow, taken from [125], © 2019 IEEE.

the required open source availability and hence SDNMM is built on it to allow for a more generic use case. IT-SDN is based on Qt which is available for all major computer operating systems and the Contiki [64] Cooja tool which is open source and easily accessible.

3.5.1 Implementation tools

There are mainly three tools required to effectively implement and evaluate the proposed modular management system. These are as follows:

(a) Microsoft Visual Studio Code (Ms VS Code)

This tool allows for editing and implementation of system algorithms and procedures. This includes context report generation, transmission to the sink or controller and process to store context information in the controller. The choice of editor (Ms VS Code) was mainly due to the tools ability to provide an integrated terminal and unified view of the management system files. The integrated terminal allows for the compilation of node firmware necessary for the Contiki Cooja simulation and also provides an interface to run the Cooja simulation tool. Figure 3.4 shows a screenshot of Ms VS Code with compiled firmware for the controller, enabled and sink nodes respectively in the integrated terminal.

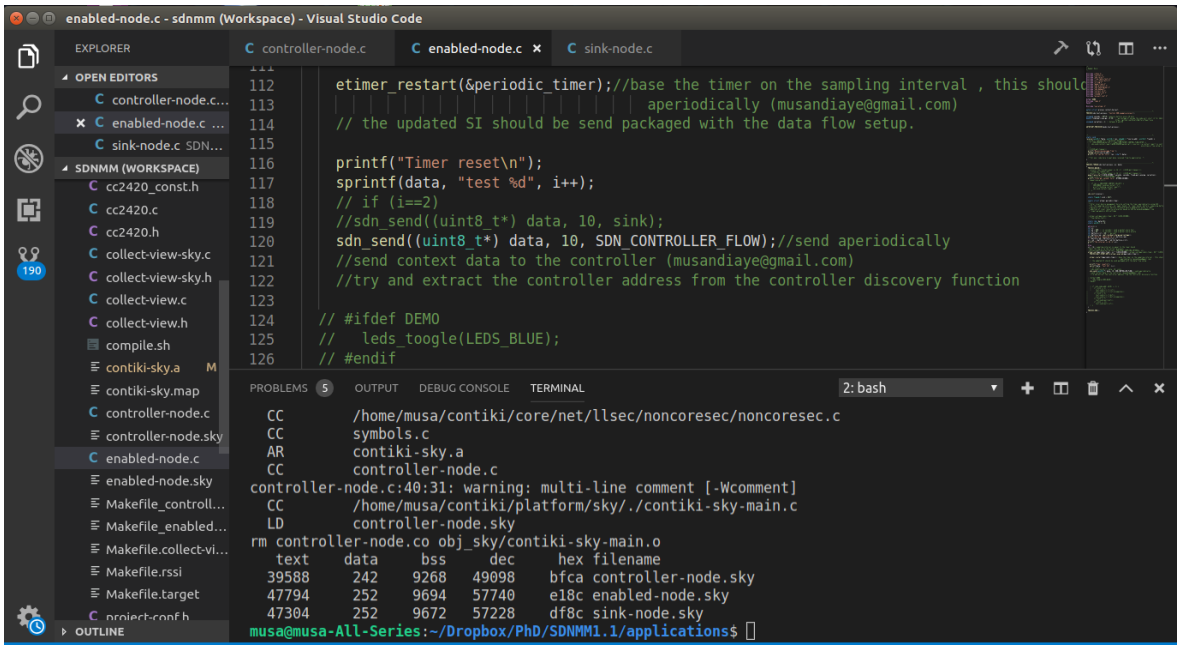

Figure 3.4. Ms VS Code edit and compilation window.

(b) Contiki Cooja

Cooja is a java based Contiki OS simulation tool for both traditional wireless sensor networks and SDN-based WSN as in the case of the proposed system. It provides a platform to test compiled node firmware and a simulation output that can be processed after to obtain performance metrics. A python script provided in IT-SDN can be used to process the simulation output alternatively a Cooja inbuilt javascript simulation editor can be used to set simulation time and also calculate performance metrics. Another useful feature in the Cooja tool required by SDNMM is the controller server serial port for serial communication between the controller node in Cooja and the controller server in Qt. The Cooja simulation tool including the controller server port connection plugin and the javascript editor are shown in Figure 3.5.

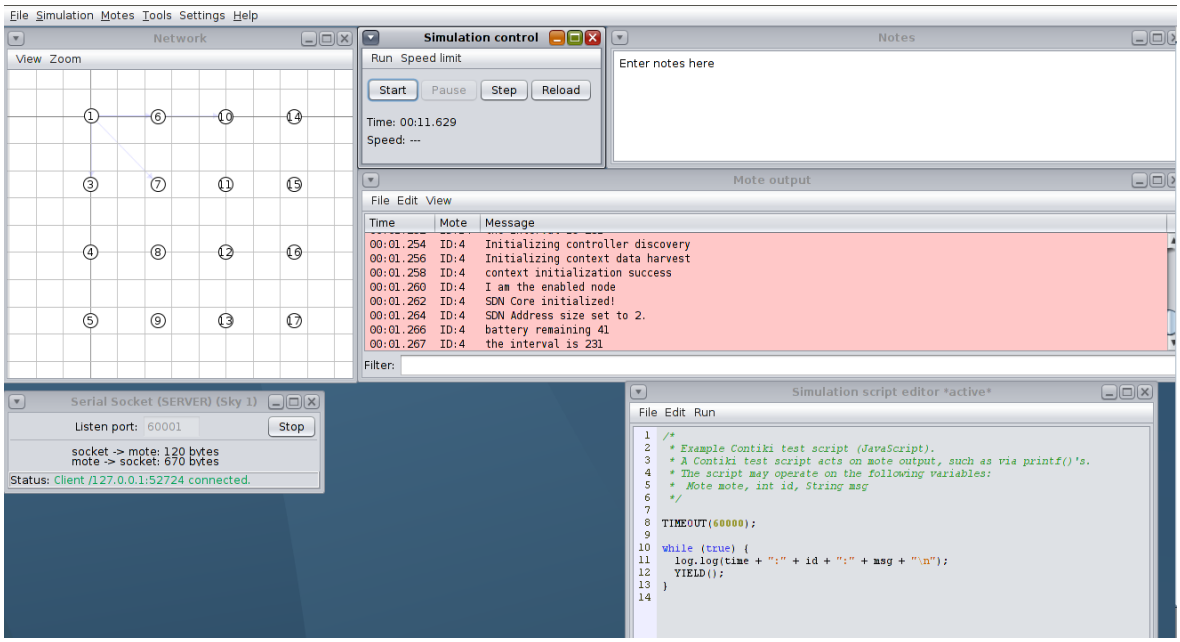

Figure 3.5. Contiki Cooja simulation environment.

(c) Qt controller server

The main controller operations including localization, routing and eventual flow setup creation occurs in a resource capable controller PC server. The server includes a serial client communicating with the controller node in a Cooja network via a set serial port number. This server has been created in Qt creator as part of IT-SDN however as part of this study, the controller server GUI has been modified to add aspects of monitoring and display of performance metrics. These extra features have been added under the management tab option in the GUI and they are discussed further under Section 3.5.2 of this chapter. The general outlook of the controller server in Qt is shown in Figure 3.6.

3.5.2 Monitoring management

The underlying IT-SDN platform uses the Cooja simulator GUI to set up and monitor various aspects of the WSN and also extends monitoring to a computer GUI developed in Qt taking advantage of a debug window to monitor network messages and configuration. As part of the proposed management system features to monitor data on network performance metrics as well as the ability to visualize these metrics in a Qt-based GUI have been added. Figure 3.7 shows this GUI. The GUI embeds a python script provided by the IT-SDN tool for calculation and display of simulation performance metrics such as packet delivery rate, packet delay, number of data packets and number of control packets.

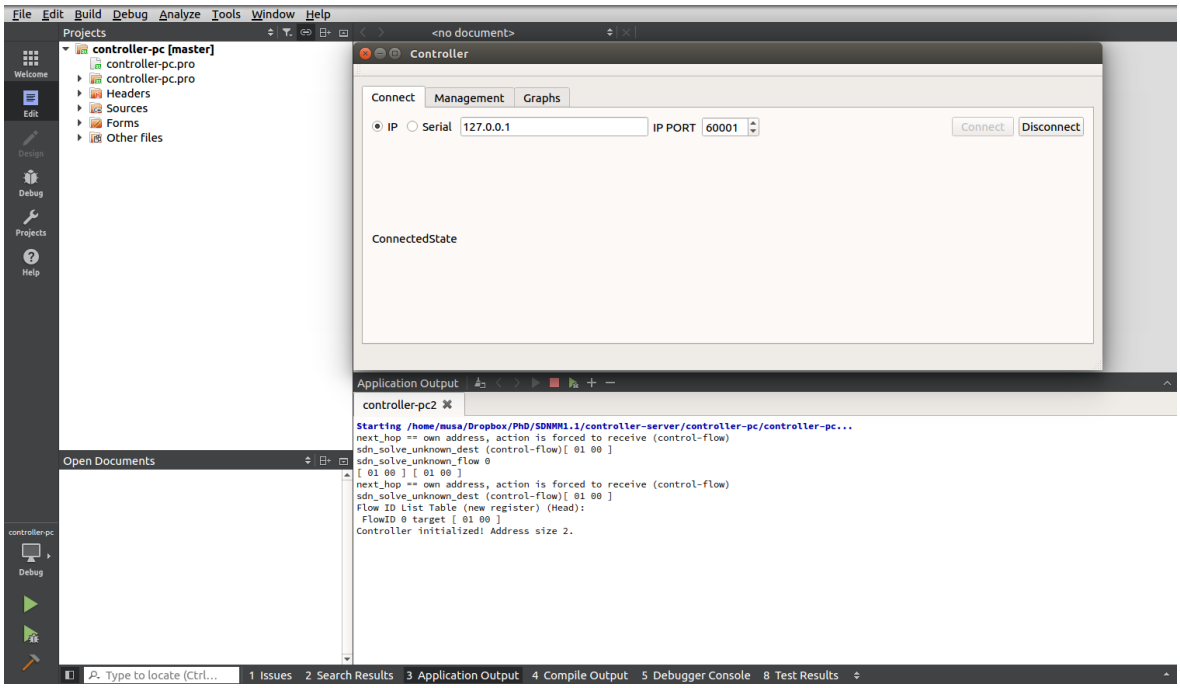

Figure 3.6. Qt controller PC window.

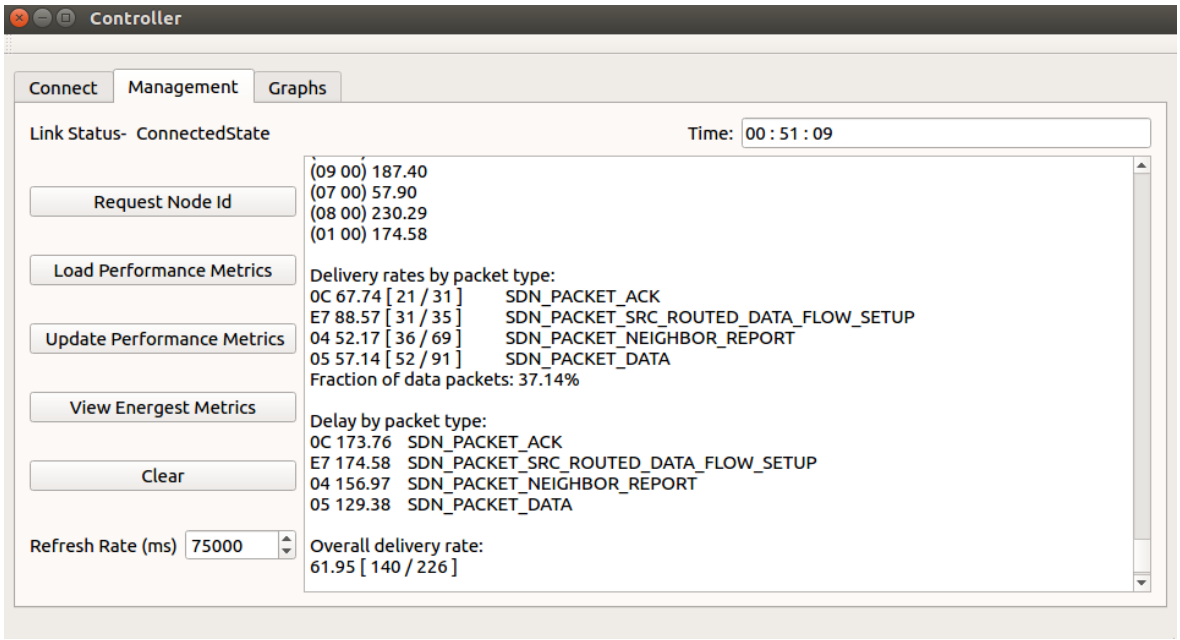

Figure 3.7. SDNMM monitoring GUI.

3.5.3 Context data approach to management modularity: A resource allocation module use case

3.5.3.1 Context data pooling

SDNMM introduces an aspect of management modularity based on a context data knowledge base stored in the controller memory using dynamically linked lists. A context collect process has been developed to collect and send context report to the controller on event change and at various sampling intervals depending on battery level. To integrate energy efficiency in the SDNMM design, data collection outside event changes are based on a Sampling Interval (SI) that is determined from the granularity settings [130]. Equation (3.1) shows how this interval is calculated. As a measure, the lower bound (lb) is set to collect data every 120 seconds and the upper bound (ub) to collect data every 600 seconds. As part of the implementation strategy, a battery level of 40 percent (%) is considered as the critical battery threshold , below which the sampling interval is recalculated at the device level and sent to the SDN-core which is middleware that manages core events between the data and control planes of the framework.

$$SI = ub - battery\left(\frac{ub - lb}{batt\_crit\_th}\right) \tag{3.1}$$

where SI is the sampling interval, ub is the upper bound, lb is the lower bound, $battery$ is the percentage of battery remaining in the node and $batt\_crit\_th$ is the critical battery threshold.

Context data collected includes battery remaining in percentage, Received Signal Strength Indicator (RSSI) in dBm, the number of tasks currently being processed by the node and the number of neighbours the node has in its locality of reference. PowertraceK [131] which is an extension of powertrace in Contiki based on a near realistic Kinetic Battery Model (KiBam) is used to monitor the battery usage and related energy parameters. Figure 3.8 shows the structure of the context report packet.

The context data is stored in a context table in the controller which forms a unified knowledge base and is accessible to the MSI and associated management modules. A sample of this context table has been extracted from the controller log file in a Cooja five (5) node simulation of SDNMM and is shown in Figure 3.9.

Header	FlowID uint16	Battery uint8	Neighbour Count uint8	Task Count uint8	RSSI uint8

Context Data

NAME		LENGTH (bytes)	FUNCTION
Header		6	Contains SDN header information such as packet type, source address and reserved slots.
FlowID		4	Used to contain source and destination node FlowID information.
Context data	Battery	1	Battery shows the percentage battery level in the source node.
	Neighbour Count	1	Shows the number of neighbours the source node has.
	Task Count	1	Data showing the number of pending tasks of the source node.
	RSSI	1	Shows the last registered received signal strength.

Figure 3.8. Context report packet format, taken from [125], © 2019 IEEE.

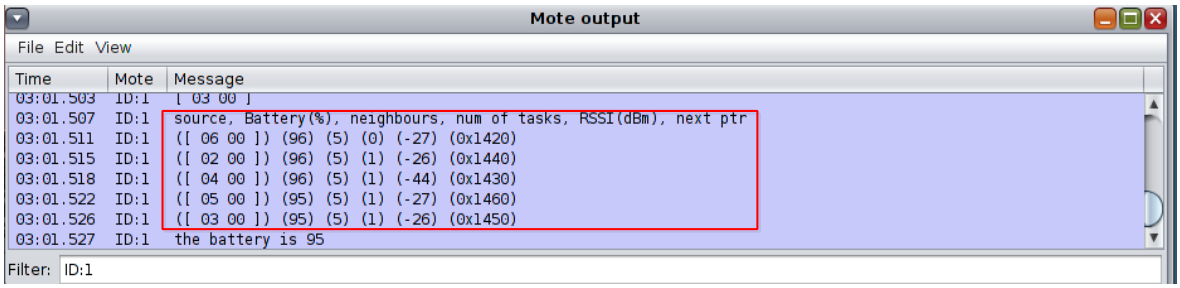

Figure 3.9. Sample of the context table log in controller (Node Id:1) dynamic memory.

3.5.3.2 The MSI function

As a modularity enabler, the MSI function shown by the state diagram in Figure 3.10 is set up based on a state machine, where a management module can exist as a state. Important factors in this function include the task definition, classification criteria and context data. Different management modules will require different context data based on application. The management modules will then transfer the

task to respective execution states based on function. The MSI function uses classification criteria to determine which management module should handle the task. Classification criteria should correspond to input values required by the various management module. Node resource values for example would be classified under resource allocation management, data encryption values would be classified under security management, node location values would classify as topology management and so on.

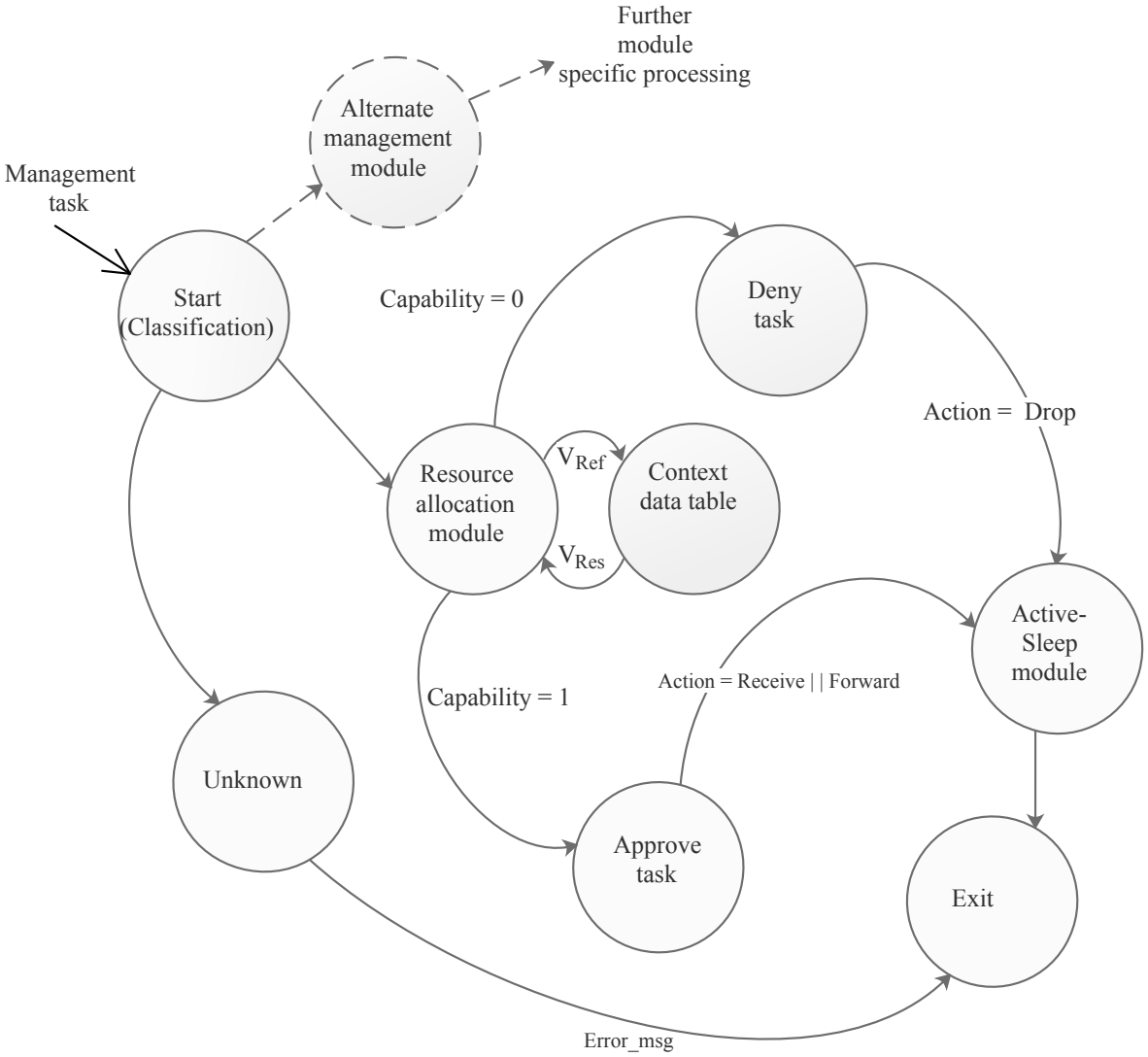

Figure 3.10. MSI state diagram, taken from [125], © 2019 IEEE.

Once the MSI receives the management task, it goes into the classification state which determines which module has to handle the task. Based on set parameters the right management module is selected and the corresponding task is sent to it otherwise the classification state flags an unknown and exits. With the case of the resource allocation module implemented in SDNMM, tasks such as data flow

requests which occur when a node requests an action on a received task and source routed data flow setups flag the resource allocation module in the classification state. In both tasks, it is necessary to check whether the destination node address is capable of handling the set task based on the available resources. The allocation module sends a verification reference (V_{ref}) to the context data knowledge base and awaits response (V_{res}) which is used to determine capability. If the node is capable, the task action is approved otherwise it is denied and the task is dropped. The active-sleep module state has been included to cater for the nodes ON and OFF states during sending down of management policies. This module can further be modified to work with a sleep-scheduling algorithm. A code snippet of this MSI function is shown in Addendum A and the pseudo code for the resource allocation module function is shown in Table 3.1.

3.5.3.3 Overall implementation structure

The overall program structure of SDNMM implementation and operation can be summarized by the block diagram shown in Figure 3.11. The block diagram shows all the core components of the SDNMM framework namely the SDN-enabled node (also referred to as an enabled node), SDN-core, the controller node, and the controller-PC. The enabled nodes and controller node communicate via a radio link while a serial link maintains data communication between the controller-PC and the controller node. The controller-PC is a Qt-based front end handling application plane protocols including monitoring, routing, and processing of data and control flow setup and or request packets. The controller node handles events between the controller-PC and the data plane, for instance, the storage of context data upon reception of a context report packet. The SDN-core plays an important role in initiating neighbour/controller discovery and context collect processes. It also handles events to send neighbour and context reports whenever an event request is posted to it. The process of engaging protocols for sending, receiving and enqueuing of packet data in the southbound region of the SDN architecture is also handled by the SDN-core. Each enabled node is responsible for the provision of context data and execution of sensing tasks and handling of messages based on flow table rules.

3.6 MODULAR SYSTEM EVALUATION

A series of simulations were set up to evaluate the performance of the resource allocation module on top of the IT-SDN platform in the SDNMM framework with several iterations to observe and

Table 3.1. Resource management module Pseudo Code, taken from [125], © 2019 IEEE.

input: pointer to task destination sensor node

output: next state based on capability and collect rate (SI)

1: *Get destination node address and set as verification reference (Vref).*
(*When a node is registered on a particular task in a flow setup configuration, check the node resource capability by referring to context table information using node address as search criteria*)

node_address = **SDN_header**(pointer)->source
Vref = **context_table_get**(node_address)

2: *Access context knowledge base and retrieve verification response with node resource information.*
(Get the battery level, given the critical battery threshold, the upper and lower sampling interval (SI) calculate the suitable sampling interval using (3.1))

if (battery_level<=crit_batt_thresh)
    calculate new SI
end if

3: *Calculate node capability based on pending tasks and neighbours available*.

if (resources >= set requirements)
    capability = True **else**
    capability = False
end if

4: *Set management action (policy) based on node capability to handle task.*

if (capability==True)
    next state = approve **else if** (capability==False)
    next state = deny && **recalculate_capable_route**()
end if

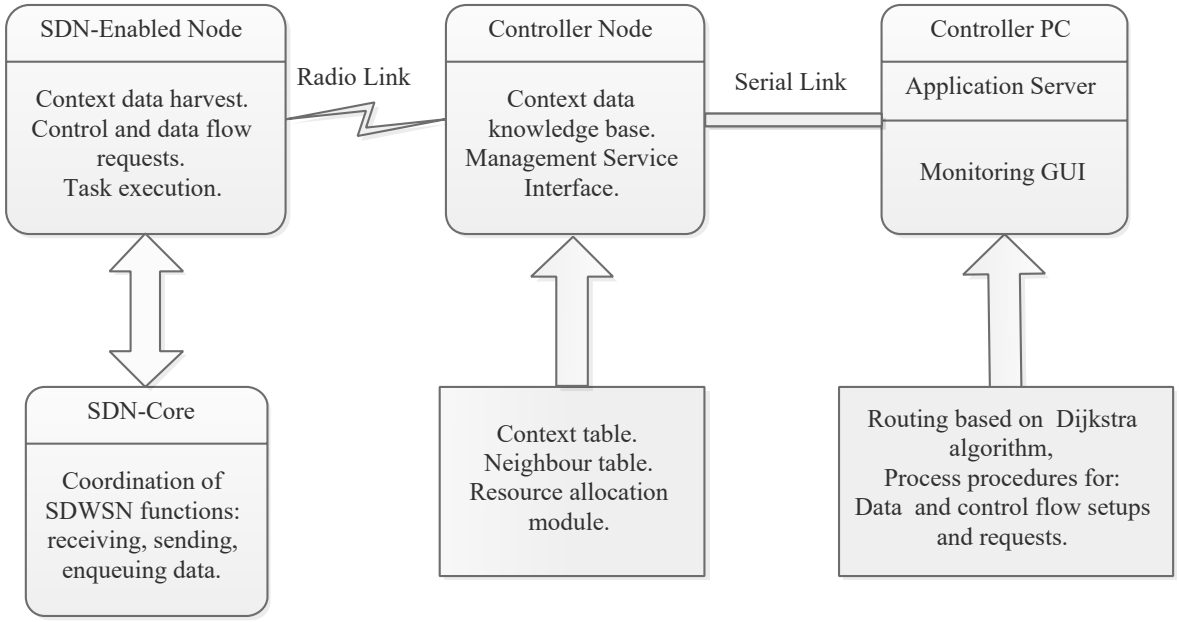

Figure 3.11. SDNMM implementation program structure, taken from [125], © 2019 IEEE.

ensure that outliers contributed to less than 5% of the results. The first set of simulations were to evaluate system scalability , the second set looked at the effect of varying network topologies on system performance and the last simulation produced results on energy efficiency based on the amount of energy consumed by 15 SDN-enabled nodes in the simulation time period for a 16 node grid setup (considering the 16th node as a non-resource constrained controller node and hence omitted in the energy efficiency evaluation for this setup).

The following performance metrics were evaluated:

(a) Number of control packets: These are the total number of control messages generated during a time period.

(b) Packet Delivery Rate (PDR): This is the ratio of the total number of successfully received test data packets at the sink to the total number of test data packets generated by the SDN-enabled nodes during a time period. PDR in percentage is calculated as shown in Equation (3.2).

$$PDR\ (\%) = \frac{\sum packets\ received}{\sum packets\ sent} \times 100 \tag{3.2}$$

(c) Packet delay: This is the average time taken by data packets from being sent by the SDN-enabled node to being successfully received by the sink during a time period. Delay is often expressed in

milliseconds as shown in Equation (3.3).

$$Delay\,(ms) = \frac{\sum (Time\ received(ms) - Time\ sent(ms))}{\sum packets\ sent} \tag{3.3}$$

(d) Energy : This is the power (watts) sustained in one second. The resulting unit is watt-sec or joule. It is given by the product of voltage (volts) , current (amps) and the time (seconds). To measure the energy consumed powertrace [64] is used, which models energy consumption based on active transmit W_{tx}, active listen W_{rx}, cpu mode W_{cpu} and low power mode W_{lpm}. Thus the total energy W_{total} is given by Equation (3.4).

$$W_{total} = W_{tx} + W_{rx} + W_{cpu} + W_{lpm}\ Joules \tag{3.4}$$

To obtain values for the discussed metrics IT-SDN uses a python script included as part of the IT-SDN package and for SDN-WISE the javascript code shown in Addendum B was used.

3.6.1 Scalability analysis

To analyse the effect of increasing network size on system performance, the evaluation method included three (3) ten (10) minute simulations running at 200% speed, each with 16, 25 and 36 nodes respectively in a grid topology. In this first setup the evaluation does not compare with SDN-WISE is it exhibits scalability problems due to the high control traffic associated with it [132]. This first simulation set yielded the results necessary to evaluate Packet Delivery Rate (PDR), packet delay in milliseconds and control traffic while having each node send a test message to the controller and hence it also acts as a sink in this case. In IT-SDN this message is sent periodically every 120 seconds while in SDNMM this message is sent aperiodically based on the Sampling Interval (SI) equation shown in (3.1). Table 3.2 shows the overall simulation specification.

Results were obtained while running baseline IT-SDN (original SDWSN platform) firmware and also while running SDNMM firmware (SDNMM) then evaluated by performing a comparison to analyze the effect of building SDNMM on top of the IT-SDN platform. The results obtained are presented in Figures 3.12, 3.13 and 3.14 respectively.

Table 3.2. Summary of simulation setup specifications, taken from [125], © 2019 IEEE.

Parameter	Value Specification
Controller	Sky mote
SDN-enabled nodes	Sky mote
Transceiver	CC2420
Transmit power	0 dBm (1 mW)
Node to node distance	25 m
Node transmit range	60 m
Node alignment	linear
Cooja simulation speed	200 %
Simulation time	10 mins
Protocol	IEEE 802.15.4

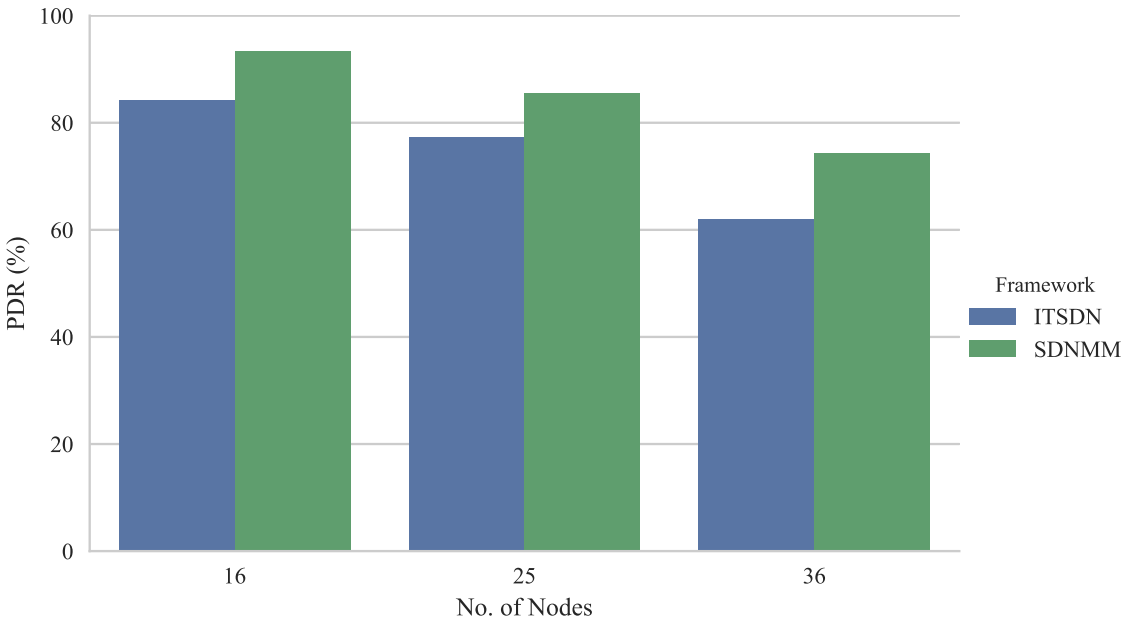

Figure 3.12. Packet delivery rate, taken from [125], © 2019 IEEE.

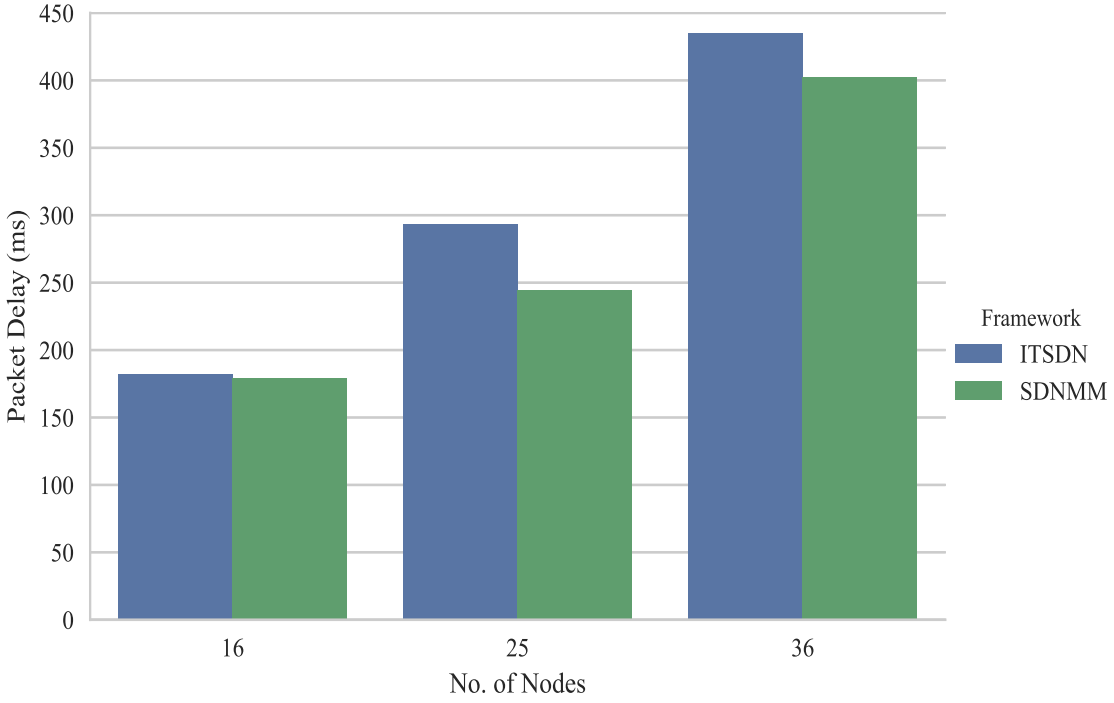

Figure 3.13. Data packet delay, taken from [125], © 2019 IEEE.

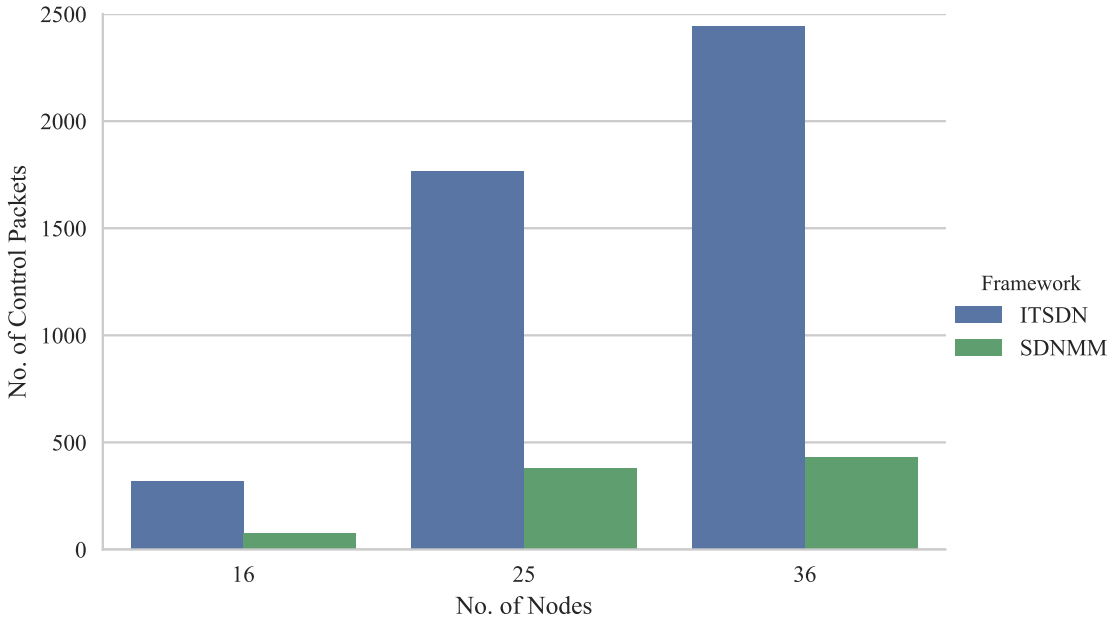

Figure 3.14. Total control packets generated, taken from [125], © 2019 IEEE.

3.6.2 Analysis based on topology

A second set of simulations were used to perform an analysis on the effect of varying network topologies on the performance of SDNMM compared to SDN-WISE and ITSDN. 16 nodes were placed in a grid, mesh and ring topology as shown in Figures 3.15, 3.16 and 3.17 respectively (based on a 16 node setup with the controller shown by ID:1 also acting as a sink) for this evaluation and the packet delivery, delay and number of control packets measured in each case. Figures 3.18, 3.19 and 3.20 show these results respectively.

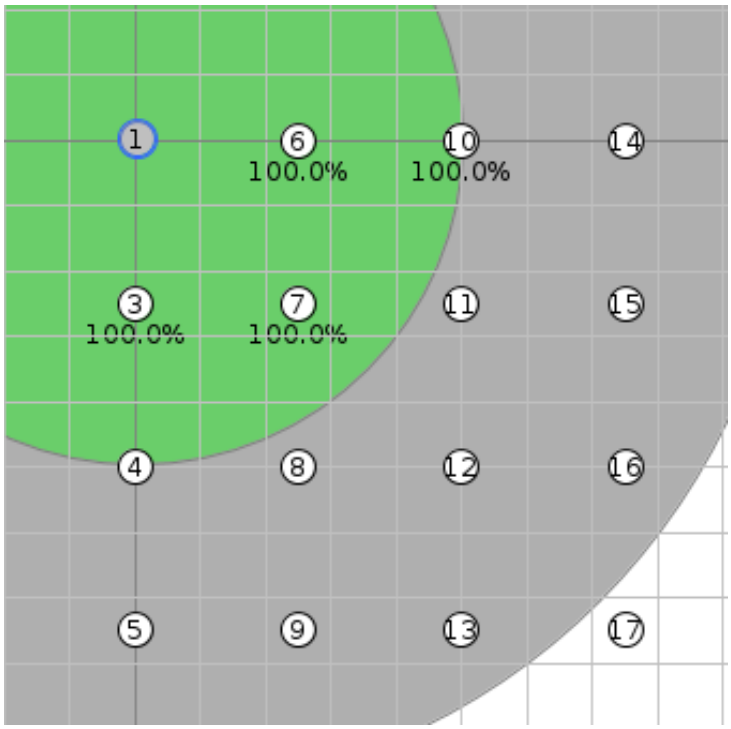

Figure 3.15. 16 node grid topology.

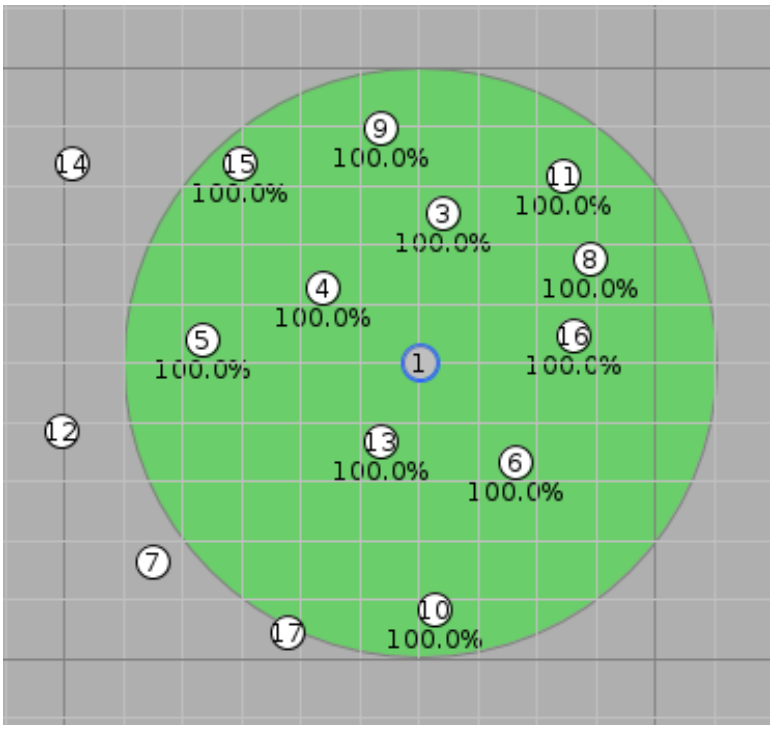

Figure 3.16. 16 node mesh (random) topology.

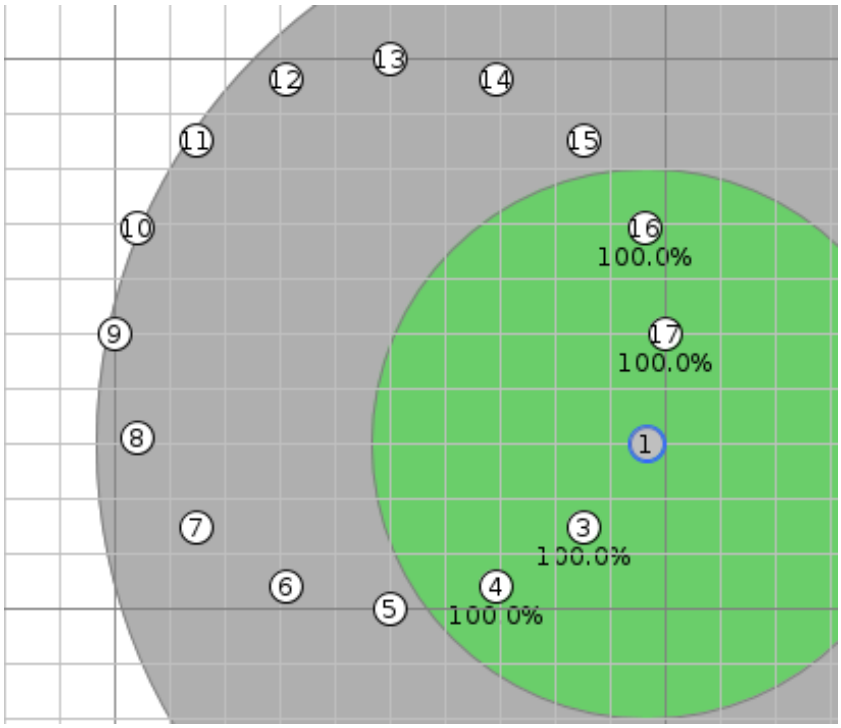

Figure 3.17. 16 node ring (chain) topology.

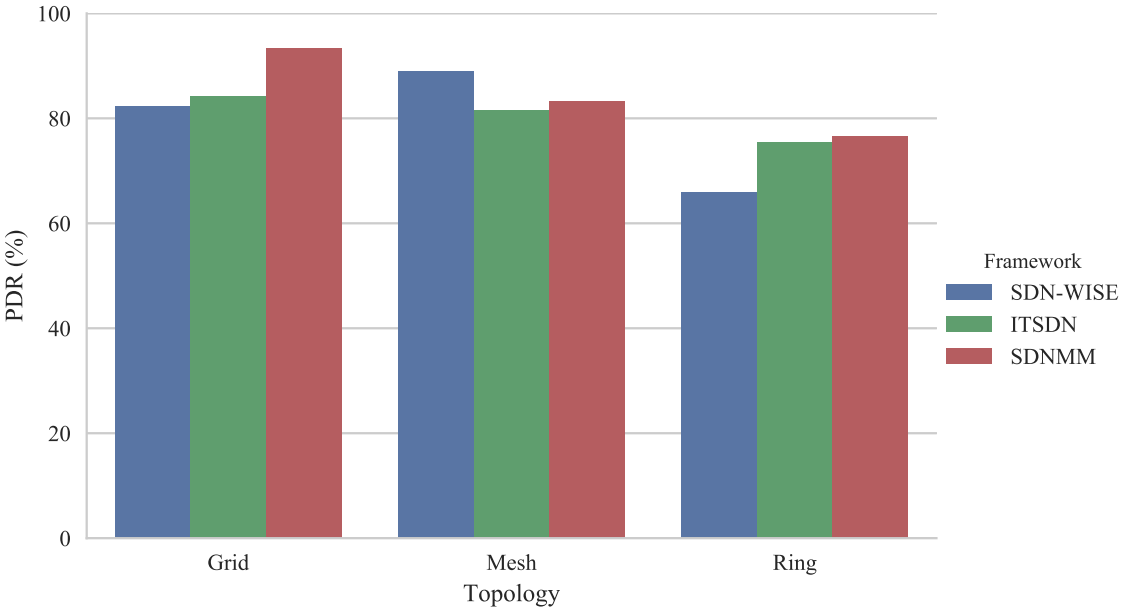

Figure 3.18. Topology packet delivery rate, taken from [125], © 2019 IEEE.

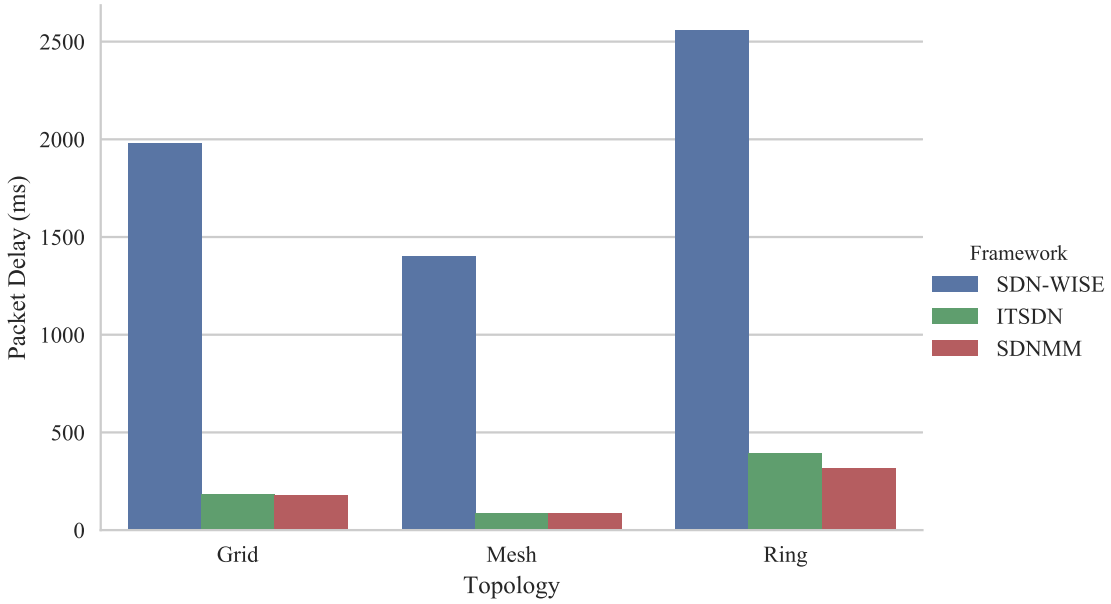

Figure 3.19. Topology packet delay, taken from [125], © 2019 IEEE.

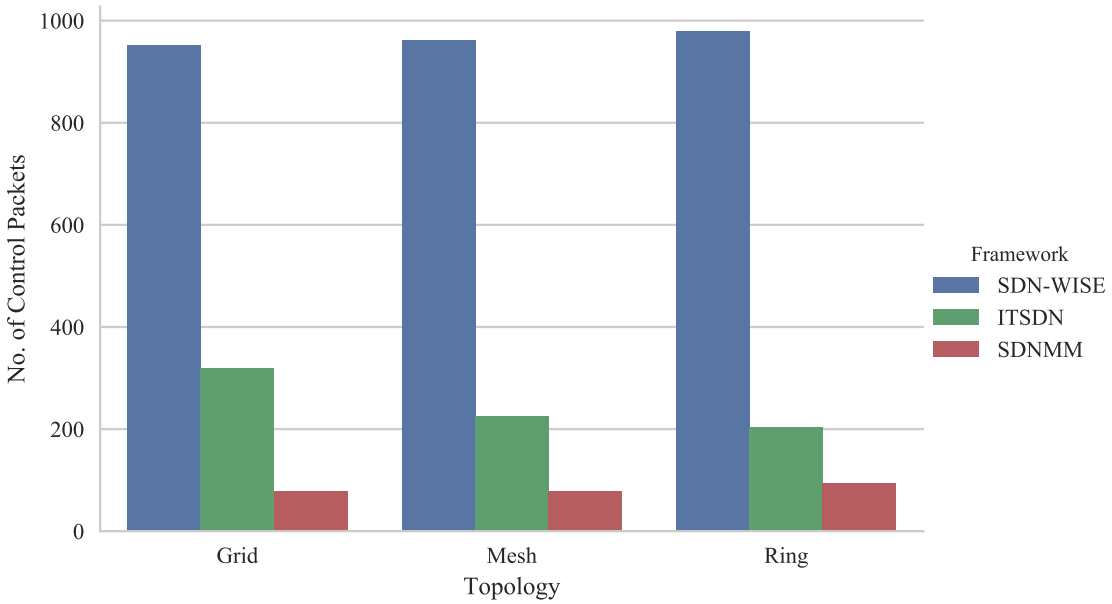

Figure 3.20. Topology control traffic, taken from [125], © 2019 IEEE.

3.6.3 Energy efficiency evaluation

The last simulation set was used to evaluate the impact on energy consumption of implementing SDNMM compared to SDN-WISE and baseline IT-SDN in a 16 node grid setup using powertraceK [131] as an energy measurement tool. The results obtained showed the total energy consumed by the SDN-enabled nodes in the data plane over a period of ten (10) minutes.

The energy model used is based on four power modes: transmit E_{tx}, listen Erx, CPU E_{cpu} and LPM E_{lpm}. E_{tx} represents the energest type time in ticks when the radio is transmitting, Erx the energest when the radio is listening, E_{cpu} represents the active computational power consumption also in ticks and E_{lpm} represents the energest type time for low power mode operation. Given the energy mode current values and voltage rating for the transceiver in the specification sheet, the individual state energy consumption in joules is shown in Equations (3.5), (3.6), (3.7) and (3.8). Dimensional analysis of the equations results in the unit watt-second which is a derived unit of energy equivalent to a joule. Equation (3.9) shows the resulting total energy consumption.

$$W_{tx} = \frac{E_{tx}\, I_{tx}\, V}{RTIMER\_sec}\, Joules \tag{3.5}$$

$$W_{rx} = \frac{E_{rx}\, I_{rx}\, V}{RTIMER\_sec}\, Joules \tag{3.6}$$

$$W_{cpu} = \frac{E_{cpu}\, I_{cpu}\, V}{RTIMER\_sec}\, Joules \tag{3.7}$$

$$W_{lpm} = \frac{E_{lpm}\, I_{lpm}\, V}{RTIMER\_sec}\, Joules \tag{3.8}$$

$$W_{total} = W_{tx} + W_{rx} + W_{cpu} + W_{lpm}\, Joules \tag{3.9}$$

where $W_{tx}, W_{rx}, W_{cpu}, W_{lpm}$ represent energy consumption in transmit, listen, active cpu and low power mode states respectively. V is the rated voltage of the transceiver in volts. $I_{tx}, I_{rx}, I_{cpu}, I_{lpm}$ are the rated currents of the transceiver in amperes during transmit, listen, active cpu and low power mode operations respectively. $RTIMER\_sec$ is the energest type tick frequency in ticks/second.

Figure 3.21 shows the plot of energy consumption in that period. The resulting computational overhead of SDNMM was evaluated by analyzing the computation cost as a percentage of the total energy cost in that time period (10 minutes). As depicted in Figure 3.22, SDNMM shows a reduced computation cost compared to both SDN-WISE and IT-SDN furthermore, it also contributes to less than 1% of the total energy cost which is mostly due to radio communications.

3.7 DISCUSSION

The results obtained show that implementing SDNMM improves the performance of the IT-SDN platform it is developed on in several key aspects. We also observe significant performance improvement over a comparable SDN-WISE framework setup. The modular management being proposed allows the implementation of a resource allocation module capable of adapting the packet rate based on resources available and also improving task allocation based on node capability. The direct result of which is reduced congestion which is reflected by the improvements in delay and control traffic levels. These ultimately result in improved packet delivery rates. Firstly, in Figure 3.12 SDNMM shows a notable improvement in packet delivery rate even as nodes are increased from 16 nodes to 25 nodes and then to 36 nodes mainly due to reduced traffic congestion hence fewer packet losses due to collisions. Figures 3.13 and 3.14 respectively show a significant reduction in packet delay and control traffic compared to IT-SDN in each case. Predetermined actions based on context in the resource allocation minimize

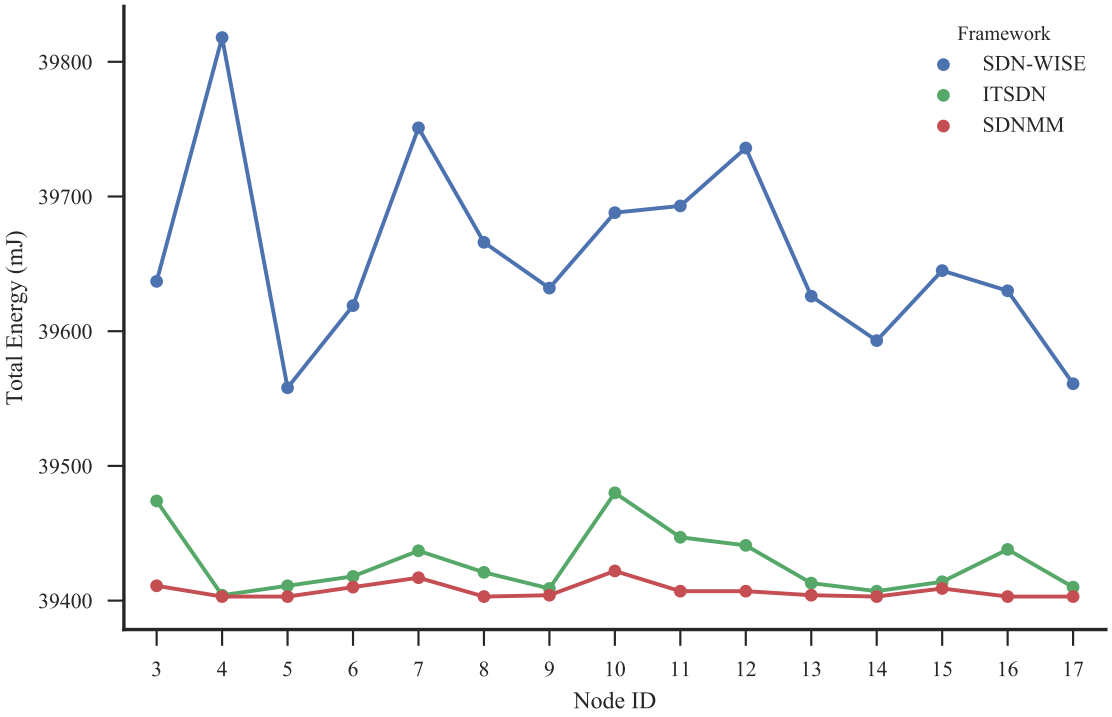

Figure 3.21. Energy consumption analysis, taken from [125], © 2019 IEEE.

delay and resulting control overhead. In comparison to SDN-WISE, the high control traffic associated with the framework hindered the simulation of 25 and 36 nodes for the set time period as the network activity would exceed allowed memory usage at those network sizes.

The SDNMM resource allocation module also showed improved performance in different topology setups. In Figures 3.18, 3.19 and 3.20 the adaptive data rate based on battery level introduced minimized congestion leading to improvement in packet delivery rate with delay and control traffic being kept at a minimum. This resulting improvement in delay and control traffic can also be alluded to the high correlation that exists between the amount of data traffic to be processed for sending to the sink or controller and the associated control overhead and delay.

In Figure 3.18 however, SDN-WISE showed around 6% improvement in packet delivery over SDNMM in the mesh topology due to the improved neighbour discovery in SDN-WISE which comes at the expense of a higher control traffic and packet delay. SDNMM still showed a considerable 83% while reducing the delay and control traffic by over 90% in the same mesh topology.

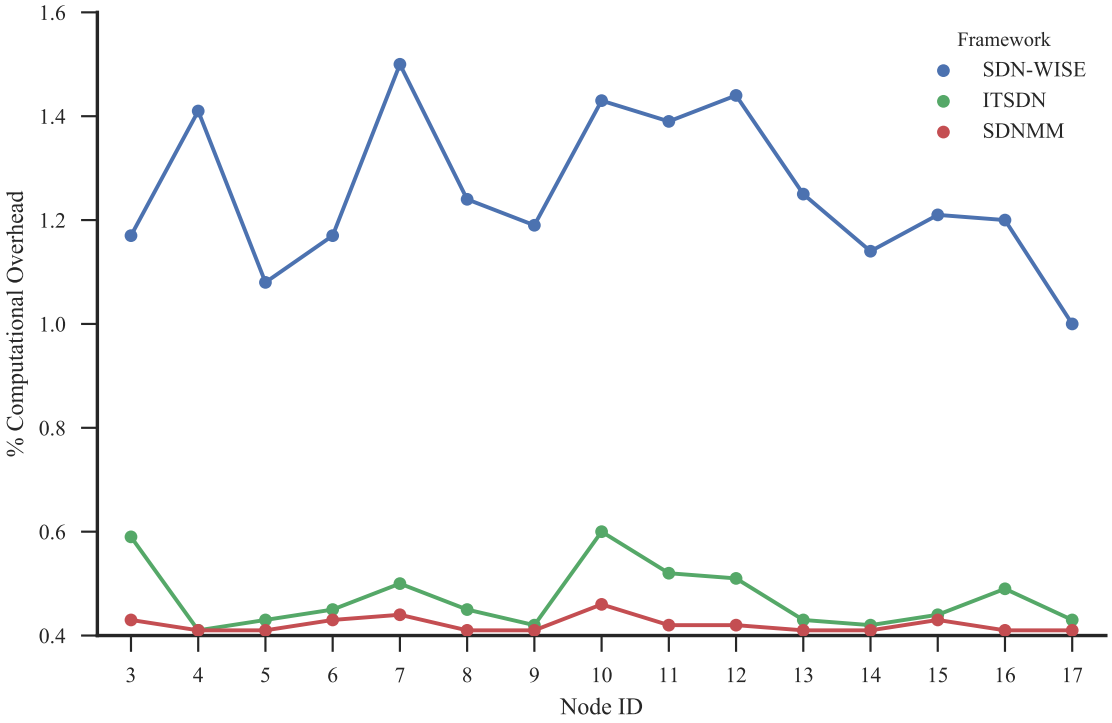

Figure 3.22. Computational overhead analysis, taken from [125], © 2019 IEEE.

Figures 3.21 and 3.22 depict the energy consumed by each SDN-enabled node in the 16-node grid setup and the resulting computational overhead respectively. The graphs show that SDNMM results in a decreased energy consumption compared to IT-SDN as all points reflecting SDNMM energy consumption fell below that of IT-SDN only. The computational overhead analysis shows the complexity cost of the algorithms on the node processor. All three frameworks presented less than 1.6% energy consumption due to computational cost, however the proposed SDNMM system demonstrated a computational cost that was even less than 0.5% of the total energy consumed. This depicts a much reduced complexity of algorithm implementation. The data sampling enabled in the MSI was mainly responsible for this increase in energy efficiency. We also notice that this is a promising phenomenon of the resource allocation module considering no dedicated energy management module was integrated and the extra energy consumed from context report generation did not impact the overall energy consumed extensively. Compared to SDN-WISE in the same 16 node grid setup, SDNMM consumed less energy in the same time period showing an improvement in energy efficiency. This improvement is also mainly due to the decreased traffic generation and transmission rates leading to a lesser period of active cpu, transmit and receive time which consume most of the node energy resources.

In terms of other SDN-based management systems, we can analyze and compare features as shown in Table 3.3. Usability as a management system is dependent on features such as the application use case, modularity, scalability, and availability of the development platform. Compared to other available SDN-based management systems, SDNMM has the advantage of providing a generic management approach based on its modularity feature that is developed on an open-source and widely available platform. SDNMM also provides an opportunity for large-scale implementation as the system framework allows for network scalability.

Table 3.3. Feature comparison of SDN-based management systems, taken from [125], © 2019 IEEE.

System	Application	Modularity	Scalability	Platform
Smart [43]	Generic	NO	NO	-
Soft-WSN [51]	Device and topology	NO	NO	Proprietary
6LoWPAN [127]	Energy	NO	NO	Open-source
Shsec [133]	Smart home security	YES	-	Proprietary
SDNMM	Generic	YES	YES	Open-source

3.8 SYSTEM CHALLENGES

Much as the results show a performance improvement of the modular management system over comparable SDWSN frameworks mainly due to the resource allocation module. The module enabled placing a limit on traffic generation based on resource availability however, there are a few challenges affecting this implementation approach that need investigation to further improve the efficiency of this implementation. First and foremost, this work proposes a modular management system based on context awareness which results in increased occupancy of the in-band traffic channel by management packets. In the sample resource allocation module implemented to evaluate system feasibility, context data packets are sent to the controller as part of control traffic to form a context data knowledge base. A major challenge to implementing such a system in SDN-based WSNs is the limited in-band channel bandwidth that is used for both control traffic and data traffic resulting in increased delay associated with the increased controller response time.

Another challenge associated with increased generation of control traffic packets is an equal increase in energy consumption by the nodes in the data plane. Consider a management application such a security management module that does not take into account aspects of resource allocation or energy, the resulting increase in energy consumption can prove to be a challenge. In addition, some forms of context data required by the management module may require further network intelligence to acquire which would result in increased overall delay and energy consumption. It is worth mentioning here that while research aims to improve several aspects of WSN performance, QoS parameters have the tendency to exhibit performance trade-offs such as gaining improved PDR and energy efficiency may result in increased overhead traffic and delay. This aspect of performance trade-off is a major challenge affecting the research area evaluated in this study.

Solutions to the challenges mentioned in this section can be resolved by load sharing using distributed controllers and implementation of overall control traffic reduction techniques. This study also explores aspects of this in the next chapter so as to efficiently manage the use of the already limited traffic channel in software-defined wireless sensor networks. To tackle the aspects of performance trade-offs the system design requirements can be considered and optimized based on that.

3.9 CHAPTER SUMMARY

It has thus far been drawn that SDN brings the aspect of centralised control and management efficiency to wireless sensor networks as highlighted in Section 3.2 of this chapter. To better take advantage of this management flexibility, a generic and modular management framework has been proposed in Section 3.3. Such a framework promises rapid prototyping of management methods by integrating them as modules based on a unified context knowledge base. We also observed that to allow generic use, the system has been built on an open source and readily available platform IT-SDN. Section 3.4 presented a management service interface that was based on a state machine architecture and a context data knowledge base providing a common interface for management modules. The context knowledge base was dependent on data population from context packets carrying context data from SDN-enabled nodes in the data plane. In Section 3.5 the implementation strategy based on a built resource allocation module was provided including a discussion on the module integration in the proposed modular system. The system was then evaluated with this resource allocation module and compared to other frameworks without the proposed modular aspect in Section 3.6. In Section 3.7 a discussion of the evaluation results

was presented. The implemented resource allocation module showed improvement for SDNMM in packet delivery rate and delay compared to baseline IT-SDN for network sizes of 16, 25 and 36 nodes respectively. The results also showed a general performance improvement over both baseline IT-SDN and SDN-WISE for a 16 nodes setup configured in grid, mesh and ring topologies respectively. In terms of energy efficiency, all the SDN-enabled nodes(15 nodes) in the 16 node grid setup consumed significantly less energy for SDNMM compared to SDN-WISE and also showed an improvement in energy efficiency over the IT-SDN platform. Section 3.8 discussed challenges of the SDNMM system. A significant bottleneck affecting the performance of SDNMM was discovered to be due to increased overhead traffic resulting from the nature of the modular management concept. Populating the context knowledge base required generation of extra control traffic for carrying context data from the data plane to the controller. Although, SDNMM showed lower control overhead compared to both IT-SDN and SDN-WISE, the fact that the system generates extra control traffic it is necessary to investigate methods of reducing accompanying control traffic in the allocated traffic channel provision in SDN-based WSNs. Such techniques would further improve network performance and quality of service. In general, the SDNMM system introduces a new niche in IoT management that requires the development of various modules for security, topology and QoS management for instance, all with the end goal of improving the efficiency of network performance and reliability.

CHAPTER 4 CONTROL MESSAGE QUENCHING FOR SDN-BASED WSN

4.1 CHAPTER OVERVIEW

In the previous chapters, a context-based modularity management system has been proposed and evaluated. However, a major backbone operation of this kind of system is the implementation of extra control traffic carrying context data to the control plane. This poses a potential bottleneck towards the quality of service in SDN-based WSN applications due to the in-band traffic channel available for both data and control traffic. As a result, this calls for exploration into potential techniques of minimizing control traffic in general which in turn would promise to leave room for extra management packets for applications similar to that discussed in this study. This chapter opens this discussion by exploring Control Message Quenching (CMQ) for software-defined wireless sensor network applications with a focus on solutions for WSNs based on the 6LoWPAN (IPV6 Low-power Wireless Personal Area Networks) protocol.

The rest of this chapter is organised as follows. In Section 4.2, a background to control message quenching in SDN-based WSNs is given including a discussion on available CMQ techniques. Section 4.3 delves into control message quenching based on elimination of flow setup request packet duplication while Section 4.4 introduces and discusses a two step CMQ algorithm that focuses on reduction of neighbour reports in a 6LoWPAN SDWSN. In Section 4.5, the simulation setup and associated performance results of the proposed neighbour discovery based CMQ algorithm are presented. Then Section 4.6 discusses in detail the performance results of the proposed neighbour discovery CMQ algorithm while Section 4.7 creates the chapter summary.

4.2 BACKGROUND ON CMQ FOR SDWSN

The operating principle of SDN-based WSNs (SDWSN) is based on the provision of flow rules and other related traffic (control traffic) to and from the data infrastructure in the data plane which in the case of WSN implementation has to occupy the same in-band traffic channel with data plane traffic [11, 37]. This is unlike in computer networks where a dedicated out-band control channel can be provisioned and hence overhead control traffic is a potential bottleneck to packet delivery, latency and controller responsiveness in SDWSN. In this regard, there has been a general interest in the SDWSN research industry to effectively reduce the levels of overhead control traffic also referred to as Control Message Quenching (CMQ) [68].

The SDN implementation in WSN enables global control of network devices via a set of control policies handled by the southbound API between the control plane and the data plane. The control policies are categorised into packet types based on function. Figure 4.1 shows the inherent control traffic packets available in the architecture of this SDN-based operation in WSNs.

The various control traffic packet types can be categorised as:

(a) Flow setup: This type of traffic flows from the control plane to the data plane and is composed of flow rules required for routing in the data plane. A flow setup packet can be issued at bootstrapping or in response to a flow setup request.

(b) Flow setup requests: In a typical Sensor OpenFlow (SOF) setup [37], when a packet arrives at an SDN-enabled sensor node, the packet source address or flow ID is cross-checked with the flow table rules in the SDN-enabled node. If a match is found, the packet is processed according to the routing actions in the table but if a mismatch occurs a flow setup request (flow request) packet is issued to the controller to send an associated flow setup.

(c) Neighbour reports / Beacon Messages: Routing requires up to date localization of sensor nodes in the data plane and regular neighbour reports or beacon messages are required to obtain a good estimate of node locations. A neighbour report contains information such as neighbour address and Estimated Transmission Count (ETX). The frequency of this kind of control packet is dependant on the underlying neighbour discovery protocol implemented in the SDN-based WSN.

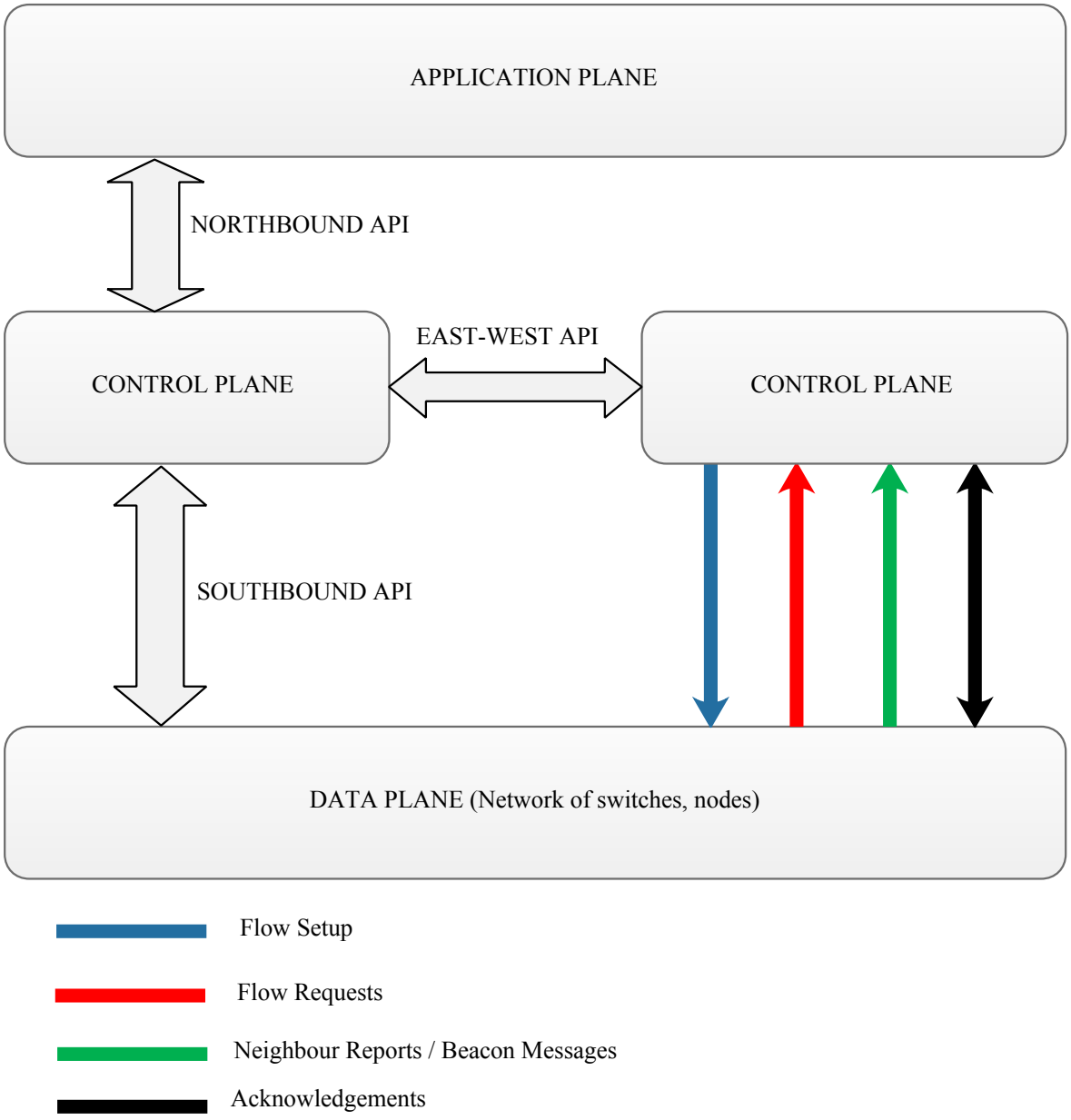

Figure 4.1. Control traffic in SDWSN.

(d) Acknowledgements: This control packet type can be used to confirm the successful delivery of a policy or data packet. Depending on the design requirements, the controller or node may require an acknowledgement packet as part of the network management policy and control.

While control traffic plays an important role in enabling the management flexibility that SDN provides, overhead control traffic has been identified as a drawback in obtaining the desired QoS. In TinySDN [11] authors mention that increased in-band control traffic places a limit on the data transmission

rate potentially leading to increased packet delay. Bera *et al*. [51] find in their implementation of an SDN-based WSN that the control traffic generated is higher than in traditional WSNs. As part of the authors' conclusion, they propose further work be done in minimizing the resulting control message overhead. Lasso *et al*. [127] also found the need to further examine network control message reduction as part of their implementation of an SDN-based IoT framework for 6LoWPAN.

In computer network based SDN, control message quenching has mainly been adopted to increase controller responsiveness not so much for bandwidth requirements due to the possibility of providing a dedicated control channel. Popular solutions for CMQ in such networks involve use of multiple controllers to share the control traffic load and hence improve controller responsiveness. Contributions such as DevoFlow [134] and that by Yu *et al*. [135] are based on this approach. One of the early adopters to implement a CMQ method that is simple and does not require modification of the control plane include that of Luo *et al*. [68]. They use a reference memory to prevent the generation of duplicate flow setup request packets and notice improved controller responsiveness upon implementation. Others such as Obadia *et al*. [136] use a greedy heuristic algorithm that calculates the cost of control traffic based on awake nodes and available controllers.

4.2.1 Categories of control message quenching

To open a discussion on minimization techniques for control traffic in SDN-based WSNs, this study identifies the main categories for control message quenching and highlights the contributions therein.

4.2.1.1 Controller distribution / Traffic load sharing

The use of distributed controllers in SDN-based applications is becoming increasingly popular as a form of control message quenching. Multiple controller integration allows for sharing of the overall traffic load and depending on the implementation can significantly improve controller responsiveness and reduce packet latency. Reliability is another direct advantage of this approach as distributed control would still result in service continuity even during the downtime of some of the controllers in the network which would not be the case with a single centralized controller [137].

The multiple controllers are usually designed in a hierarchical manner with a global controller coordinating and synchronising multiple local controllers which may each be assigned to a cluster of sensor nodes. The global controller plays an active role in giving partial authorisation to the local controllers to perform routing and flow operations locally. There have been implementations of this kind of CMQ in SDWSN literature. Oliveira *et al*. [11] for example use distributed controllers in their TinySDN implementation to tackle challenges such as communication latency, overhead control traffic and limited energy supply. Issues of controller placement have also been investigated in TinySDN with a noticeable improvement in performance when the controllers are placed nearer to the end devices (sensor nodes). Galluccio *et al*. [12] in SDN-WISE provide support for distributed controllers to improve controller response times while highlighting the need for security in the controller software. It has been observed that the use of multiple controllers in an SDN-based WSN provides a promising solution however it offers challenges in terms of controller synchronisation and also requires redesigning the control plane.

4.2.1.2 Control aggregation / Fusion

Data aggregation and/or data fusion has been used severally in reducing data traffic both in traditional WSNs and SDWSNs. Techniques such as taking the average value of a measurand like temperature, humidity etc. and sending only one average value of a cluster area to the sink instead of multiple readings have been implemented. However, the idea of using this aggregation or data fusion method in reducing control messages in SDN-based WSN is still very niche and limited in terms of application. This form of control message quenching can be approached either by modifying flow table rules to combine multiple flow instructions into a single flow packet or by simple concatenation of packets that are sent along the same paths. The various weights of the elements being fused in the aggregation process needs to be considered.

Friedman *et al*. [138] open a discussion on modification of flow table rules to enable data aggregation in their proposed SDN-based WSN architecture. Although, in their discussion, a demonstration of fusion of data traffic such as temperature and humidity is given and not specifically control traffic packets. It can be observed and concluded that the implementation of such a technique for minimizing control traffic may require an algorithm to group common flow table items and discard duplicate data from multiple related flow packets.

In SDN-WISE an early attempt at packet concatenation to minimize control traffic is made as part of the In-Network Packet Processing (INPP) layer. Galluccio *et al*. [12] use the INPP layer to concatenate small packets being sent along the same route and hence minimizing the overhead. However, it is clear that there is a need for further network coding to effectively target control traffic minimization in the implementation of SDN-WISE.

The general concept of using data aggregation/fusion to reduce the number of control packets arriving at the controller and as a result improving controller response time is illustrated in Figure 4.2.

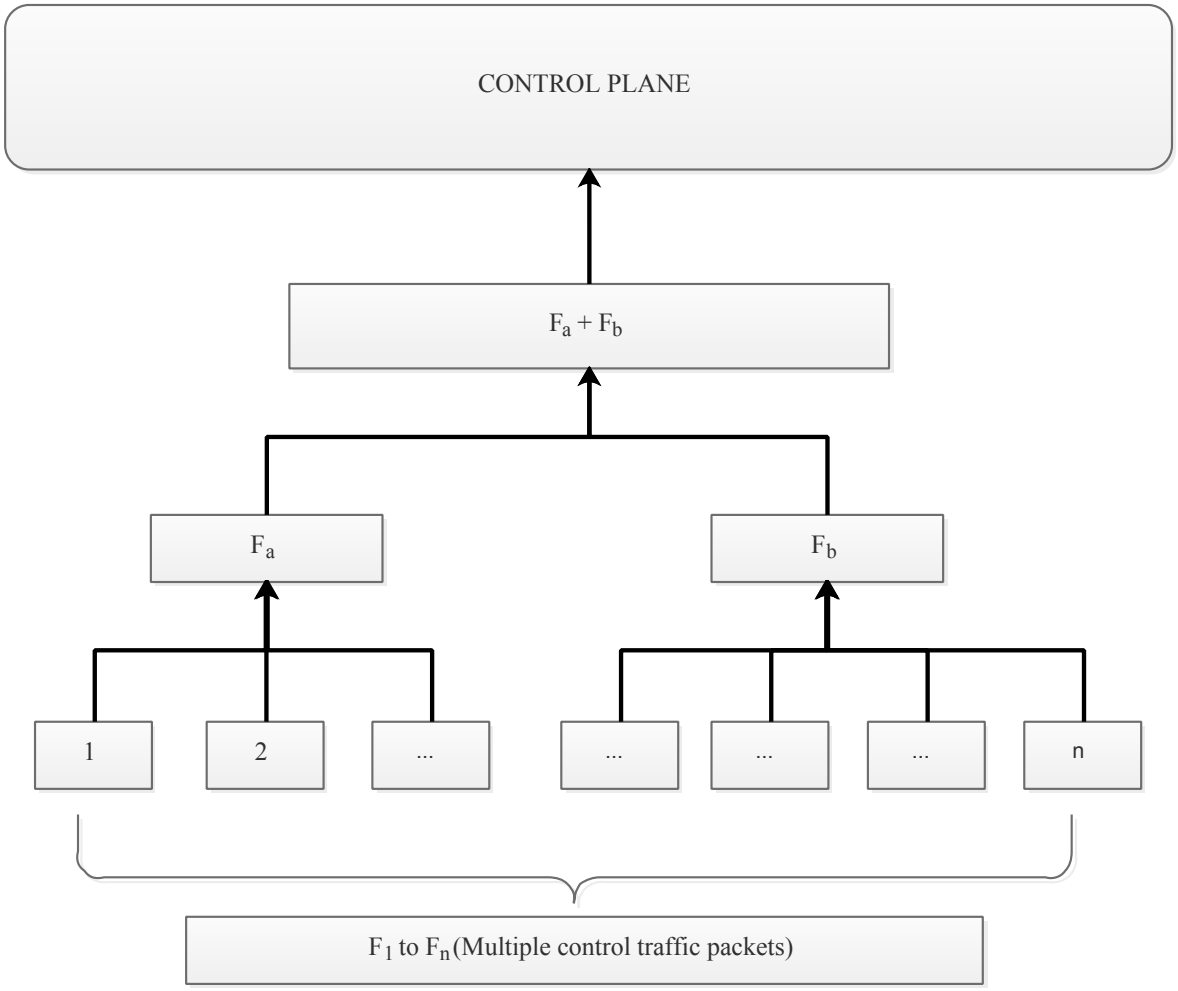

Figure 4.2. Control flow aggregation/fusion concept.

4.2.1.3 Heuristics / Cognitive approach

Network intelligence and learning approaches can be used to ensure a minimum and optimal control traffic rate. Calculations performed can ensure that the cost of transmitting control data are not in excess of what is required based on available and awake devices in the network at that particular time. The feasibility of this mechanism in reducing control traffic has been demonstrated for conventional SDN by Obadia *et al*. [136]. While network learning and heuristics promise the potential of optimizing control traffic, this technique is still very niche in SDN-based WSNs. Additionally, active-sleep scheduling techniques can further be used to minimize the amount of control traffic operations required in a network to meet the desired QoS at a particular time.

4.2.1.4 Collect-based approach

A more direct approach to reducing control traffic is based on locally preventing excess collection and transmission of control policies. The collect protocol for neighbour discovery, for example, can be adjusted to send fewer neighbour reports or beacon messages to the controller. Galluccio *et al*. [12] mention how the frequency of topology discovery packet generation and associated neighbour information reports affect the traffic overhead in SDN-WISE. However, it is important to note that adjusting the collect protocol in a bid to reduce control traffic overhead affects other network performance parameters and as such, this trade-off should be taken into account during design and implementation.

Another simpler approach of reducing control messages is by preventing transmission of duplicate control packets which may result from recurrent requests for the message to be sent to the controller. Luo *et al*. [68] identify a typical scenario where duplicate flow setup requests are prevented from being generated. Flow setup requests are generated upon table mismatch and once a similar packet (same source/destination address or flow ID) arrives at a node another flow setup request of the same type will be generated as long as a flow setup from the initial request has not arrived from the controller to update the node table. A similar case can also occur on the controller side during flow setup and a notable solution to preventing this form of duplication is by using a reference memory that can either be external or internal to the devices. Upon initial flow setup request, the associated source/destination address can be stored in a dynamic memory list and this list can be referred to thereafter whenever

a mismatch occurs. If the source/destination pair details do not match anything in the list, a flow setup request is generated otherwise the flow setup request is suppressed and the packet is enqueued awaiting to be processed when the original flow setup update arrives. Asaduzzaman *et al*. [139] recently proposed a similar technique but based on a separate and external memory reference. However, in SDN-based WSN this approach to CMQ still remains widely unexplored. Figure 4.3 shows the basic implementation setup of this control traffic reduction method.

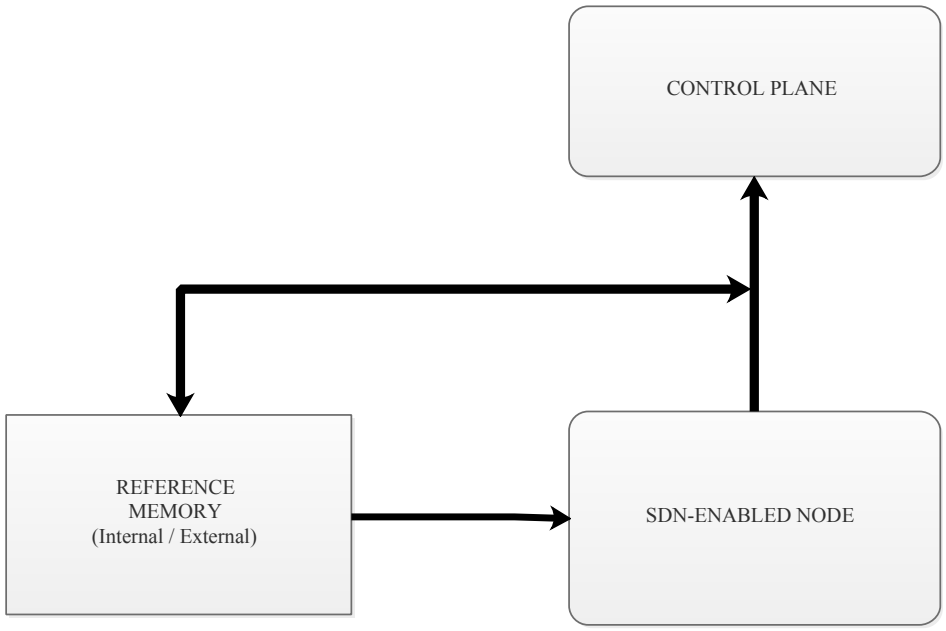

Figure 4.3. CMQ based on reference memory.

Table 4.1 shows a summary of a qualitative comparison of the control message quenching techniques discussed in this section. We compare what modifications to the SDN architecture are required, the complexity of implementation and the current state of the application in the SDWSN field.

Table 4.1. Comparison SDN-based WSN CMQ techniques.

Method	Modification	Complexity	Application
Distributed controller	Control plane	Medium - High	Common
Control aggregation	Flow table/ flow rules	Medium - High	Limited
Collect	Data or control planes	Low	limited
Heuristics	Variable	Medium - High	Very limited

4.2.2 Design requirements for SDWSN-based CMQ

In a bid to reduce control overhead in SDWSNs, it should be taken into consideration that there exists a trade-off between the level of control traffic minimization and the design requirements of SDN-based WSNs such as packet delivery rate (PDR), latency and energy efficiency. In terms of using distributed controllers to quench control traffic, the implementation should account for the resulting packet delay in using multiple controllers. Techniques to improve synchronization and reduce delay need to be investigated and implemented. Kobo *et al*. [129] propose the use of fragmentation to reduce the end to end delay in multiple controller setups for SDWSNs.

In scenarios where topology discovery packets such as beacon messages are lowered to reduce overhead, the design requirement should also take into account the complexity of the network topology and how much the network parameters and node locations fluctuate. Rapid variations in network parameters require frequent and up to date topology discovery messages. The level of control traffic minimization should be optimized against the packet delivery rate and delay. Energy efficiency should also be taken into account as poor topology discovery may result in poor and energy inefficient packet routing and link failures.

While aggregation methods may result in improved energy efficiency, the time complexity in implementing the method may lead to packet delay. This is also the same for heuristic methods which involve lots of complex computations.

4.2.3 Prospects in CMQ for SDN-based WSN

Prospects in this research area require further investigation into methods of control message quenching involving heuristics and machine learning. There is a need to implement mathematical optimization techniques that increase the design requirement yield based on the trade-off that exists between overhead reduction and the desired QoS in SDWSN. There is a need to implement aggregation or data fusion techniques for control flow traffic with a clear outline of how the flow tables and rules can be modified to achieve this. While researchers have begun proposing more efficient and low latency ways of implementing distributed controllers in SDN-based WSNs there is still a need for more work in this area in terms of synchronization and also security. The efficiency of flow table rules in the data plane

can also further be investigated especially the aspect of nodes sharing and synchronizing rules that have already been issued by the controller instead of each node making a request whenever need arises while it is possible that another node in the plane may have the required flow rules at that particular time.

As part of the research prospects highlighted in this section, the rest of the chapter investigates and implements CMQ techniques based on flow setup requests and neighbour discovery messages.

4.3 CMQ BASED ON FLOW SETUP REQUESTS (FR-CMQ)

A simple technique to minimize control messages is to reduce or prevent duplicate flow setup requests based on the FR-CMQ algorithm. The FR-CMQ algorithm is a control message quenching algorithm with the aim of reducing or preventing duplicate flow setup request packets going to the controller. Consider a scenario where a packet from a particular source arrives at a node and a flow table lookup returns a mismatch. The direct result would be a generation of a flow setup request packet for that packet source-destination pair. If a similar packet arrives at the node before the corresponding flow setup arrives, another request would be sent to the controller resulting in duplication of request packets. Hence the FR-CMQ algorithm shown in Algorithm 1 uses a source-destination pair reference list stored in dynamic memory to cross-check calls for flow setup requests. If a request has already been made, the packet is simply queued until the associated flow setup arrives and then it is processed accordingly.

Algorithm 1 FR-CMQ algorithm

```
1:  packet arrives at node                          ▷ look up flow table
2:  if match==1 then process packet
3:  else
4:      if S&D ∈ List == 1 then                     ▷ refer to source/destination list
5:          enqueue packet                          ▷ await original flow setup
6:      else
7:          insert S&D                              ▷ update dynamic list
8:          send flow setup request
9:      end if
10: end if
```

In terms of implementation and evaluation of the FR-CMQ algorithm, the IT-SDN platform makes a suitable and effective tool. Contiki lists can be used to create a reference memory for source and destination addresses each time a flow setup request is generated. The available Contiki Cooja simulator provides a suitable platform to evaluate the algorithm in association with the integrated Qt-based controller. Other SDN-based WSN platforms are available such as SDN-WISE [12] however, IT-SDN provides better open-sourceness and developer support for such forms of implementation.

4.4 NEIGHBOUR DISCOVERY CMQ FOR SDN-BASED WSN

Thus far, control message quenching based on reducing duplicate flow setup requests has been introduced. However, another promising control message quenching technique based on minimizing and optimizing neighbour discovery or beacon messages in SDN-based WSNs is proposed in this study. A typical neighbour report rate for SDN-based 6LoWPANs with an ETX-based neighbour discovery protocol is about one (1) neighbour report packet per node per minute [124]. As the network scales up in size the number of neighbour reports increase considerably and hence affects controller responsiveness. Therefore, an algorithm (ND-CMQ) which focusses on the 6LoWPAN protocol is proposed to effectively limit the number of neighbour report messages without significantly affecting the performance of the network in terms of packet delivery, energy consumption, and packet latency.

4.4.1 6LoWPAN SDWSN neighbour report process

The neighbour report process is triggered by a neighbour announcement protocol that is called periodically depending on a set time interval. This time interval is referred to as the proactive probing interval (PPI). Setting a low PPI improves the accuracy of the current neighbour status but may result in more frequent neighbour reporting if an algorithm is not in place to curb report duplication. The PPI is controlled by a timer which triggers an announcement when it expires and restarts in a timer sub-process to maintain the interval.

Upon the announcement, each node collects neighbour addresses and associated metric which are stored in an updated neighbour report list table by the underlying neighbour discovery (ND) protocol. The neighbour report list is stored in a dynamic linked list memory to enable an address based access

mechanism of neighbour information. The process of sending a neighbour report to the controller as part of control traffic is handled by the SDN send process. The report list is packed into a neighbour report packet that should not exceed 127 bytes as required by the 6LoWPAN protocol after which it is enqueued and sent. The process is summarised by the flow chart in Figure 4.4.

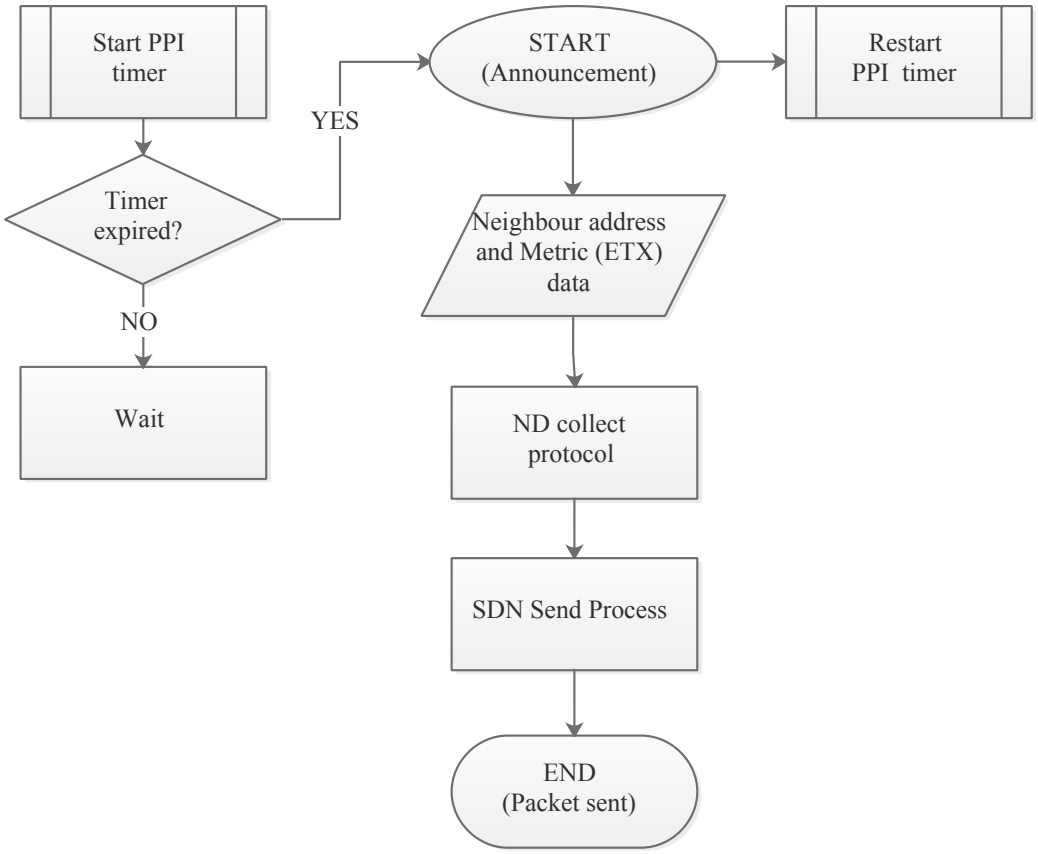

Figure 4.4. Neighbour report process.

4.4.2 Proposed ND-CMQ algorithm

To develop an algorithm for control message quenching of neighbour reports we need to consider the underlying event triggers. It is identified that the main scenarios that trigger a neighbour report is when there is a change in neighbour address or its associated metric during the neighbour announcement. The problem is even if a single change occurs either in neighbour address or metric the entire list is sent as a report resulting in redundancy.

The first solution to this is to optimize the neighbour change event process. Since the variations in ETX occur much more frequently and a possible solution is to use a threshold value necessary for

optimal routing. The lower the value the more optimal it is for routing, therefore we create a condition in the algorithm to only post a neighbour change event when the new announced ETX value is less than the current value. The algorithm should also take into consideration an upper threshold value that usually represents a broken link. It is important to post the update when the ETX reaches this max value to allow for removal of that particular edge. This would effectively reduce the neighbour report packet queue size taking into consideration that controller responsiveness is highly affected by packet queue size. In Algorithm 2, this solution is implemented as part of the ND collect protocol process stage.

The second solution is to prevent sending of duplicate neighbour information. A copy of neighbour addresses and their associated metrics can be stored in allocated dynamic memory and each time a need to send a neighbour report arises, the neighbour information can be compared with what has been already sent. In short, we traverse through the stored neighbour table looking for copies of entries to the new neighbour information, if found the entry is suppressed and not transmitted. This second solution is implemented as part of the send process stage shown in Figure 4.4. Generally, the ND-CMQ algorithm is a two-stage process that includes neighbour event filtering shown in Algorithm 2 (stage 1) and duplication suppression in Algorithm 3 (stage 2).

Algorithm 2 Neighbour event filtering

```
1:  neighbour_announcement()                          ▷ function announcing neighbours
2:  if neighbour_node address ∉ neighbour list then
3:      add neighbour address and metric to list;     ▷ update reference list
4:      post event neighbour change;                  ▷ (ev=neighbour_change)
5:  end if
6:  if neighbour_node address ∈ neighbour list then
7:      if neighbour_metric < list_metric             ▷ improved metric (lower value)
8:      || neighbour_metric==max_metric then          ▷ max is link loss
9:          post event neighbour change;              ▷ (ev=neighbour_change)
10:     else
11:         printf ("No neighbour info change");      ▷ suppress neighbour event post
12:     end if
13: end if
```

In IT-SDN [124] the neighbour send process involves sending the entire neighbour report list regardless

of unchanged neighbour information. This send process is modified to prevent filling the neighbour report packet with neighbour information that has not changed since the previous broadcast as follows. Each time neighbour information is sent successfully it is saved in a table allocated in dynamic memory for previous broadcast neighbour information (neighbour_ptable). This previous neighbour table is then used as a reference for future broadcasts to compare and look for unchanged neighbour information which we suppress. This is shown in Algorithm 3.

The next sections will now compare and evaluate how this ND-CMQ algorithm performs against the earlier introduced FR-CMQ algorithm and baseline IT-SDN. As already determined from previous chapter results, baseline SDN-WISE already has a much higher rate of control messages than baseline IT-SDN therefore within the scope of this study, the performance of the CMQ algorithms will only be compared to baseline IT-SDN in this chapter.

Algorithm 3 Duplicate entry removal

```
1:  for each neighbour address in neighbour list for broadcast do
2:      if neighbour address ∉ previous table list then        ▷ check for duplicate address
3:          add item address and metric to previous table list;
4:          concatenate to neighbour report packet;            ▷ ready for enqueue
5:      end if
6:      if item address ∈ previous table list then             ▷ compare associated metric
7:          if item metric != list_metric then                 ▷ check for duplicate metric
8:              concatenate to neighbour report packet;        ▷ ready for enqueue
9:          end if
10:     else printf ("neighbour information unchanged");       ▷ suppress information for send
11:     end if
12: end for
```

4.5 SIMULATION SETUP AND RESULTS

Both the FR-CMQ algorithm that focuses on reducing duplicate flow setup requests [68] and the proposed ND-CMQ algorithm are implemented on the IT-SDN [124] platform. The performances are compared in three states: the platform without any control message quenching algorithm (ITSDN), the platform with FR-CMQ algorithm (FR-CMQ) and the platform with ND-CMQ algorithm (ND-CMQ).

The simulation parameters are shown in Table 4.2. The simulation evaluates the performance of the control message quenching algorithms based on the resultant number of control packets, packet delivery rate, packet delay and energy efficiency. The effect of varying network topology on the performance of the CMQ algorithms is also evaluated and a scalability analysis performed on a sample topology, in this case, ring topology was used.

Table 4.2. Simulation setup specifications.

Parameter	Value Specification
Controller	Sky mote
SDN-enabled nodes	Sky mote
Sink nodes	Sky mote
Transceiver	CC2420
Transmit power	0 dBm (1 mW)
Node transmit range	50 m
Test data packet size	10 bytes
Test data transmit freq.	1 packet/min
Cooja simulation speed	200 %
Protocol	IEEE 802.15.4

4.5.1 Evaluation based on topology

The performance is first evaluated for 25 nodes arranged in three topology setups namely the ring, mesh and grid with a 10-byte test message sent to the sink every minute for a 10 minute period. It is worth mentioning here that in this setup, the network consists of a standalone controller node and sink node together with SDN-enabled nodes to maximize the potential of control traffic generation. This would enable a higher evaluation level for the control message quenching techniques.

Figures 4.5, 4.6 and 4.7 show these topology arrangements respectively. Node 1 represents the controller, node 2 is the sink and the rest of the nodes are SDN-enabled nodes.

The results obtained for the topology configurations over a 10 minute simulation in Contiki Cooja is shown in Figures 4.8, 4.9 and 4.10.

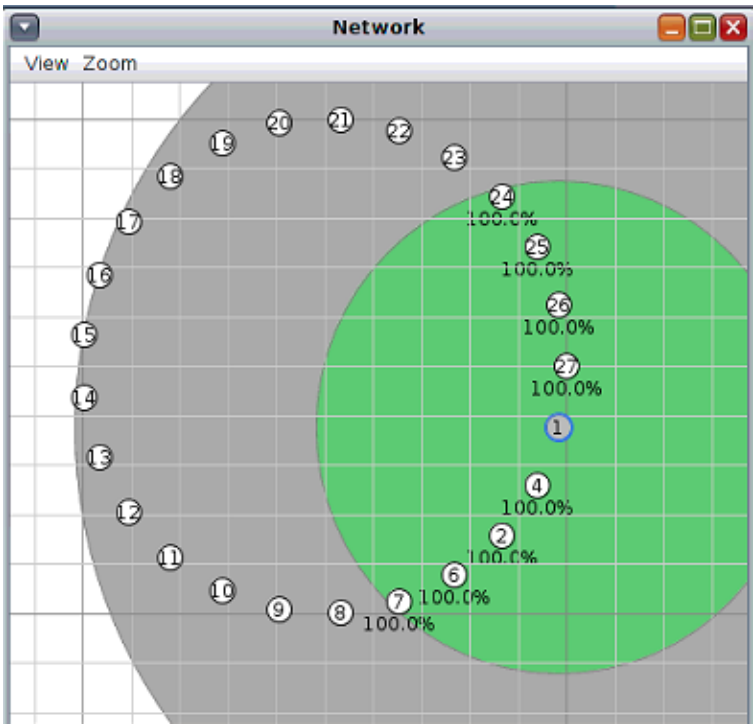

Figure 4.5. Ring topology setup.

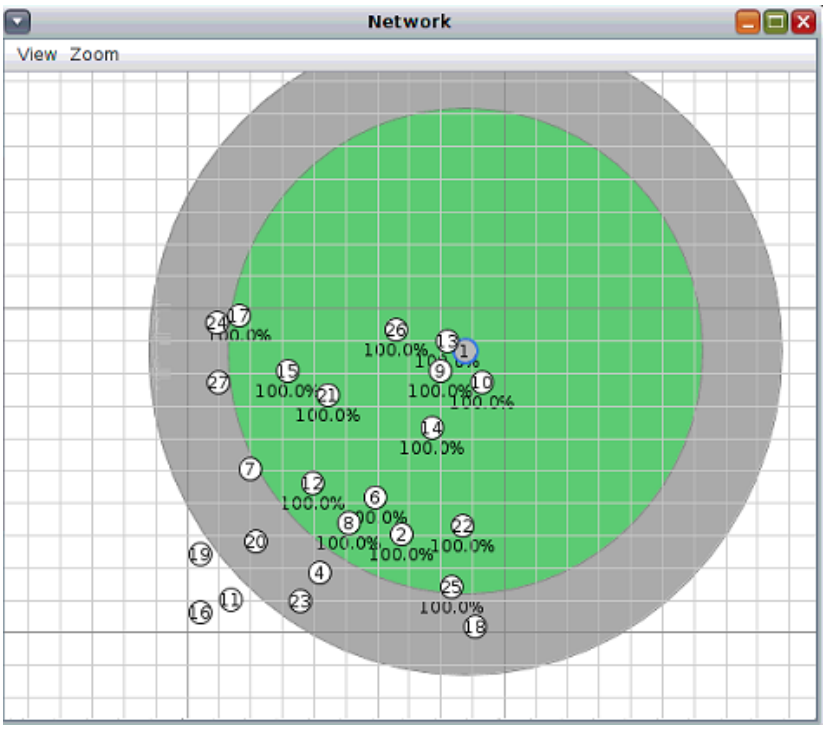

Figure 4.6. Mesh topology setup.

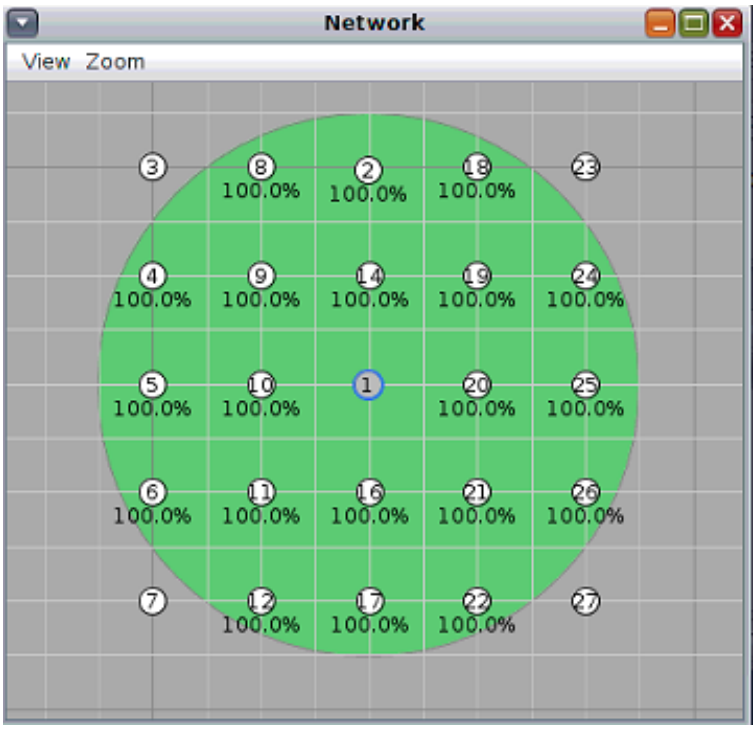

Figure 4.7. Grid topology setup.

To consider and compare the energy consumption of individual SDN-enabled nodes in the given time period, a sample space of 24 nodes for each of the topology configurations including the sink (Node ID 2) is taken. The resulting energy performance graphs are shown in Figures 4.11, 4.12 and 4.13.

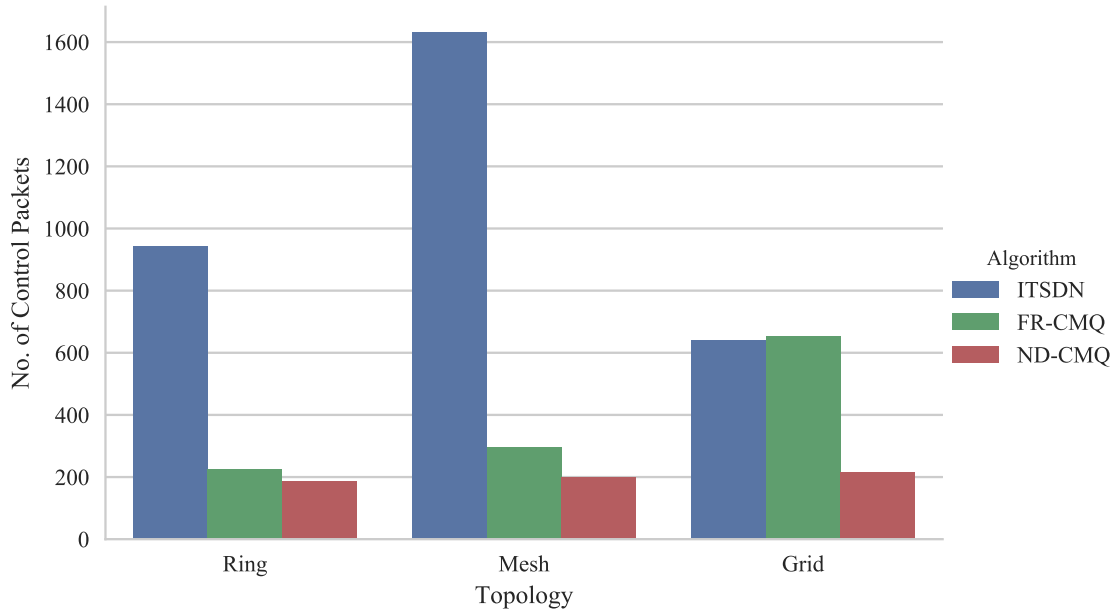

Figure 4.8. Total number of control packets.

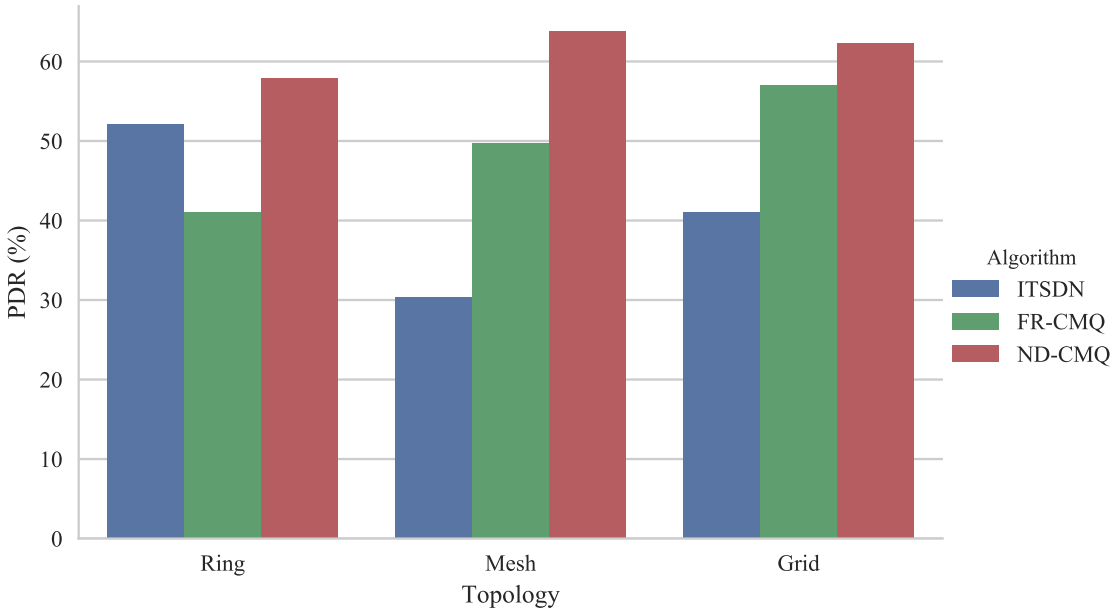

Figure 4.9. Packet delivery rate.

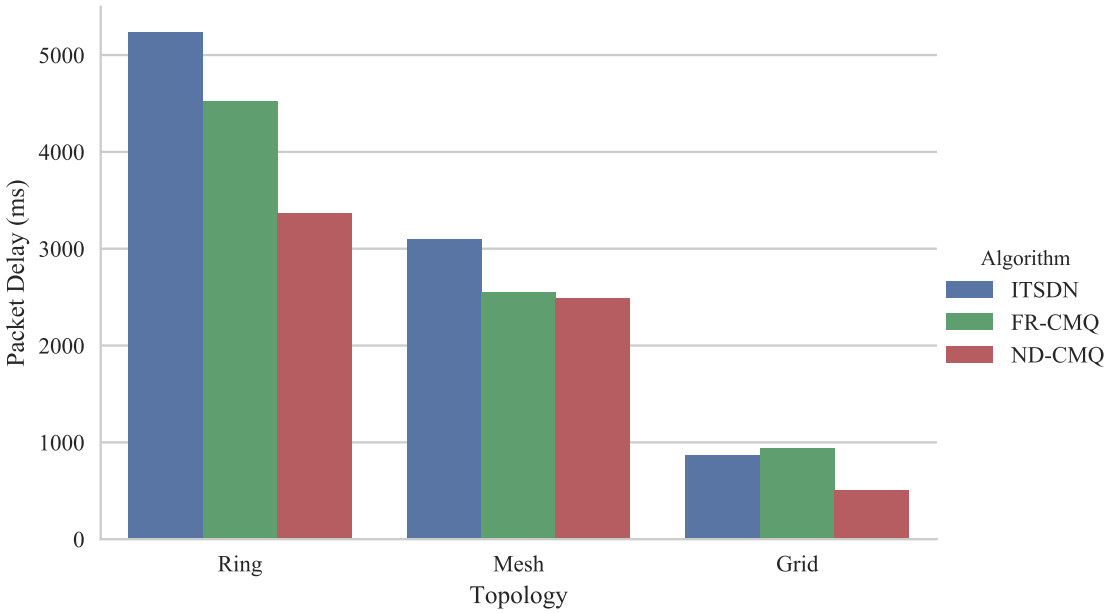

Figure 4.10. Data packet delay.

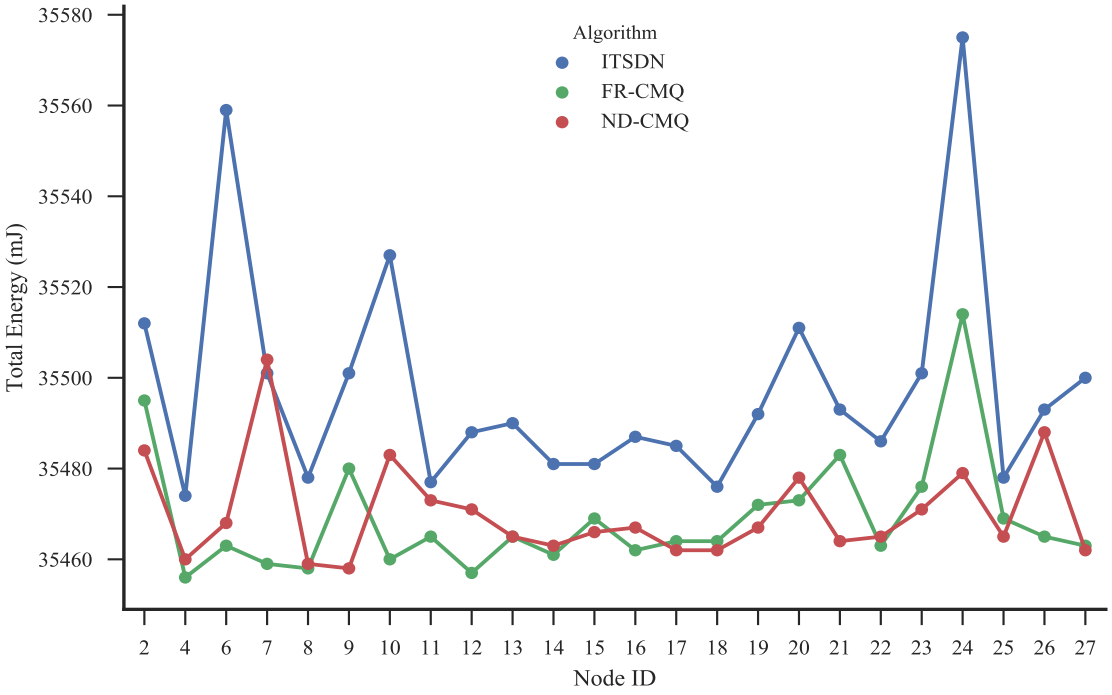

Figure 4.11. Energy consumption in ring topology.

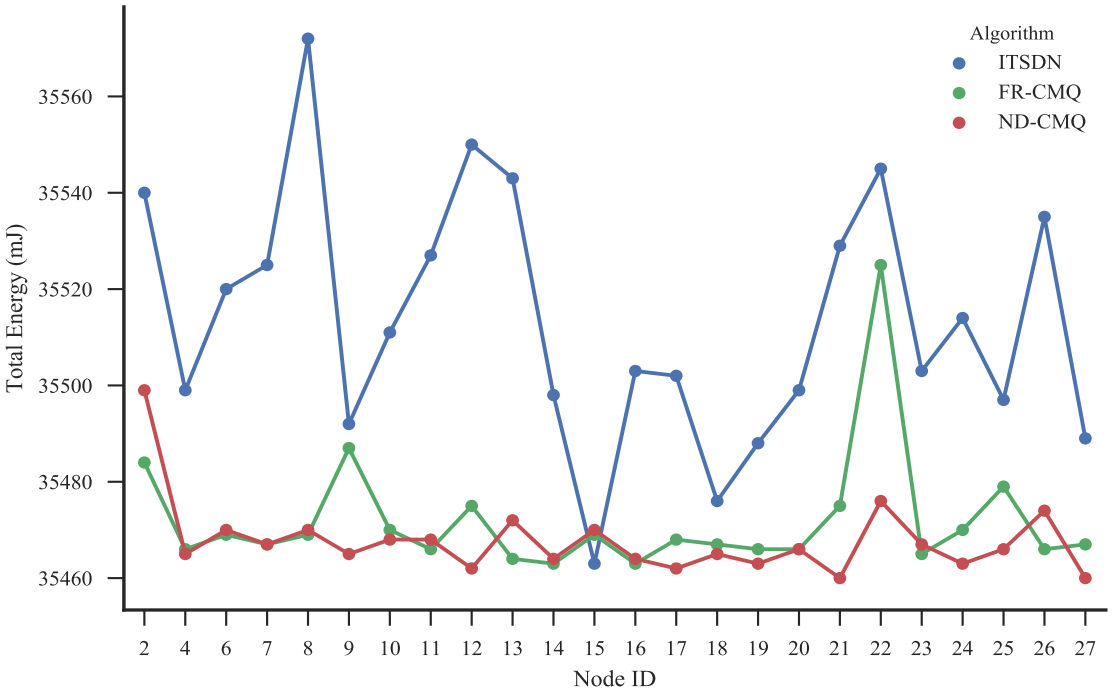

Figure 4.12. Energy consumption in mesh topology.

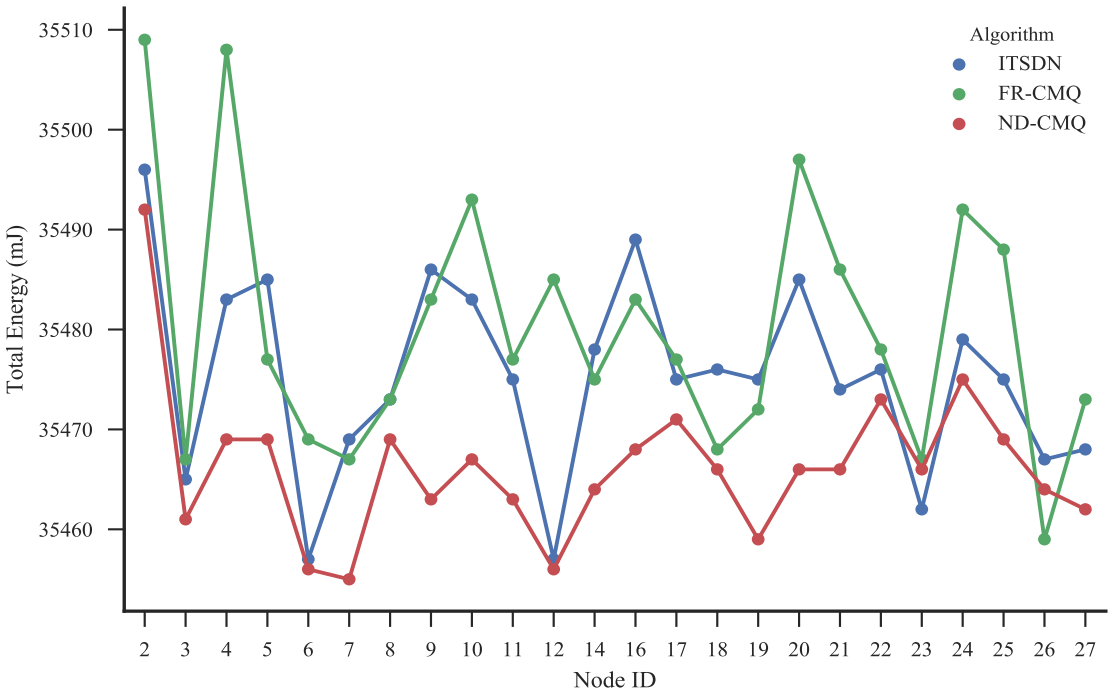

Figure 4.13. Energy consumption in grid topology.

4.5.2 Evaluation for network scalability

From prior experiments of this setup, the ring topology promised better scalability performance than the grid or mesh topologies, therefore in the second set of simulations this topology is used to evaluate the performance of the control message quenching algorithms as the network increases in size. The results obtained are shown in Figures 4.14, 4.15 and 4.16 for the number of control packets generated, PDR and delay respectively.

To further evaluate the control message build up, a 30-minute simulation for the 25 node setup was simulated and the number of control messages taken note of periodically. The resulting graph is shown in Figure 4.17.

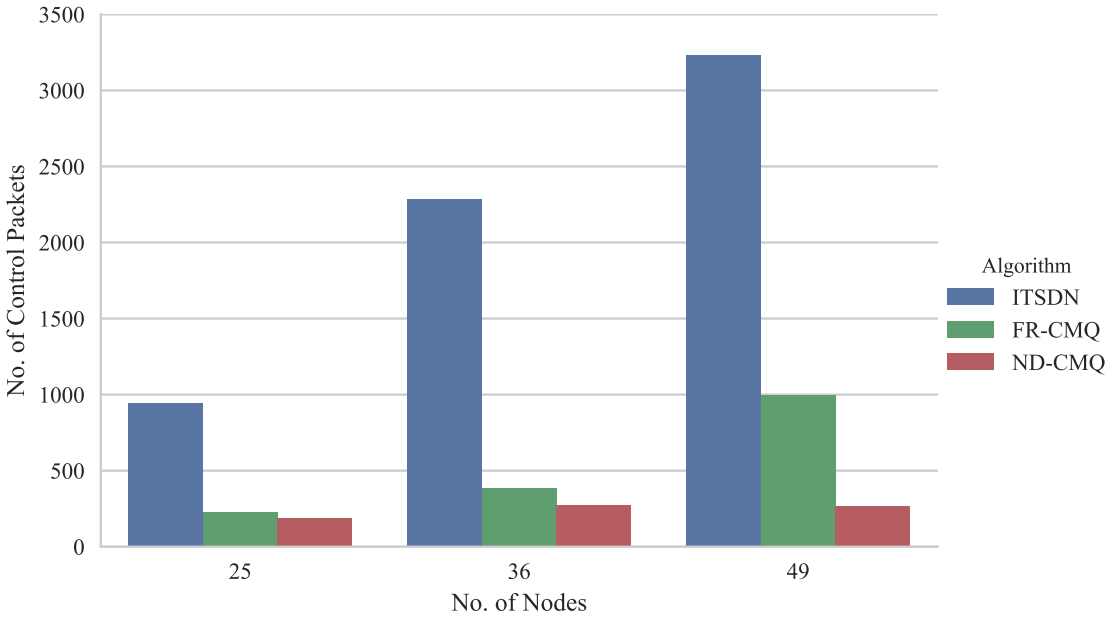

Figure 4.14. Control messages generated.

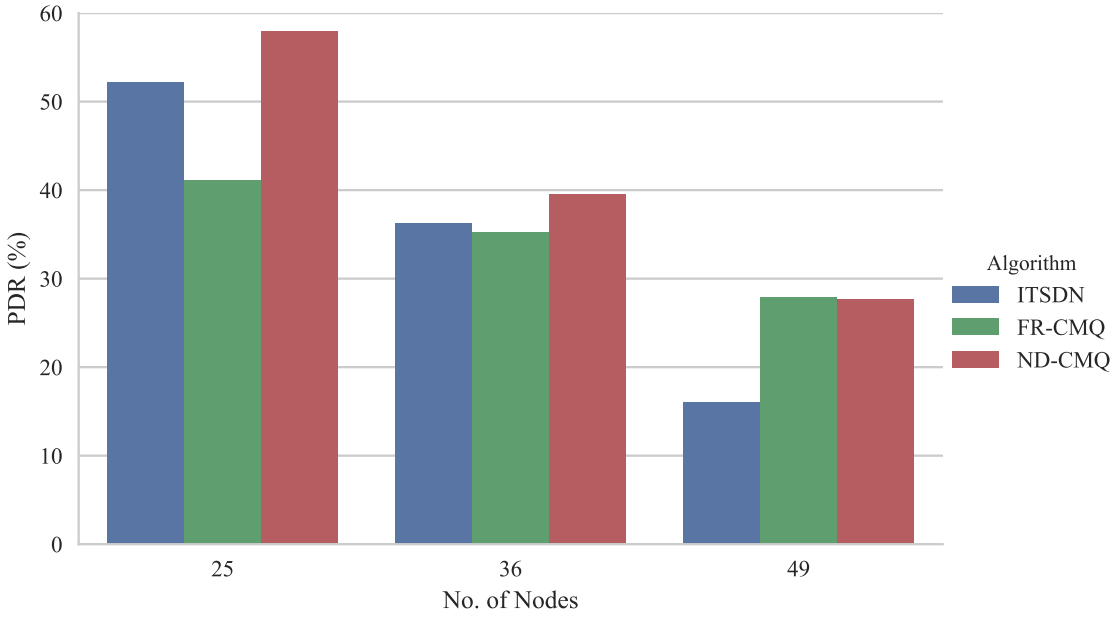

Figure 4.15. Data packet delivery.

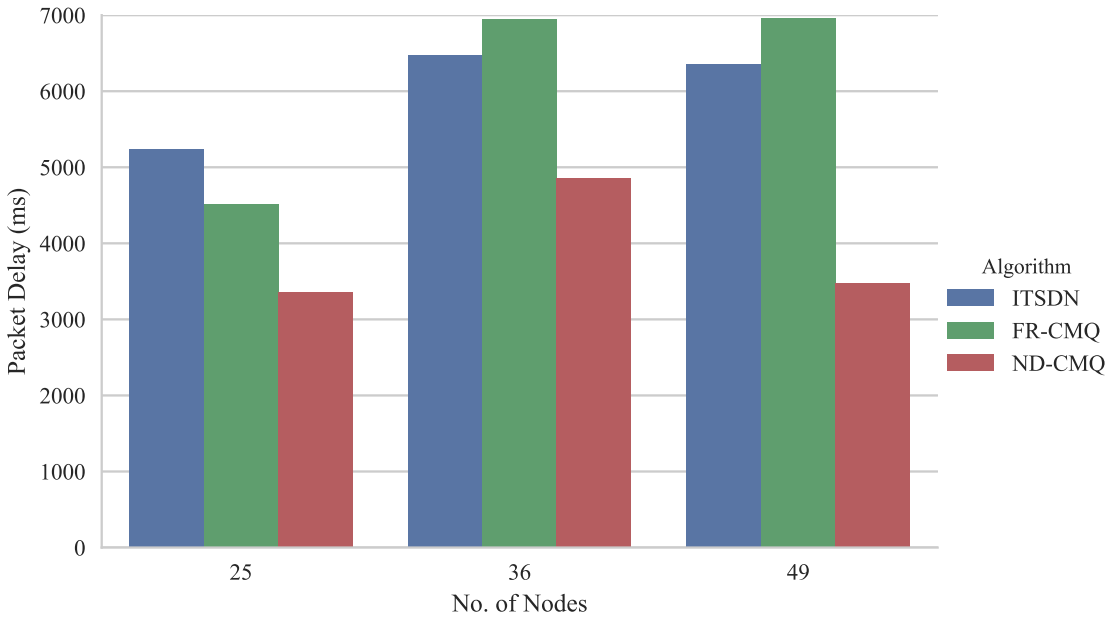

Figure 4.16. Data packet delay.

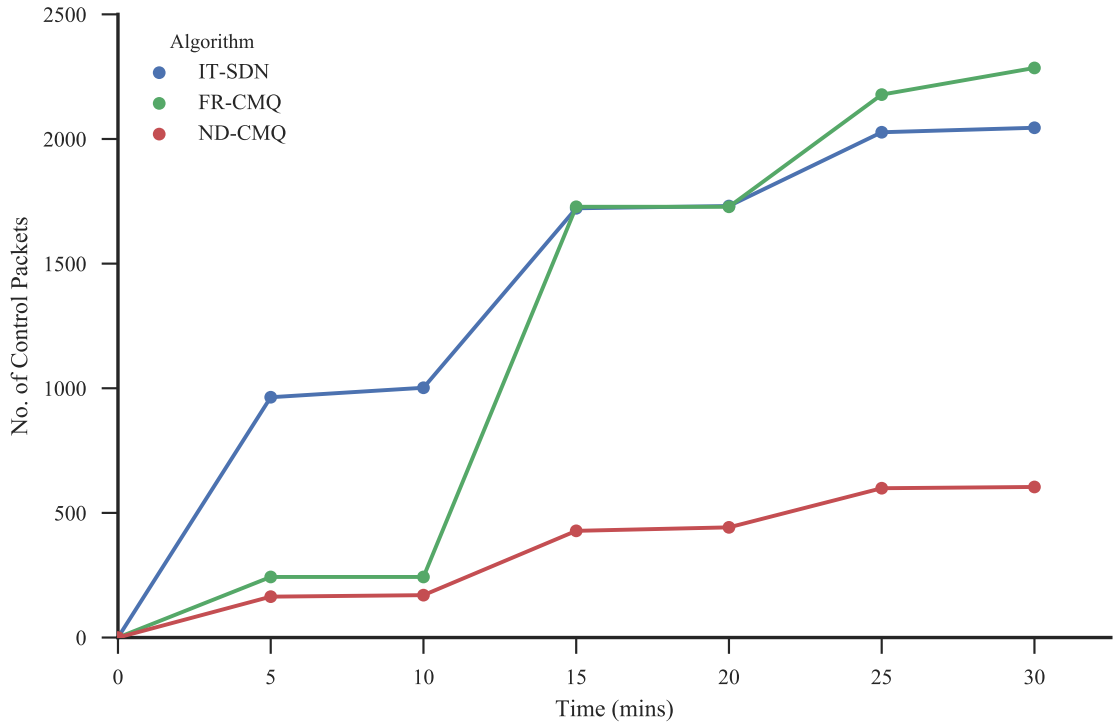

Figure 4.17. 25 node control message build-up.

4.6 DISCUSSION

The results thus far show the efficiency of neighbour report message quenching in reducing the overall control traffic messages. In varying topologies implementing a neighbour report quenching algorithm such as the proposed ND-CMQ shows a significant reduction in control messages. In Figure 4.8 we observe that compared to when no CMQ algorithm is implemented, there is a reduction in control messages of about 80% in the ring topology, 87% in the mesh and 66% in the grid topology with the ND-CMQ algorithm. The main reason for the improvement here is due to Algorithm 2 removing neighbour reports that do not meet the conditions of registering a neighbour change event and Algorithm 3 removing duplicate report entries. Even with an alternative CMQ algorithm such as that based on flow setup requests (FR-CMQ), the ND-CMQ still showed a further reduction in control messages of approximately 17% in the ring topology, 33% in the mesh topology and 67% in the grid topology. This improvement in control message reduction has been further highlighted in the cumulative graph in Figure 4.17 over a 30 minute period. We observe that the CMQ benefit is greater with an algorithm

based on neighbour reports than one based on flow setup requests because in the SDN-based WSN setup the magnitude of duplicate neighbour reports or beacon messages occupying the in-band traffic channel is much greater than that of duplicate flow setup request packets.

When considering control message reduction as the network scales in a ring topology, for example, we still see a significant reduction in control messages at each network size with ND-CMQ. In such a scenario, Figure 4.14 shows that compared to when no CMQ algorithm is implemented the ND-CMQ algorithm shows a control message reduction of approximate 80%, 88% and 91% at 25, 36 and 49 nodes respectively. Compared to FR-CMQ for increasing network sizes of 25, 36 and 49 nodes respectively, ND-CMQ introduces a control message reduction of 17%, 28%, and 73%. It can be concluded that with larger network sizes there is an increased amount of duplicate messages from which the network leverages from the duplicate entry removal algorithm at a higher rate.

The direct benefit of fewer control messages being generated by sensor nodes is improved energy efficiency. Most of the energy is lost during transmission and reception of packets and therefore less control packets and reduced duplicate neighbour information results in a lower traffic payload and consequently reduced energy consumption. Figures 4.11, 4.12 and 4.13 represent the resulting impact of the CMQ algorithms on the overall node energy consumption. In the grid topology result shown in Figure 4.13, the proposed ND-CMQ algorithm showed the greatest energy efficiency with 21 out of the 24 nodes consuming the least amount of energy. This followed by the mesh topology in Figure 4.12 with 14 out of 24 nodes consuming the lowest energy. The results in Figure 4.11 showed the least energy efficiency for ND-CMQ which can be aligned to localization needs of the ring setup considering the associated traffic congestion of the topology.

Another benefit of reduction in control messages is increased controller responsiveness which leads to improvements in packet delivery rate (PDR) and packet delay in all three topology configurations. Significant improvements in this PDR included: a 33% improvement in packet delivery with ND-CMQ compared to when no CMQ algorithm is implemented for the mesh topology and a 17% improvement in packet delivery over the FR-CMQ algorithm for the ring topology, this observed from the graph in Figure 4.9. Further improvements in packet delivery and packet delay have also been shown even as the network size is increased. However, in Figure 4.15 the improvement gap in PDR as the network scaled upwards reduced due to trade-off characteristics.

4.7 CHAPTER SUMMARY

This chapter introduced and discussed various aspects of control message quenching (CMQ) in SDN-based WSNs. In Section 4.2 a background to CMQ for SDN-based WSNs was given. The problem of overhead control traffic was addressed with a further focus on available CMQ techniques. Section 4.3 introduced the aspect of control message quenching based on removing duplicate flow setup requests (FR-CMQ). The associated algorithm was presented and the simplicity of implementation highlighted. In Section 4.4 a neighbour discovery control message quenching (ND-CMQ) algorithm to effectively reduce the number of control messages in an SDN-based WSN was presented. The algorithm focused on reducing the number of neighbour reports generated and thereby aiding the effective use of the in-band traffic channel. This would effectively benefit the modular management scheme established in this study, as limiting the adjacent control traffic would free up bandwidth for context packets required for management purposes. Section 4.5 provided details on the simulation setup and performance evaluation of the ND-CMQ algorithm including the presentation of simulation results. In Section 4.6 the performance results of the neighbour discovery CMQ algorithm were discussed in detail. Implementation of the proposed algorithm showed improvements in packet delivery rate, delay, and energy efficiency over the baseline ITSDN [124] framework it was implemented on and also over the FR-CMQ algorithm.

CHAPTER 5 CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

The Internet of Things (IoT) has brought with it several applications in industry and society at large. We are seeing more implementation on a much larger scale for smart city development and also in the medical industry for targeted health needs. The direct impact of this is the deployment of Wireless Sensor Networks (WSN) to the scale of hundreds of thousands of sensor nodes in varying environments which include harsh and hard to reach areas. Most of these sensor nodes are provided by different vendors and hence results in a heterogeneous setup that creates a management challenge especially with remote node reprogramming and general vertical application integration. This management complexity is further magnified with large scale WSN implementations and by the resource-constrained nature of WSNs. Software-Defined Networking (SDN), a paradigm based on providing global control of networks by separating the network intelligence from the data infrastructure (switches, nodes) promises the efficiency of flexible management in solving the above said WSN management problem. The resulting resource-capable and logically centralized controller maintains a global view of the entire network allowing ease of policy implementation and network reprogramming regardless of network heterogeneity. While we agree that SDN introduces greater management flexibility to wireless sensor networks, it has been observed from current literature that there exists very few to no generic frameworks that define taking advantage of this flexibility or taking into consideration the possibility of management modularity.

The first major objective of this research study was to do a thorough literature review on network management of both traditional WSN and SDN-based WSN. Part of the literature study showed that

traditional wireless sensor network technology has had proposals for generic models of managing several network aspects such as topology, routing, energy, security, network devices and associated software. These models being generic have enabled utility in multiple vendor applications with the aid of policies towards promoting vertical integration of management techniques. It is generally accepted that having such frameworks in place provides a point of reference for network administrators to easily and quickly implement management policies with sufficient cross-vendor compatibility. Integration of software-defined networking provides a global view of the network at large which makes it easy to implement cross-vendor management policies. The available Sensor OpenFlow protocol has provided a guide to SDN-based implementation frameworks for WSNs. Accordingly, such frameworks have been developed most commonly SDN-WISE and IT-SDN however, such frameworks enable wireless sensor network developers to implement and enable SDN-based control at the protocol level. Therefore, in this study, it was found that there was a greater need for SDN-based frameworks for wireless sensor networks that focus mainly on the management policy aspect to aid rapid prototyping and implementation of management policies by administrators.

In line with the aforementioned need, this study proposed a novel SDN-based modular management framework that envisioned a generic platform suitable for use across several SDN-based vendor applications with the aspect of adding or removing management policies as components or modules. This resulted in the second major objective of this study; developing a mechanism or interface to enable the aspect of management modularity. A secondary objective to this involved implementing the proposed system on a generic and widely available SDN-based WSN framework, IT-SDN in this case. To achieve the primary objective, two aspects of network management were introduced namely a Management Service Interface (MSI) and a context knowledge base. The MSI played a crucial role in the development and implementation of an SDN-based modular management system. It provided a common interface based on context data from the data plane to management modules that existed as states. Therefore, the MSI was state-machine based to allow management modules or components to exist as states interacting with a common context knowledge base.

The secondary objective of this was to implement the MSI and context knowledge database on the IT-SDN platform. This objective was achieved by building the MSI as part of the controller node firmware and using contiki lists to develop dynamic memory allocation for context information in the controller. Populating the context knowledge base involved generation of extra management control packets which were referred to as context packets by the SDN-enabled nodes in the data plane.

Information such as node battery level, number of neighbours, RSSI and number of active tasks in the node was collected and packaged in a context packet destined for the context knowledge base in the controller. To evaluate the feasibility of the modular management system, a sample resource allocation module was built and included as a state in the MSI. Functions of the module involved basic decision making on the actions each node should take based on having at least a threshold level of the number of neighbours and battery level. Dynamic sampling of the task generation was also calculated and the sampling interval determined based on each node battery level. The main idea of this implementation was to demonstrate the potential of predetermined task decisions being made by a module in the MSI based on the context knowledge base. This potential can further be harnessed to include information such as the encryption state of data in the data plane or other network vulnerability information for a security management module to make useful global decisions.

Findings from the implementation of the sample resource allocation module showed improvements in packet delivery rate, delay, energy efficiency and general control overhead over comparable frameworks in this case baseline IT-SDN and SDN-WISE for increasing node sizes in a grid configuration. Further simulations were done based on varying network topology configurations such as grid, ring and mesh where improvements in overall performance were exhibited by the proposed modular system. However, it is crucial to the research study to conclude that in this particular case where the improvements had been introduced based on control of the task sampling intervals which inherently meant lower frequency of task generation leading to lesser traffic congestion and thus improving PDR, delay and energy efficiency and ultimately lowering overhead traffic; there may be other application scenarios that are not resource allocation based that may result in increased control traffic due to the extra context data packets. This study considered the fact that the MSI is dependent on an up-to-date context knowledge base that may lead to increased overhead control traffic which would be a potential bottleneck to the performance of the SDN-based WSN. It took into account that the architecture of the target network was dependent on a limited in-band traffic channel for both control and data traffic.

This potential bottleneck called for further research into solutions to minimize overall control traffic in SDN-based WSNs. Thus, the third major objective of this study was to open a conversation on control message quenching techniques for SDN-based WSNs, implement and evaluate message quenching of neighbour discovery messages specifically for SDN-based 6LoWPANs. The focus in relation to the overall research study was to propose a management technique that can minimize adjacent control traffic in the in-band traffic channel thus providing more bandwidth allocation for management packets.

Management packets such as the introduced context traffic in the proposed modular management system would potentially benefit from such a mechanism. The first step to achieving this objective was to provide and discuss the literature background on control message quenching both for computer network based SDN and SDWSN. Findings showed that the simplest and least resource-demanding methods of reducing control traffic in SDN-based WSN involved integrating an algorithm to prevent duplication and redundancy of control messages such as duplicate flow setup requests or neighbour discovery messages. A novel algorithm was proposed with the goal of increasing the efficiency of neighbour report generation which would end up significantly reducing the overall number of neighbour reports.

The proposed Neighbour Discovery Control Message Quenching (ND-CMQ) algorithm was then evaluated against the performance of a message quenching algorithm based on flow setup requests (FR-CMQ) and also against the performance of baseline IT-SDN. Both the ND-CMQ and FR-CMQ algorithms were implemented on the IT-SDN platform and evaluated for PDR, delay, energy efficiency and resulting control traffic. The performance parameters were evaluated for varying network sizes and varying topology setups (grid, mesh and ring). Results showed promising and improved performance for the ND-CMQ algorithm particularly for the level of control traffic reduction both for varying topology and increasing network sizes of 25, 36 and 49 nodes arranged in a ring topology. Notable improvements over FR-CMQ and baseline IT-SDN were made for packet delivery rate, delay and energy efficiency for 25 nodes in the ring, mesh and grid topologies.

As the network size was increased, performance in terms of packet delay was better with the ND-CMQ algorithm however in terms of packet delivery rate the performance exhibited trade-off characteristics. The level of PDR improvement kept reducing as the network scale increased with FR-CMQ showing a slightly better performance over ND-CMQ at 49 nodes. This trend in performance was due to the nature of control message quenching based on neighbour discovery or beacon messages. That is the major challenge associated with this technique, as there exists a trade-off between the magnitude of neighbour reports/beacon messages quenched and the network performance especially the packet delivery. The findings in this research generally conclude that a system is necessary to be put in place to optimize the trade-off that exists between how much of the neighbour reports are to be reduced and the design requirements of the network performance (QoS). Potential solutions to this engineering problem have been included as part of future work study.

5.1.1 Summary of contributions

Throughout this research study, several contributions to scientific research were made and can be summarised as follows:

1. A comprehensive literature study surveying approaches and techniques of SDN-based management. The outcome of this contribution resulted in the publication of a survey paper detailing contributions and crucial aspects in the design of SDN specifically for improvement in wireless sensor network management. The study carried out also provided a tutorial basis for researchers interested in leveraging SDN for flexible management of WSNs. SDN integration in the management of essential components such as topology, energy, security, QoS, network device and software were thoroughly discussed and compared. Based on the findings, the study also poised to showcase crucial research gaps in the field of SDN-based management of WSNs. It was found particularly that while several SDN implementation techniques for WSN were proposed in the literature, there was still a greater need for a generic framework/architecture to uniformly guide network administrators intending to harness the potential of SDN-based management. From the current statistics on research platforms, the resultant paper is increasingly being read, cited and recommended for researchers interested specifically in the management aspects and the benefits that SDN introduces to wireless sensor networking.
2. A design and implementation of a generic and modular management system for SDN-based WSNs. The core work of this research study was covered by this contribution. The modular management framework provided the research community with a general solution to integrating SDN for management purposes in WSN. Potential benefits included rapid implementation of management policies though management modularity and ease of access to the framework by implementation on the widely available IT-SDN tool. Management modularity saw the benefit of adding management policies as modules or components and this was demonstrated using a sample resource allocation tool. Performance evaluation showed simulation results giving improved packet delivery, delay, energy efficiency and control overhead for the proposed system compared to baseline IT-SDN and SDN-WISE. Considering that the modularity concept was based on the generation of extra management packets, it called for an investigation into generally reducing overhead control traffic in SDN-based WSN setups. The findings from the proposed system were accepted for publication in the IEEE Systems Journal.

3. A discussion on Control Message Quenching (CMQ) for wireless sensor networks based on SDN was opened and introduced to the SDN-based WSN research community. A state of the art review was given on CMQ approaches available for SDWSN while making references to adoptions made in the computer network based SDN. As part of the ambition to tailor SDN control message quenching techniques for WSN, challenges and design requirements were also outlined. Future research prospects in this field of study were also identified with potential research directions on reducing control traffic overhead. This aspect of the research study was compiled and accepted as part of the proceedings for the 2019 SATNAC conference.
4. In continuity on the discussion of control message quenching for SDN-based WSNs, a demonstration and discussion of a CMQ algorithm specifically for the reduction of flow setup request packets (FR-CMQ) in SDN-based 6LoWPANs was made. Implementation and simulation of the algorithm showed improvements over the baseline IT-SDN platform it was implemented on. The improvements included parameters such as overall control messages generated and energy efficiency. However, the study also found inconsistencies in the performance relating to packet delay and packet delivery as the network size varied. Therefore, potential solutions to improving the CMQ technique performance were provided under future work. This portion of the research study resulted in a paper accepted as part of the proceedings for the 2019 INDIN conference, a flagship conference for industrial based sensor network research.
5. A design and implementation for a control message quenching algorithm to effectively limit and control the generation of neighbour reports in SDN-based 6LoWPAN environments. A two-staged algorithm (ND-CMQ) was developed to effectively reduce the number of neighbour discovery messages going to the controller. The first stage filtered conditions necessary to trigger a neighbour event ensuring non-redundancy in neighbour information with the second stage playing the role of preventing information duplication. The algorithm was implemented in the IT-SDN platform and evaluated against baseline IT-SDN and the FR-CMQ algorithm. The algorithm showed improvements across varying topologies and different network sizes, this especially with the number of control packets generated and energy efficiency. However, the results in terms of packet delivery and delay exhibited some characteristics of trade-off performance. Conclusions from the findings suggested potential solutions for minimizing this trade-off and optimizing performance. These suggestions have been placed as part of future research work contributions. The contributions made by this portion of the study were compiled and submitted for review and publication in the IEEE Wireless Communication Letters.

5.2 FUTURE WORK

Future work will involve the implementation of the modular management system on distributed controllers together with the use of information fusion techniques [106] specifically flow-rule aggregation [140] in a bid to further reduce the resulting control overhead. Further energy efficiency can be achieved by adding a dedicated energy management module with power management functions and an energy efficient routing mechanism in the modular system to compensate for energy losses due to generating context data. To further investigate the efficiency of such a modular management system, it is vital that resulting running complexity against varying density of sensor nodes be analysed. Additionally, the aspect of load balancing the whole network should be considered to improve energy consumption and extend network lifetime. It is also expected that machine learning methods can be used in the collection of context data more intelligently and efficiently [141] based on management module requirement. Context data could take any form beyond that collected in the demonstration carried out in this study, for example, a security management module could request the encryption status of data in the data plane as context and make decisions based on it. It is worthwhile to mention here that with management modularity in place, concepts based on adjusting transmission power [127] and transmission ranges to improve network lifetime can be built as flexible modules based on the context data. As other SDWSN development platforms become more available, the SDNMM system framework will be implemented on them and the resulting effects evaluated, for instance, it would be interesting to investigate modular management in SDN-WISE [12]. In general, the generic SDN-based modular management system developed and implemented in this research study has introduced a niche in IoT management requiring the development of various modules for security, topology and QoS management to mention a few. These various modules can all play a role in improving the efficiency of network performance and reliability.

With regards to the NDCMQ algorithm, there is a need to investigate and optimize the proactive probing interval of neighbour reports. This can be done in a bid to further improve packet delivery and reduce delay while keeping control messages at a minimum. Traditional multi-objective optimization can be used to meet trade-off requirements of PDR, delay and energy efficiency. Mathematical optimization techniques can be looked into to satisfy this requirement. Machine learning and other cognitive efforts can also be investigated towards achieving reduced control traffic overhead and improved controller responsiveness in general. Such intelligent methods can be used to optimize the proactive probing interval for neighbour discovery while keeping the PDR and delay trade-offs to a minimum.

REFERENCES

[1] M. J. Mudumbe and A. M. Abu-Mahfouz, "Smart water meter system for user-centric consumption measurement," in *Proceedings of the IEEE 13th International Conference on Industrial Informatics (INDIN)*, July 2015, pp. 993–998.

[2] A. M. Abu-Mahfouz, Y. Hamam, P. R. Page, K. Djouani, and A. Kurien, "Real-time dynamic hydraulic model for potable water loss reduction," *Procedia Engineering*, vol. 154, no. 1, pp. 99–106, 2016.

[3] B. N. Silva, M. Khan, C. Jung, J. Seo, D. Muhammad, J. Han, Y. Yoon, and K. Han, "Urban planning and smart city decision management empowered by real-time data processing using big data analytics," *Sensors*, vol. 18, no. 9, pp. 1–19, 2018.

[4] K. Xu, X. Wang, W. Wei, H. Song, and B. Mao, "Toward software-defined smart home," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 116–122, 2016.

[5] S. Oueida, Y. Kotb, M. Aloqaily, Y. Jararweh, and T. Baker, "An edge computing based smart healthcare framework for resource management," *Sensors*, vol. 18, no. 12, pp. 1–22, 2018.

[6] G. Tolle and D. Culler, "Design of an application-cooperative management system for wireless sensor networks," in *Proceeedings of the 2nd European Workshop on Wireless Sensor Networks*, February 2005, pp. 121–132.

[7] H. Song, "UPnP-based sensor network management architecture," in *Proceedings of the Conference on Mobile Computing and Ubiquitous Networking*, April 2005, pp. 1–6.

[8] L. B. Ruiz, J. M. Nogueira, and A. A. Loureiro, "MANNA: A management architecture for wireless sensor networks," *IEEE Communications Magazine*, vol. 41, no. 2, pp. 116–125, 2003.

[9] C. Orfanidis, "Ph.D. forum abstract: Increasing robustness in wsn using software-defined network architecture," in *Proceedings of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, April 2016, pp. 1–2.

[10] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, "A survey on software-defined wireless sensor networks: Challenges and design requirements," *IEEE Access*, vol. 5, no. 2, pp. 1872–1899, 2017.

[11] B. T. De Oliveira, L. B. Gabriel, and C. B. Margi, "TinySDN: Enabling multiple controllers for software-defined wireless sensor networks," *IEEE Latin America Transactions*, vol. 13, no. 11, pp. 3690–3696, 2015.

[12] 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 *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, April 2015, pp. 513–521.

[13] R. Alves, D. Oliveira, G. Nez, and C. B. Margi, "IT-SDN: Improved architecture for SDWSN," in *Proceedings of the 35th Brazilian Symposium on Computer Networks and Distributed Systems*, May 2017, pp. 15–19.

[14] T. M. Cao, B. Bellata, and M. Oliver, "Design of a generic management system for wireless sensor networks," *Ad Hoc Networks*, vol. 20, no. 12, pp. 16–35, 2014.

[15] P. Di Dio, S. Faraci, L. Galluccio, S. Milardo, G. Morabito, S. Palazzo, and P. Livreri, "Exploiting state information to support QoS in software-defined WSNs," in *Proceedings of the Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, June 2016, pp. 1–7.

[16] M. Ndiaye, G. P. Hancke, and A. M. Abu-Mahfouz, "Software-defined networking for improved wireless sensor network management: A survey," *Sensors*, vol. 17, no. 5, pp. 1–32, 2017.

[17] B. Zhang and G. Li, "Survey of network management protocols in wireless sensor network," in *Proceedings of the International Conference on E-Business and Information System Security*, May 2009, pp. 1–5.

[18] W. L. Lee, A. Datta, and R. Cardell-Oliver, "Network management in wireless sensor networks," in *Handbook of Mobile Ad Hoc and Pervasive Communications*, L. T. Yang, X. Liu, and M. K. Denko, Eds. Boca Raton, FL, USA: CRC Press, Inc., 2014, pp. 1–20.

[19] H. Karl and A. Willig, *Protocols and architectures for wireless sensor networks*, 1st ed. John Wiley & Sons, 2007.

[20] Y. W. Ma, J. L. Chen, Y. M. Huang, and M. Y. Lee, "An efficient management system for wireless sensor networks," *Sensors*, vol. 10, no. 12, pp. 11 400–11 413, 2010.

[21] Y. Zhao, M. N. Anjum, M. Song, X. Xu, and G. Wang, "Optimal resource allocation for delay constrained users in self-coexistence WRAN," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, December 2015, pp. 1–6.

[22] J. Louw, G. Niezen, T. D. Ramotsoela, and A. M. Abu-Mahfouz, "A key distribution scheme using elliptic curve cryptography in wireless sensor networks," in *Proceedings of the IEEE International Conference on Industrial Informatics (INDIN)*, July 2016, pp. 1166–1170.

[23] A. S. Reegan and E. Baburaj, "Key management schemes in wireless sensor networks: A survey," in *Proceedings of the International Conference on Circuits, Power and Computing Technologies (ICCPCT)*, March 2013, pp. 813–820.

[24] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[25] N. Ntuli and A. Abu-Mahfouz, "A simple security architecture for smart water management system," *Procedia Computer Science*, vol. 83, no. 7, pp. 1164–1169, 2016.

[26] Z. Yu and Y. Guan, "A key management scheme using deployment knowledge for wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 10, pp. 1411–1425, 2008.

[27] Y. Zhang, J. Liang, B. Zheng, S. Jiang, and W. Chen, "Key management scheme based on route planning of mobile sink in wireless sensor networks," *Sensors*, vol. 16, no. 2, pp. 1–18, 2016.

[28] Y. Zhang, J. Liang, B. Zheng, and W. Chen, "A hybrid key management scheme for WSNs based on PPBR and a tree-based path key establishment method," *Sensors*, vol. 16, no. 4, pp. 1–18, 2016.

[29] P. Dayal and P. Kumar, "Fault mitigation by topology management in WSN: A survey," in *Proceedings of the International Conference on Communications and Signal Processing (ICCSP)*, April 2015, pp. 836–840.

[30] M. Turon, "MOTE-VIEW: A sensor network monitoring and management tool," in *Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors (EmNetS-II)*, May 2005, pp. 11–17.

[31] Y. Yang, P. Xia, L. Huang, Q. Zhou, Y. Xu, and X. Li, "SNAMP: A multi-sniffer and multi-view visualization platform for wireless sensor networks," in *Proceedings of the 1st IEEE Conference on Industrial Electronics and Applications*, May 2006, pp. 1–4.

[32] C. Buschmann, D. Pfisterer, S. Fischer, S. P. Fekete, and A. Kröller, "Spyglass: A wireless sensor network visualizer," *ACM Sigbed Review*, vol. 2, no. 1, pp. 1–6, 2005.

[33] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, November 2003, pp. 126–137.

[34] M. Yu, J. Song, J. Kim, K. Y. Shin, and P. S. Mah, "NanoMon: A flexible sensor network monitoring software," in *Proceedings of the 9th International Conference on Advanced Communication Technology*, February 2007, pp. 1423–1426.

[35] F. P. Garcia, R. M. C. Andrade, C. T. Oliveira, and J. N. De Souza, "EPMOSt: An energy-efficient passive monitoring system for wireless sensor networks," *Sensors*, vol. 14, no. 6, pp. 10 804–10 828, 2014.

[36] B. Srbinovski, M. Magno, F. Edwards-Murphy, V. Pakrashi, and E. Popovici, "An energy aware adaptive sampling algorithm for energy harvesting WSN with energy hungry sensors," *Sensors*, vol. 16, no. 4, pp. 1–19, 2016.

[37] T. Luo, H.-P. Tan, and T. Q. Quek, "Sensor OpenFlow: Enabling software-defined wireless sensor networks," *IEEE Communications Letters*, vol. 16, no. 11, pp. 1896–1899, 2012.

[38] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and openflow: From concept to implementation," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2181–2206, 2014.

[39] X. N. Nguyen, D. Saucez, C. Barakat, and T. Turletti, "Rules placement problem in OpenFlow networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1273–1286, 2016.

[40] H. Kim and N. Feamster, "Improving network management with software-defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, 2013.

[41] R. Jain and S. Paul, "Network virtualization and software-defined networking for cloud computing: A survey," *IEEE Communications Magazine*, vol. 51, no. 11, pp. 24–31, 2013.

[42] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, "Software-defined wireless networks: Unbridling SDNs," in *Proceedings of the European Workshop on Software-Defined Networking*, October 2012, pp. 1–6.

[43] A. De Gante, M. Aslan, and A. Matrawy, "Smart wireless sensor network management based on software-defined networking," in *Proceedings of the 27th Queen's Biennial Symposium on Communications (QBSC)*, June 2014, pp. 71–75.

[44] H. Kim, T. Benson, A. Akella, and N. Feamster, "The evolution of network configuration: A tale of two campuses," in *Proceedings of the ACM SIGCOMM Conference on Internet Measurement*, November 2011, pp. 499–514.

[45] D. Zeng, T. Miyazaki, S. Guo, T. Tsukahara, J. Kitamichi, and T. Hayashi, "Evolution of software-defined sensor networks," in *Proceedings of the IEEE 9th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, December 2013, pp. 410–413.

[46] I. T. Haque and N. Abu-Ghazaleh, "Wireless software-defined networking: A survey and taxonomy," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2713–2737, 2016.

[47] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[48] H. Huang, J. Zhu, and L. Zhang, "An SDN based management framework for IoT devices," in *Proceedings of the IET Irish Signals Systems Conference and China-Ireland International Conference on Information and Communications Technologies (ISSC/ CIICT)*, June 2014, pp. 175–179.

[49] T. Miyazaki, D. Shitara, R. Kawano, Y. Endo, Y. Tanno, and H. Igari, "Robust wireless sensor network featuring automatic function alternation," in *Proceedings of the 20th International Conference on Computer Communications and Networks (ICCCN)*, July 2011, pp. 1–6.

[50] J. Li, X. Chang, Y. Ren, Z. Zhang, and G. Wang, "An effective path load balancing mechanism based on SDN," in *Proceedings of the IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*, September 2014, pp. 527–533.

[51] S. Bera, S. Misra, S. K. Roy, and M. S. Obaidat, "Soft-WSN: Software-defined WSN management system for IoT applications," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2074–2081, 2018.

[52] A. Darwish and A. E. Hassanien, "Wearable and implantable wireless sensor network solutions for healthcare monitoring," *Sensors*, vol. 12, no. 9, pp. 12 375–12 376, 2012.

[53] X. Chang, J. Li, G. Wang, Z. Zhang, L. Li, and Y. Niu, "Software-defined backpressure mechanism for edge router," in *Proceedings of the IEEE 23rd International Symposium on Quality of Service (IWQoS)*, June 2015, pp. 171–176.

[54] B. T. de Oliveira, C. B. Margi, and L. B. Gabriel, "TinySDN: Enabling multiple controllers for software-defined wireless sensor networks," in *Proceedings of the IEEE Latin-America Conference on Communications (LATINCOM)*, November 2014, pp. 1–6.

[55] P. Fraga-Lamas, T. M. FernÃąndez-CaramÃl's, M. SuÃąrez-Albela, L. Castedo, and M. GonzÃąlez-LÃşpez, "A review on internet of things for defense and public safety," *Sensors*, vol. 16, no. 10, pp. 1–44, 2016.

[56] N. Soetens, J. Famaey, M. Verstappen, and S. LatrÃl', "SDN-based management of heterogeneous home networks," in *Proceedings of the 11th International Conference on Network and Service Management (CNSM)*, November 2015, pp. 402–405.

[57] H. Kumar, H. H. Gharakheili, and V. Sivaraman, "User control of quality of experience in home networks using SDN," in *Proceedings of the IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, December 2013, pp. 1–6.

[58] Y. Yiakoumis, K.-K. Yap, S. Katti, G. Parulkar, and N. McKeown, "Slicing home networks," in *Proceedings of the ACM SIGCOMM Workshop on Home Networks*, August 2011, pp. 1–6.

[59] R. M. Abuteir, A. Fladenmuller, and O. Fourmaux, "SDN based architecture to improve video streaming in home networks," in *Proceedings of the IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, March 2016, pp. 220–226.

[60] P. Gallo, K. Kosek-Szott, S. Szott, and I. Tinnirello, "SDN@home: A method for controlling future wireless home networks," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 123–131, 2016.

[61] G. A. Akpakwu, G. P. Hancke, and A. M. Abu-Mahfouz, "Packets distribution in a tree-based topology wireless sensor networks," in *Proceedings of the IEEE 14th International Conference on Industrial Informatics (INDIN)*, July 2016, pp. 1181–1184.

[62] A. Dunkels, "Full TCP/IP for 8-bit architectures," in *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services*, May 2003, pp. 85–98.

[63] M. Durvy, J. Abeillé, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, and A. Dunkels, "Making sensor networks IPv6 ready," in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, November 2008, pp. 421–422.

[64] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - A lightweight and flexible operating system for tiny networked sensors," in *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, November 2004, pp. 455–462.

[65] C. Fuchs, "IP-based communication in wireless sensor network," in *Proceedings of the TUM Seminar on Sensor Nodes - Operation, Network and Application (SN)*, July 2011, pp. 67–73.

[66] P. Levis and D. Gay, *TinyOS programming*, 1st ed. Cambridge University Press, 2009.

[67] K. Feng, X. Huang, and Z. Su, "A network management architecture for 6LoWPAN network," in *Proceedings of the 4th IEEE International Conference on Broadband Network and Multimedia Technology*, October 2011, pp. 430–434.

[68] T. Luo, H. Tan, P. C. Quan, Y. W. Law, and J. Jin, "Enhancing responsiveness and scalability for OpenFlow networks via control-message quenching," in *Proceedings of the International Conference on ICT Convergence (ICTC)*, October 2012, pp. 348–353.

[69] J. Zhou, H. Jiang, J. Wu, L. Wu, C. Zhu, and W. Li, "SDN-based application framework for wireless sensor and actor networks," *IEEE Access*, vol. 4, no. 3, pp. 1583–1594, 2016.

[70] F. Olivier, G. Carlos, and N. Florent, "SDN based architecture for clustered WSN," in *Proceedings of the 9th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, July 2015, pp. 342–347.

[71] M. Ba, O. Flauzac, B. S. Haggar, F. Nolot, and I. Niang, "Self-stabilizing K-hops clustering algorithm for wireless ad-hoc networks," in *Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication*, January 2013, pp. 1–10.

[72] M. Capelle, S. Abdellatif, M. J. Huguet, and P. Berthou, "Online virtual links resource allocation in software-defined networks," in *Proceedings of the IFIP Networking Conference (IFIP Networking)*, May 2015, pp. 1–9.

[73] R. Rahmani, S. Rahman, and T. Kanter, "Context-based logical clustering of flow-sensors-exploiting HyperFlow and hierarchical DHTs," *RNIS: Research Notes in Information and Service Sciences*, vol. 14, no. 19, pp. 721–728, 2013.

[74] R. Rahmani, H. Rahman, and T. Kanter, "On performance of logical-clustering of flow-sensors," *International Journal of Computer Science Issues (IJCSI)*, vol. 10, no. 5, pp. 1–13, 2013.

[75] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in *Proceedings of the Internet Network Management Conference on Research on Enterprise Networking*, April 2010, pp. 1–6.

[76] B. R. Stojkoska and V. Kirandziska, "Improved MDS-based algorithm for nodes localization in wireless sensor networks," in *Proceedings of EUROCON Conference*, July 2013, pp. 608–613.

[77] F. Su, W. Ren, and H. Jin, "Localization algorithm based on difference estimation for wireless sensor networks," in *Proceedings of the International Conference on Communication Software and Networks*, February 2009, pp. 499–503.

[78] A. M. Abu-Mahfouz and G. P. Hancke, "An efficient distributed localisation algorithm for wireless sensor networks: based on smart reference–selection method," *International Journal of Sensor Networks*, vol. 13, no. 2, pp. 94–111, 2013.

[79] A. M. Abu-Mahfouz and G. P. Hancke, "Evaluating ALWadHA for providing secure localisation for wireless sensor networks," in *Proceedings of the AFRICON Conference*, September 2013, pp. 1–5.

[80] A. M. Abu-Mahfouz, G. P. Hancke, and S. J. Isaac, "Positioning system in wireless sensor networks using ns-2," *Software Engineering*, vol. 2, no. 4, pp. 91–100, 2012.

[81] A. M. Abu-Mahfouz and G. P. Hancke, "ALWadHA localization algorithm: Yet more energy efficient," *IEEE Access*, vol. 5, no. 3, pp. 6661–6667, 2017.

[82] Y. Zhu, Y. Zhang, W. Xia, and L. Shen, "A software-defined network based node selection algorithm in WSN localization," in *Proceedings of the IEEE 83rd Vehicular Technology Conference (VTC Spring)*, May 2016, pp. 1–5.

[83] S. S. Bhunia, S. K. Das, S. Roy, and N. Mukherjee, "Mobility management in IP based wireless sensor network using TinyOS," in *Proceedings of the 6th International Conference on Sensing Technology (ICST)*, December 2012, pp. 759–764.

[84] *IEEE Standard for Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Prioritization of Management Frames*, IEEE Standard 802.11ae-2012 (Amendment to IEEE Standard 802.11-2012), 2012.

[85] Y. Li, Y. Pan, and P. Wang, "Research and implementation of a mobility management mechanism for wireless sensor networks based on IEEE 802.15.4," in *Proceedings of the IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems*, March 2011, pp. 260–264.

[86] H. Yang and Y. Kim, "SDN-based distributed mobility management," in *Proceedings of the International Conference on Information Networking (ICOIN)*, January 2016, pp. 337–342.

[87] F. Meneses, D. Corujo, C. GuimarÃčes, and R. L. Aguiar, "Multiple flow in extended SDN

wireless mobility," in *Proceedings of the 4th European Workshop on Software Defined Networks*, September 2015, pp. 1–6.

[88] H. Ali-Ahmad, C. Cicconetti, A. de la Oliva, V. Mancuso, M. R. Sama, P. Seite, and S. Shanmugalingam, "An SDN-based network architecture for extremely dense wireless networks," in *Proceedings of the IEEE SDN for Future Networks and Services (SDN4FNS)*, November 2013, pp. 1–7.

[89] S. KukliÅĎski, Y. Li, and K. T. Dinh, "Handover management in SDN-based mobile networks," in *Proceedings of the IEEE Globecom Workshops (GC Wkshps)*, December 2014, pp. 194–200.

[90] M. B. Aleksander, L. Dubchak, V. Chyzh, A. Naglik, A. Yavorski, N. Yavorska, and M. Karpinski, "Implementation technology software-defined networking in wireless sensor networks," in *Proceedings of the IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, September 2015, pp. 448–452.

[91] J. Petajajarvi, K. Mikhaylov, A. Roivainen, T. Hanninen, and M. Pettissalo, "On the coverage of LPWANs: Range evaluation and channel attenuation model for LoRa technology," in *Proceedings of the 14th International Conference on ITS Telecommunications (ITST)*, December 2015, pp. 55–59.

[92] K. Mikhaylov, J. Petaejaejaervi, and T. Haenninen, "Analysis of capacity and scalability of the LoRa low power wide area network technology," in *Proceedings of the 22nd European Wireless Conference*, May 2016, pp. 1–6.

[93] M. Anwander, G. Wagenknecht, T. Staub, and T. Braun, "Management of heterogeneous wireless sensor networks," *6. Fachgespräch Sensornetzwerke*, vol. 2, no. 3, pp. 63–66, 2007.

[94] S. R. Chowdhury, M. F. Bari, R. Ahmed, and R. Boutaba, "PayLess: A low cost network monitoring framework for software-defined networks," in *Proceedings of the IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–9.

[95] M. Yu, L. Jose, and R. Miao, "Software-defined traffic measurement with OpenSketch," in *Proceedings of the Symposium on Networked Systems Design and Implementation (NSDI)*, April 2013, pp. 29–42.

[96] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "OpenTM: Traffic matrix estimator for OpenFlow networks," in *Proceedings of the International Conference on Passive and Active Network Measurement*, April 2010, pp. 201–210.

[97] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha, "Flowsense: Monitoring network utilization with zero measurement cost," in *Proceedings of the International Conference on Passive and Active Network Measurement*, March 2013, pp. 31–41.

[98] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers, "OpenNetMon: Network monitoring in OpenFlow software-defined networks," in *Proceedings of the IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–8.

[99] C. Cao, L. Luo, Y. Gao, W. Dong, and C. Chen, "TinySDM: Software-defined measurement in wireless sensor networks," in *Proceedings of the ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, April 2016, pp. 1–12.

[100] P. Jayashree and F. I. Princy, "Leveraging SDN to conserve energy in WSN- An analysis," in *Proceedings of the 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*, March 2015, pp. 1–6.

[101] M. López, J. Sabater, M. Daemitabalvandani, J. Sabater, J. M. Gómez, M. Carmona, and A. Herms, "Software management of power consumption in WSN based on duty cycle algorithms," in *Proceedings of the EUROCON Conference*, July 2013, pp. 399–406.

[102] C. Zhu, L. T. Yang, L. Shu, T. Q. Duong, and S. Nishio, "Secured energy-aware sleep scheduling algorithm in duty-cycled sensor networks," in *Proceedings of the IEEE International Conference on Communications (ICC)*, June 2012, pp. 1953–1957.

[103] S. Ozen and S. Oktug, "Forwarder set based dynamic duty cycling in asynchronous wire-

less sensor networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, April 2014, pp. 2432–2437.

[104] D. Zeng, P. Li, S. Guo, T. Miyazaki, J. Hu, and Y. Xiang, "Energy minimization in multi-task software-defined sensor networks," *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3128–3139, 2015.

[105] R. Huang, X. Chu, J. Zhang, and Y. H. Hu, "Energy-efficient monitoring in software-defined wireless sensor networks using reinforcement learning: A prototype," *International Journal of Distributed Sensor Networks*, vol. 11, no. 10, pp. 360–428, 2015.

[106] A. M. Abu-Mahfouz and G. P. Hancke, "Localised information fusion techniques for location discovery in wireless sensor networks," *International Journal of Sensor Networks*, vol. 26, no. 1, pp. 12–25, 2018.

[107] H. R. Dhasian and P. Balasubramanian, "Survey of data aggregation techniques using soft computing in wireless sensor networks," *IET Information Security*, vol. 7, no. 4, pp. 336–342, 2013.

[108] W. Ejaz, M. Naeem, M. Basharat, A. Anpalagan, and S. Kandeepan, "Efficient wireless power transfer in software-defined wireless sensor networks," *IEEE Sensors Journal*, vol. 16, no. 20, pp. 7409–7420, 2016.

[109] Y. Lu, X. Huang, B. Huang, W. Xu, Q. Zhang, R. Xu, and D. Liu, "A study on the reliability of software-defined wireless sensor network," in *Proceedings of the IEEE International Conference on Smart City/ SocialCom/ SustainCom (SmartCity)*, December 2015, pp. 129–134.

[110] A. Mahmud and R. Rahmani, "Exploitation of OpenFlow in wireless sensor networks," in *Proceedings of the IEEE International Conference on Computer Science and Network Technology (ICCSNT)*, December 2011, pp. 594–600.

[111] O. Flauzac, C. GonzÃąlez, A. Hachani, and F. Nolot, "SDN based architecture for IoT and improvement of the security," in *Proceedings of the IEEE 29th International Conference on*

Advanced Information Networking and Applications Workshops, March 2015, pp. 688–693.

[112] O. Flauzac, F. Nolot, C. Rabat, and L. A. Steffenel, "Grid of security: A new approach of the network security," in *Proceedings of the 3rd International Conference on Network and System Security (NSS)*, October 2009, pp. 67–72.

[113] Y. Sun, Z. Lin, and Y. Ma, "A lottery SMC protocol for the selection function in software-defined wireless sensor networks," *IEEE Sensors Journal*, vol. 16, no. 20, pp. 7325–7331, 2016.

[114] M. O. Farooq and T. Kunz, "Operating systems for wireless sensor networks: A survey," *Sensors*, vol. 11, no. 6, pp. 5900–5930, 2011.

[115] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer *et al.*, "TinyOS: An operating system for sensor networks," in *Ambient Intelligence*, W. Weber, J. M. Rabaey, and E. Aarts, Eds. Springer, Berlin, Heidelberg, 2005, pp. 115–148.

[116] T. Miyazaki, S. Yamaguchi, K. Kobayashi, J. Kitamichi, S. Guo, T. Tsukahara, and T. Hayashi, "A software-defined wireless sensor network," in *Proceedings of the International Conference on Computing, Networking and Communications (ICNC)*, February 2014, pp. 847–852.

[117] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "Reprogramming wireless sensor networks by using SDN-WISE: A hands-on demo," in *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2015, pp. 19–20.

[118] J. Portilla, A. De Castro, E. De La Torre, and T. Riesgo, "A modular architecture for nodes in wireless sensor networks," *Journal of Universal Computer Science*, vol. 12, no. 3, pp. 328–339, 2006.

[119] Y. E. Krasteva, J. Portilla, J. M. Carnicer, E. de la Torre, and T. Riesgo, "Remote HW-SW reconfigurable wireless sensor nodes," in *Proceedings of the 34th Annual IEEE Industrial Electronics Conference (IECON)*, November 2008, pp. 2483–2488.

[120] S. Natheswaran and G. Athisha, "Remote reconfigurable wireless sensor node design for wireless sensor network," in *Proceedings of the International Conference on Communication and Signal Processing*, April 2014, pp. 649–652.

[121] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the 1st Workshop on Hot Topics in Software Defined Networks*, August 2012, pp. 7–12.

[122] G. Wang, Y. Zhao, J. Huang, Q. Duan, and J. Li, "A K-means-based network partition algorithm for controller placement in software defined network," in *Proceedings of the IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–6.

[123] S. T. Sany, S. Shashidhar, M. P. Gilesh, and S. D. M. Kumar, "Performance evaluation and assessment of FlowVisor," in *Proceedings of the International Conference on Information Science (ICIS)*, August 2016, pp. 222–227.

[124] C. B. Margi, R. C. Alves, G. A. N. Segura, and D. A. Oliveira, "Software-defined wireless sensor networks approach: Southbound protocol and its performance evaluation," *Open Journal of Internet Of Things (OJIOT)*, vol. 4, no. 1, pp. 99–108, 2018.

[125] M. Ndiaye, A. M. Abu-Mahfouz, and G. P. Hancke, "SDNMM - A generic SDN-based modular management system for wireless sensor networks," *IEEE Systems Journal*, pp. 1–11, 2019, to be published.

[126] T. Theodorou and L. Mamatas, "CORAL-SDN: A software-defined networking solution for the internet of things," in *Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, November 2017, pp. 1–2.

[127] F. F. J. Lasso, K. Clarke, and A. Nirmalathas, "A software-defined networking framework for IoT based on 6LoWPAN," in *Proceedings of the Wireless Telecommunications Symposium (WTS)*, April 2018, pp. 1–7.

[128] B. T. de Oliveira and C. B. Margi, "Distributed control plane architecture for software-defined wireless sensor networks," in *Proceedings of the IEEE International Symposium on Consumer*

Electronics (ISCE), September 2016, pp. 85–86.

[129] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, "Fragmentation-based distributed control system for software-defined wireless sensor networks," *IEEE transactions on industrial informatics*, vol. 15, no. 2, pp. 901–910, 2019.

[130] N. D. Phung, M. M. Gaber, and U. Rohm, "Resource-aware online data mining in wireless sensor networks," in *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining*, March 2007, pp. 139–146.

[131] A. Riker, M. Curado, and E. Monteiro, "Neutral operation of the minimum energy node in energy-harvesting environments," in *Proceedings of the IEEE Symposium on Computers and Communication (ISCC)*, July 2017, pp. 1–6.

[132] N. Q. Hieu, N. Huu Thanh, T. T. Huong, N. Quynh Thu, and H. V. Quang, "Integrating trickle timing in software-defined WSNs for energy efficiency," in *Proceedings of the IEEE 7th International Conference on Communications and Electronics (ICCE)*, July 2018, pp. 75–80.

[133] P. K. Sharma, J. H. Park, Y.-S. Jeong, and J. H. Park, "Shsec: SDN based secure smart home network architecture for internet of things," *Mobile Networks and Applications*, vol. 24, no. 3, pp. 913–924, 2019.

[134] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling flow management for high-performance networks," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 254–265, 2011.

[135] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with DIFANE," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 351–362, 2011.

[136] M. Obadia, M. Bouet, J. Rougier, and L. Iannone, "A greedy approach for minimizing SDN control overhead," in *Proceedings of the 1st IEEE Conference on Network Softwarization (NetSoft)*, April 2015, pp. 1–5.

[137] H. I. Kobo, G. P. Hancke, and A. M. Abu-Mahfouz, "Towards a distributed control system for software-defined wireless sensor networks," in *Proceedings of the IECON 43rd Annual Conference of the IEEE Industrial Electronics Society*, October 2017, pp. 6125–6130.

[138] R. Friedman and D. Sainz, "An architecture for SDN based sensor networks," in *Proceedings of the 18th International Conference on Distributed Computing and Networking*, January 2017, pp. 1–10.

[139] A. Asaduzzaman, A. Almohaimeed, and K. K. Chidella, "Shared entry logger to eliminate duplicate requests to SDN controller," in *Proceedings of the IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, January 2019, pp. 579–584.

[140] I. Maity, A. Mondal, S. Misra, and C. Mandal, "Tensor-based rule-space management system in SDN," *IEEE Systems Journal*, pp. 1–8, 2018, to be published.

[141] B. Cheng, J. Zhang, G. P. Hancke, S. Karnouskos, and A. W. Colombo, "Industrial cyberphysical systems: Realizing cloud-based big data infrastructures," *IEEE Industrial Electronics Magazine*, vol. 12, no. 1, pp. 25–35, 2018.

ADDENDUM A MSI FUNCTION

Table A.1. MSI code implementation

```
uint8_t msi( uint8_t * packet_ptr ) {
 uint8_t SI;//Collect rate
 management_state=0;
 nxtstate=0;
 exit_flag=0;
 mgt_action=0;
 management_state = CLASSFICATION;
 while(exit_flag == 0){
    switch(management_state){
     case CLASSFICATION :
      nxtstate = classVerify(packet_ptr);
      management_state=nxtstate;
      break;
     case DENY :
      mgt_action = drop();
      nxtstate = ACTIVE_SLEEP_MODULE;
      management_state=nxtstate;
      break;
     case APPROVAL :
      mgt_action = approve();
      nxtstate = ACTIVE_SLEEP_MODULE;
      management_state=nxtstate;
      break;
     case RES_MANAGEMENT_MODULE :
      printf("resources␣management␣state␣\n");
      nxtstate = Res_mgtModule(packet_ptr);
      management_state=nxtstate;
      break;
```

```
      case ACTIVE_SLEEP_MODULE:
       printf("checking␣on/off␣state␣of␣dest␣node");
       //refer to on/off state context data
       if (state==ON) send_down()
       else wake();
       exit_flag =1;
       break;
      case UNKNOWN :
       exit_flag = 1;
       break;
      default:
       printf ("Unknown␣management␣event.\n");
       exit_flag = 1;
       break;
         } //case
     }  // while
  return mgt_action;
}//msi
```

ADDENDUM B SDN-WISE PARAMETER EVALUATION

Table B.1. SDN-WISE Cooja javascript code

```
            /*
 * SDN-WISE Contiki test script (JavaScript).
 * A Contiki test script acts on mote output, such as via printf()'s.
 * The script may operate on the following variables:
 *  Mote mote, int id, String msg
 */
TIMEOUT(600000, log.log("Total␣PDR␣" + totalPDR + "\n"),
log.log("Total␣Control␣" + totalcontrol + "\n" +  "\n"
+ "the␣average␣delay␣" + averagedelay + "\n"));
packetsReceived= new Array();packetsSent = new Array();
sentTime=new Array();receiveTime = new Array();
delayArray=new Array();
delayaveseqtotal=new Array();

serverID = 1;
nodeCount = 27;
seqCount =9;
Datapacket = 0;
Reportpacket=0;
Beaconpacket=0;
Requestpacket=0;
Regpacket=0;
sentdata=0;
receiveddata=0;
//size=25
totalPDR = 0;
```

```
for(i=0; i<=nodeCount; i++){
sentTime[i] = new Array();
receiveTime[i]= new Array();
delayArray[i]= new Array();
}
for(i = 0; i <= nodeCount; i++) {
    for(j=0; j<=seqCount; j++) {
        sentTime[i][j] = 0;
        receiveTime[i][j] = 0;
        delayArray[i][j] = 0;
        }
packetsReceived[i] = 0;
packetsSent[i] = 0;
delayaveseqtotal[i]=0;
}

function add(accumulator, a) {
    return accumulator + a;
}
while(1) {

YIELD();
delaytotalseq=0;
delayaveseq=0;
totaldelay=0;
averagedelay=0;
delayCount=0;
msgArray = msg.split('␣');
//log.log(msgArray+ "\n");
```

```
/*------------------------------PDR-----------------------------*/
//sent data
if(msgArray[0].equals("Send")) {
    sentdata++;
    }
//received data
if(msgArray[0].equals("Received") && msgArray.length == 10 ) {
    //log.log("message length" + msgArray.length + "\n");
   // log.log("the sender id is " + msgArray[6] +  " the seq is
" + msgArray[3]+  "\n");

    //if(msgArray.length == 7){
    receiveddata++;
    //log.log("received value  " + receiveddata + "\n");
   // }
    }

totalPDR = parseInt((receiveddata / sentdata)*100);
/*------------------------------PDR-----------------------*/
//log.log("the pdr  " + totalPDR + "\n");

/*--------------------NO OF CONTROL PACKETS-----------------*/
if (msgArray[0].equals("[PHD]:")){
    //log.log(msgArray[1] + "\n");
if(msgArray[1].equals("Datapacket")) {
    Datapacket++;
}
if(msgArray[1].equals("Beaconpacket")){
    //log.log("beacon found found \n");
    Beaconpacket++;
}
if(msgArray[1].equals("Requestpacket")){
    Requestpacket++;
}
if(msgArray[1].equals("Regpacket")){
    Regpacket++
}
if(msgArray[1].equals("Reportpacket")){
    Reportpacket++
}
}
totalcontrol=parseInt(Requestpacket+Regpacket+Reportpacket);
/*--------------------NO OF CONTROL PACKETS----------------------*/
```

```
/*-----------------------PACKET DELAY--------------------------*/
//if(msgArray[0].equals("Send")) {
  //  log.log("The msg is from  " + parseInt(msgArray[3]) + "  at time  " +
(time/1000) + "\n");
  //  }
 //log.log("the array value is " + sentTime[parseInt(17)][8] + " \n ");
if(msgArray[0].equals("Send")) {
   // log.log("the array value is " + sentTime[id][parseInt(msgArray[3])] + " \n ");
      sentTime[parseInt(id)][(parseInt(msgArray[3]))] =(time/1000);
    //log.log("the send time is " + msgArray[7] + "\n");
    //log.log("the id is   " + id + "   the seq is  " + parseInt(msgArray[3]) + "
the time is  " + sentTime[id] [parseInt(msgArray[3])] + " same as " + time + "\n");
    }
 if(msgArray[0].equals("Received") && msgArray.length == 10 ) {
     //log.log("the receive time is " + parseInt(msgArray[9]) +  "
" + msgArray[9] + "\n");
     receiveTime[parseInt(msgArray[6])][parseInt(msgArray[3])] =(time/1000);
     }
for(i = 0; i <= nodeCount; i++) {
    for(j=0; j<=seqCount; j++) {
delayArray[i][j]=(parseInt(receiveTime[i][j]) - parseInt(sentTime[i][j]));
}
}

for(i = 0; i <= nodeCount; i++) {
        count=0;
        delaytotalseq=0;
        for(j=0; j<=seqCount; j++) {

            if(delayArray[i][j]>0) {
                count++;//counts the number of elements greater than zero
                delaytotalseq=delaytotalseq+parseInt(delayArray[i][j]);
                    }//endif
            delayaveseq=parseInt(delaytotalseq/count);
             }//endforj
            delayaveseqtotal[i]=delayaveseq;

           }//end
```

```
//log.log("The average delay for node i  " + delayaveseqtotal[5] + "\n");
for(n = 0; n <= nodeCount; n++) {
    if(delayaveseqtotal[n] > 0) {
        delayCount++;
        totaldelay=totaldelay+parseInt(delayaveseqtotal[n]);
        }
    }
    //log.log("the total delay " + totaldelay + "\n");
  averagedelay=parseInt(totaldelay/delayCount);

/*-----------------------PACKET DELAY----------------------------*/
}
```