A Software-Defined Survivability Approach for Wireless Sensor Networks in Future Internet of the Things

A Thesis Submitted in Partial Fulfilment of the Requirements for the

Degree of DOCTOR OF PHILOSOPHY

To Department of Electronic and Computer Engineering
College of Engineering, Design and Physical Sciences
BRUNEL UNIVERSITY LONDON
London, United Kingdom

Presented by

Abdullah Al Atawi

Supervised by

Professor Hamed Al-Raweshidy

December 2018

Declaration

I declare that this thesis is my own work and is submitted for the first time to the Post-Graduate Research Office. The study was originated, composed and reviewed by myself and my supervisors in the Department of Electronic and Computer Engineering, College of Engineering, Design and Physical Sciences, Brunel University London UK. All the information derived from other works has been properly referenced and acknowledged.

Abdullah Al Atawi

December 2018

Dedication

This PhD work is dedicated to my beloved family, wife Amani and children Khalid, Waleed, Shadi, Bassil and Julianna.

Acknowledgments

I would like to express my sincere gratitude to my advisor Prof. Hamed Al-Raweshidy for his guidance, support and advice throughout my research and academic works.

Abstract

The Internet of the Things (IoT) is evolving rapidly, and its significant impacts are expected to affect many application domains. Challenges in areas that humans have been striving to understand, measure, or predict—such as wildlife, healthcare, or environmental hazards—are likely to be addressed by the time IoT emerges.

The underlying elements of IoT are wireless sensor networks (WSNs), which consist of a large number of sensor nodes. In the IoT sphere, sensor nodes represent tangible objects—Things—that monitor changes, collect information, and eventually send it through the Internet to a recipient party. Inherently, however, a wireless sensor node relies on limited computational resources with a limited power source. These undesirable qualities result in a low level of dependability. This research explores the viability of applying the unfolding network programmability concepts to overcome survivability obstacles in WSNs and the IoT. In particular, it examines the viability of software-defined networking (SDN) in network lifetime maximisation, failure detection, and failure recovery problems in WSNs.

Software-defined networking is a new network programmability concept that separates the traditionally-tied control and data planes. It offloads the route computations and management from network devices to a logically centralised controller. This separation directly leads to better allocation of computational resources for the network nodes and allows endless orchestration possibilities for the controller. This thesis proposes an SDN-based solution to increase the survivability and resilience of WSN environments. Following an approach that conforms with the centralised nature of SDN environments and considers the limited resources of the WSN.

A routing algorithm based on A-star was developed for WSNs, then deployed within an SDN environment to maximise the network lifetime. Apart from

finding the path with the lowest energy burden, the algorithm offloads most of the control traffic from sensor nodes to the controller. This algorithm resulted in improved resource utilisation among the nodes due to plane decoupling. Additionally, it increased the lifetime of the network by 22.6% compared to the widely explored LEACH protocol.

This thesis also investigates different failure detection and recovery practices in the SDN architecture. The simulation results show that adopting bidirectional forwarding detection (BFD) with the asynchronous echo mode for WSN in an SDN environment reduces control traffic for failure detection to between 27% and 48%. The thesis also evaluates the performance of multiple recovery approaches when adopting the premises of SDN. The simulation results indicate that path protection, using group tables from the OpenFlow protocol, has a recovery time up to eight times shorter than the restoration time. The results of the study reveal that using protection as a failure recovery technique significantly reduces control traffic overhead.

Table of Contents

List of Figures

List of Tables

Acronyms

6LoWPAN	:	IPv6 over Low-Power Wireless Personal Area Networks
API	:	Application Programming Interface
AODV	:	Ad-Hoc On Demand Vector Protocol
B.A.T.M.A.N.	:	Better Approach To Mobile Ad-hoc Networking Protocol
BFD	:	Bidirectional Forwarding Detection
BS	:	Base Station
CAM	:	Content Addressable Memory
C^4ISRT	:	Command, Control, Communications, Computers, Intelligence, Surveillance, Reconnaissance, and Targeting
CapEX	:	Capital Expenditures
CH	:	Cluster Head
CSMA	:	Carrier Sense Multiple Access
DAG	:	Destination-Oriented Graph
DODAG	:	Destination-Oriented Directed Acyclic Graphs
DoS	:	Denial-of-Service
DSSS	:	Direct Sequence Spread Spectrum
DYMO-low	:	Dynamic MANET On-demand for 6LoWPAN protocol
ECG	:	Electrocardiography
GPL	:	GNU Public License
HaaS	:	Hardware as a Service
IETF	:	Internet Engineering Task Force
ILP	:	Integer Linear Programming
IoT	:	Internet of Things
LEACH	:	Low-Energy Adaptive Clustering Hierarchy
LoWPAN	:	Low-Power Wireless Personal Area Networks
LR- WPAN	:	Low-Rate Wireless Personal Networks
LOS	:	Loss of Signal
MANET	:	Mobile Ad hoc Network
MEMS	:	Micro Electrical Mechanical Systems
MILP	:	Mixed-Integer Linear Program
MPLS	:	Multi-Protocol Label Switching

Acronyms

NBI	:	North Bound Interface
NFV	:	Network Functions Virtualization
NIC	:	Network Interface Card
NOX	:	Network Operating System
ONF	:	Open Network Foundation
OvS	:	Open Virtual Switch
OSPF	:	Open Shortest Path First Protocol
OpEX	:	Operating Expenses
QoS	:	Quality of Service
RF	:	Radio Frequency
RTT	:	Round Trip Time
RSSI	:	Received Signal Strength Indicator
SaaS	:	Software as a Service
SBI	:	South Bound Interface
SDN	:	Software Defined Networking
SDx	:	Software-Defined Everything
SDWLAN	:	Software-Defined Wireless Local Area Network
SDWSN	:	Software Defined Wireless Sensor Network
TCAM	:	Ternary Content-Addressable Memory
TDMA	:	Time division Multiple Access
TCP	:	Transmission Control Protocol
VANET	:	Vehicular Ad Hoc Network
VLAN	:	Virtual Local Area Network
WAC	:	Wireless Access Controller
WMN	:	Wireless Mesh Network
WLAN	:	Wireless Local Area Network
WSN	:	Wireless Sensor Networks

Chapter 1 | Introduction

1.1 Introduction

With the enormous advancement and spread of information technology (IT), people and organisations depend heavily on the flow of information and the availability of technology. Under the umbrella of IT, the three major significant parts are software, hardware, and networking. Harmony in the development of these elements ensures a coherent digital ecosystem. In addition, any lack of innovation in one of these three will adversely affect the growth of the other two. Software has improved dramatically as it has moved from the command line interface (CLI) to higher level programming languages through Web and Web 2.0 technologies to mobile applications. In addition, hardware development has improved dramatically, with increased capacity in memory, storage capacity, size, and processing speed. According to Bell's law, a new class of computer is developed approximately every 10 years that is less expensive[2] and 100 times smaller than its predecessor [3]. However, networking remains restrained by the same Transmission Control Protocol/Internet Protocol (TCP/IP) that was introduced in the early 1980s [4]. With the inherited complexities from TCP/IP becoming increasingly complex by the addition of new protocols, the limitations caused by TCP are the main reasons for the lack of innovation in networking[5], in addition to other problems, including the high costs associated with the operation and management of networks. A clear example was the introduction

of IPv6, when the world noticed that traditional IP is insufficient for the growing number of users. Many services and protocols have been introduced to address the new changes, with a deployment cost of approximately $25 billio[6][7].

Due to growing bandwidth demands and strict limits on the latency of present applications and network characteristics, TCP is constantly being developed. This development has caused a growing number of protocols and requests for comments (RFCs) ($\simeq$ 8,300) [8]. Consequently, networking protocols that were constituted to a very agile standard turned into an inelegant collection of protocols.

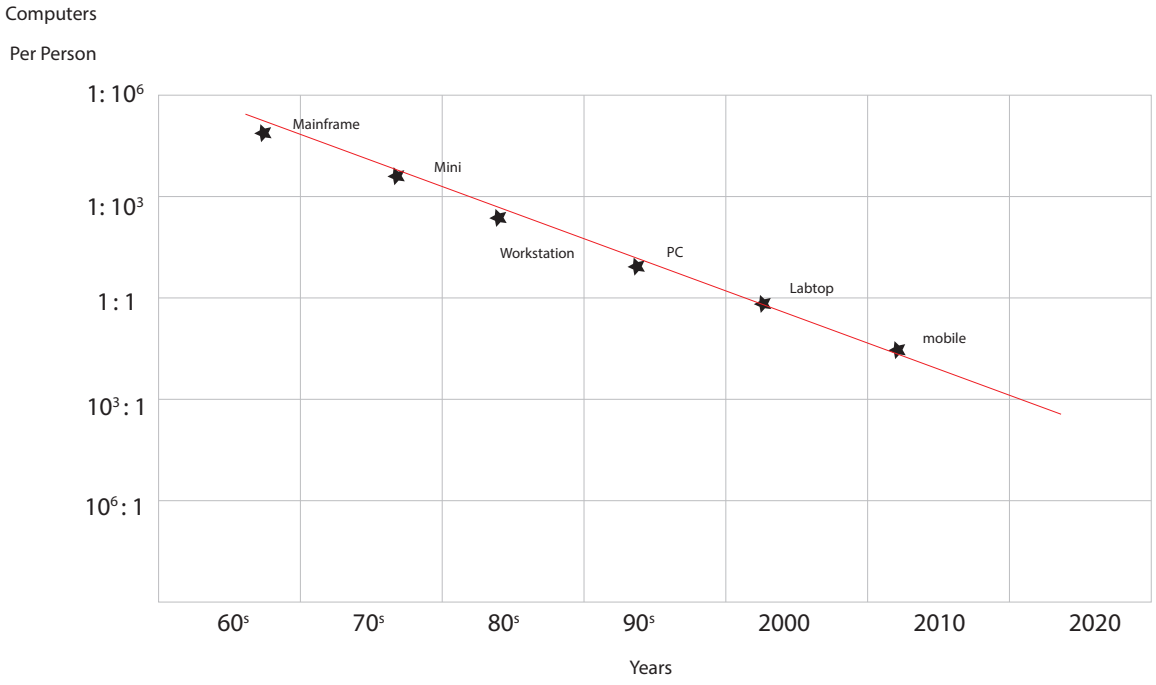

Figure 1.1 – Bell's law: a new computing class emerges every 10 years

New challenges, such as the increased demand from end users for media streaming and mobility, have added to the problem. Telecommunication companies would have to spend more capital expenditure (CapEX) to cope with this demand, including investing in different large-scale networking devices (switches, routers, load-balancers, firewalls,

etc.). Telecom companies would also have to spend more resources to maintain and run different networking devices (OpEX).

1.1.1 Network Virtualisation

To address existing shortcomings and future needs, the concept of network virtualisation has been embraced by a wide range of stakeholders across industry and academia. Two of the prominent network virtualisation approaches that complement each other and aim to change the networking status quo are software-defined networking (SDN) [9] and network function virtualisation (NFV) [10]. NFV follows a 'white box' approach, in which standard x86 computers can perform common networking functionalities, instead of only costly devices. This performance is achieved by virtualising networking tasks on top of a dedicated operating system (OS). Thus, one white box could have a firewall, load-balancer, or router running in a virtual OS inside that box. The result is reduced complexity, OpEX, and CapEX. In addition, this approach has the added benefit of quicker implementation, yielding a more rapid time to market.

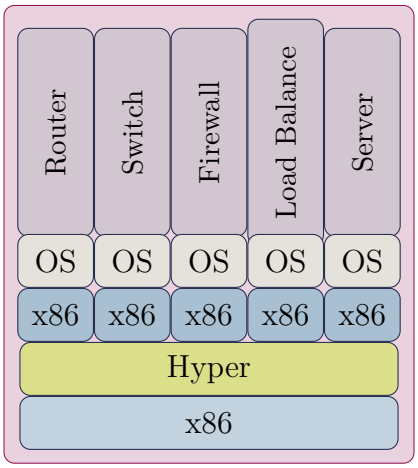

Figure 1.2 – Network function virtualisation: Combines the functionality of various networking devices into a single virtualised architecture reducing CapEX and OpEX

SDN, on the other hand, is based on the separation of control and media planes.

Here, the control plane can be programmed, and the data plane is abstracted.

SDN borrows the concept of abstraction from object-oriented programming (OOP). When a class is defined whenever repeated functionality is performed and common characteristics are not redefined each time. Similarly, SDN treats networking devices as objects that perform basic forwarding functionality. This approach results in maximising hardware and power utilisation.

Other complex and intelligent calculation tasks are moved to a controller. Using SDN, the controller can operate various vendor devices through standard communication protocol, which leads to easier network orchestration and simplified operation and maintenance.

Moving the logic and route computations from the nodes to a central controller results in removing existing complicated calculations of network devices. Furthermore, this change leads to easier network orchestration and maintenance because of the centralisation of the configuration process. For wireless networks, the central controller can visualise a network-wide picture, instead of consuming the power of the nodes trying collaboratively to find the best route for transmission.

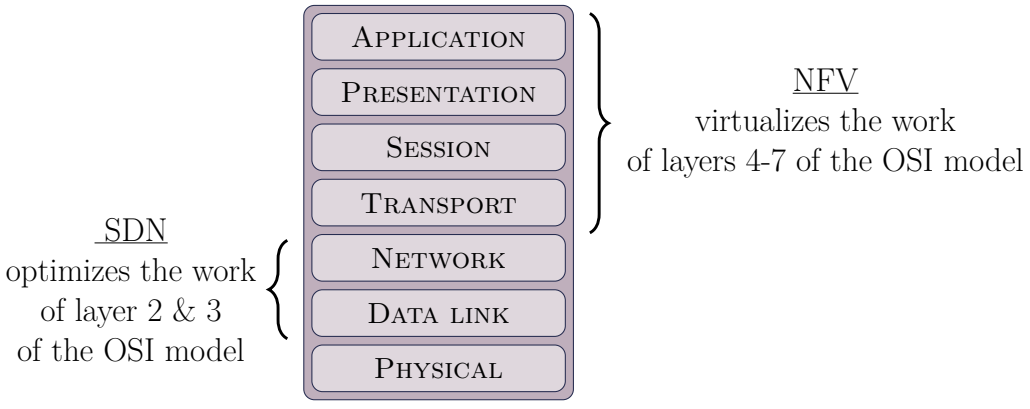

Figure 1.3 – Applicability of software-defined networking and network function virtualisation

Software Defined Networking	Network Function Virtualization
Planes decoupling Control plane is moved from the network devices that are left with the forwarding functionality	**System Architecture Decoupling** Different applications from different vendors work in the same x86 architecture
Virtualise the programmability of network devices	Virtualise and consolidate the functionality of network appliances
Change of network topology	Change of system architecture
Requires changes to interfaces, control module, and applications	The change is only moving applications from dedicated hardware to generic computer
Future: not yet implemented in commercial scale	Present: Prove-of-concept already exists
Standardised by ONF	Not Standardised

Table 1.1 – Software Defined Networking (SDN) vs Network Function Virtualization (NFV)

1.1.2 Characteristics of Software Defined Networking

an SDN [9] is a new paradigm in networking that abandons traditional TCP/IP concepts by abstracting its protocol stack from seven layers to an abstracted forwarding plane and a control plane. The former is concerned only with forwarding data based on a `FlowTable`; whereas, the latter oversees the network and controls the flow between devices from a centralised location. The control plane can be centralised and managed remotely, while the forwarding plane forwards the basic data based on instructions from the control plane. While there are many challenges associated with the design of an SDN [9] , `OpenFlow` [11] is a successful implementation of SDN concepts with its innovative specifications [12]. The simple approach of SDN/OpenFlow has enabled researchers to explore new applications [13] that were not possible in traditional networking [14].

In the past couple of years, most research attention has been directed towards the integration of wired networks with SDN; even giant technical corporations, such as Google, have implemented SDN [15] in their internal wide area networks (WANs). Even so, little literature is available related to the integration of wireless networks with SDN.

The existing prominent implementation of SDN is `OpenFlow`. `OpenFlow` specifications [12] [16] [17] [18] define the architecture and messaging protocol. In `OpenFlow` architecture, there is an `OpenFlow` controller and an `OpenFlow` switch, which communicate through an `OpenFlow` protocol. The controller manages one or more switches, formulates the flows, and programmes `OpenFlow` switches. The controller can also operate in the same namespace as the network applications. SDN promotes the abstraction of the second and third functionalities above for the following benefits:

- Functions such as firewall, load balancing, and intrusion detection become an application that runs in the controller. Even basic functions, such as media access control (MAC) learning are implemented to be performed only by the controller.

- The above have an added benefit of saving customers from buying dedicated boxes for certain functionalities, hence reduced CapEX.

- When network devices are left only for data forwarding, enormous improvement is obtained. The device only has to check the rule of the first packet of the data sent, then will perform either the forwarding or drop and no further processing is required. The rest of the traffic is treated as data flow.

- For the network administrator, a view of the network will be visible so that they only think about the big picture. Host A can communicate with Host B, there is no need for physical configurations on both devices. The controller performs the MAC learning and calculates the route based on the chosen protocol, then installs the rules for every affected network device in the network.

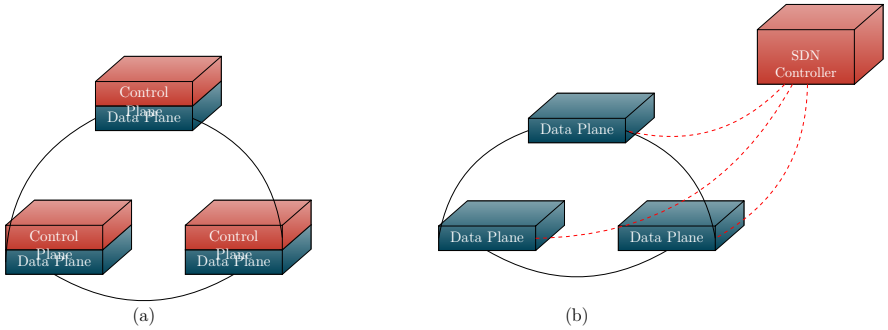

Figure 1.4 – Traditional vs software-defined networking: (a) networking switch performs the control and management tasks in addition to forwarding traffic; (b) the control and management tasks are delegated to the dedicated controller.

1.2 Wireless Sensor Networks

Wireless sensor networks (WSNs) are a special type of network that can sense the phenomenon of interest. Activates in the soundings, such as motion, pressure, or temperature of the environment and electromagnetic and acoustic waves [19]. Initially developed to detect dangerous situations during attacks in battleground and surveillance military applications, WSNs have a much broader array of exciting potential applications.

WSNs and future Internet of Things (IoT) applications can help solve global challenges such as the ageing population, food security, and many others. In addition, the quality of life of the general public is expected to improve thanks to commercial products, including healthcare, security, and environmental solutions. In the healthcare sector, the WSN will be more efficient and will have more operational time; thus, providing patients or ageing people with more freedom to perform their daily tasks [20].

The advanced development of micro-electro-mechanical systems (MEMS) over recent years has brought greater accessibility to small, low-cost, low-power sensor devices. A typical sensor node consists of one or more sensing units, a power source, a wireless communication unit, a processing unit and limited memory. The sensing part of the node

contains one or more sensors and an analogue-to-digital converter (ADC). Typically, the power source is in the form of a battery. Once this power source is depleted it cannot be easily replaced.

Different sensor nodes may contain advanced capabilities according to their intended application. Mobility support, for example, is required in certain applications where WSN node is coupled with mobile entities such as vehicles or people.

Despite their compelling applications, WSNs have some negative attributes that could impact their capability to perform their intended function, most notably, the limited on-board resources of their nodes in terms of battery, processor, and memory. Of these resources, the finite power resource is the most significant challenge in WSNs research. The limited power resource of the node prevents WSNs from capitalising on their fullest potential and causes them to have a deprived functioning lifetime. To address the limited battery resource problem, an enormous number of energy conservation solutions and protocols have been proposed over the past decade[21] [20].

Routing, the process of calculating and using the route for transmitting sensed data in WSNs, is the foremost energy consumption element in WSN node operation [22]. A meticulous design of WSNs routing protocol is strongly desired to extend network lifetime. However, due to other existing limitations regarding processing and memory, routing is unconventional in the case of WSNs. This issue leads to the necessity for innovative solutions to make the IoT and WSN more resilient, dependable, and survivable; that is, providing the expected level of functionality based on existing resources. Additionally, addressing the challenge of WSN failure recovery is yet another survivability and resilient feature that is desired in future IoT solutions.

1.3 Motivations

As discussed previously, WSNs involve several challenges in areas such as energy efficiency, security, computational capabilities, communication bandwidth, data storage, and scalability. These issues may threaten the growth of WSNs. It is expected that by 2020, more than 50 billion devices will be connected to at least one form of network, most of which will be composed of sensors and actuators [23]. Thus, it is imperative to invent methods of optimising WSNs to ensure that their applications in various fields remain effective. One emerging area aimed at countering the traditional weaknesses of WSNs is software-defined networking (SDN), which is a field that separates control logic from the network device to ensure that the only role of the device is the forwarding of information. Software-defined wireless networks (SDWSNs) are the integration of SDN and WSN. In the SDN model, functions that consume a high amount of energy are relocated to a centralised controller from the physical node. In this system, the node does not perform functions such as information management, major processing, or routing [23]. These functions are performed at the application or controller level. Thus, the SDWSN is a new paradigm for improving WSN efficiency by enabling a high level of abstraction for the various functions of the current networks. The benefits of SDWSNs are the following:

- For energy preservation, SDN algorithms can be applied to the controller to replace the resource-constrained nodes.
- An SDN can improve the interoperability of most current WSNs because it can alleviate the low reliability of vendors by enabling the control of infrastructure components from a central point. This allows a single protocol to be used for various elements even when they are from different manufacturers.
- The scalability of WSN can also be improved by SDN through the centralised

enhancement of topological organisation and network efficacy.

- Software-defined networking is poised to improve communication in WSNs by introducing changes such as allowing the centralised controller to effectively manage duty-cycling functionality, media access, and sleep/duty scheduling .
- Security can also be improved by SDN through the centralisation of security management that simplifies the execution and configuration of defence mechanisms.

This dissertation is motivated by the need to develop a solution to existing survivability and resilience problem in WSNs. As most efforts in this field have focused on the traditional networking techniques, these solutions continue to suffer from the challenges that network programmability can solve. With SDN and NFV being embraced heavily in giant networking and telecommunication companies [24], this aggravates the need to push the research to involve wireless networking technologies.

On the other hand, energy and survivability pose a challenge in the face of the advancing IoT. With WSNs being the key element in the IoT, the energy consumption, stability, and agility of the WSNs dramatically affect the IoT. However, the success of SDN in reducing the OpEX, energy consumption, and computational tasks of networking devices stimulates the need to discover its potential in the WSNs domain.

1.4 Aim and Objectives of the Research

The main aim of this thesis is to increase WSNs survivability in the SDN environment by (1) increasing network lifetime and (2) improving the failure recovery process. The goals of this research are addressed through the following objectives:

- Review and study the contrasting features of both SDN and WSNs. To achieve this, a comprehensive literature review of the topics and their new standards and

solutions is carried out. In addition, simulation tools and their documentations are examined.

- Develop a mathematical model for WSNs based on the SDN standard of decoupling the control plane and the data plane. The linear programming model objective is to maximise network lifetime, and the residual energy of the nodes is used as a constraint. The problem was modelled and validated through AMPL optimisation software.

- Develop a heuristic algorithm to prolong WSNs lifetime in the SDN environment, with the goal of making the SDN controller perform a pathfinding process and ensuring a fair resource utilisation of WSNs nodes. An A-star-based algorithm was developed and simulated in MATLAB program[25]. The results were validated be comparing the packet-delivery ratio and network lifetime with the LEACH protocol [26].

- Introduce failure-detection methodologies to a software-defined wireless sensor network (SDWSN) environment. An IoT emulation environment was used to architect and test the bidirectional forwarding detection (BFD) and loss of signal (LoS) detection based on OpenFlow specifications and implemented within Open vSwitch (OvS).

- Devise a failure recovery technique in the IoT emulation environment. Protection with different BFD schemes and restoration are compared regarding various parameters.

1.5 Contributions to Knowledge

Many energy-efficient and failure-recovery protocols in WSNs have been proposed in the past to overcome the above-mentioned problems. With the new emerging technologies

that provide alternative unexplored solutions, this issue raises many new and challenging questions in the WSNs and IoT research domain. Of the newly unexplored issues, this thesis addresses the following two major research topics.

1.5.1 WSN Lifetime Maximisation in the SDN Environment

Software-defined networking introduces the concept of decoupling the control plane 'routing and management' from the data plane 'traffic forwarding'. This new paradigm in networking has the potential to reduce the amount of energy required in calculating the route in WSNs. On the other hand, SDN depends on a centralised controller to perform the calculations, which could conflict with the decentralised nature of traditional WSNs. This issue presents new, unexplored challenges when combining the two environments. When addressing the problem of lifetime maximisation, it is necessary to efficiently balance the energy consumption of the nodes through the centralised SDN controller, in addition to selecting the best path. With a network-wide perspective within the controller, its duty is implementing a lightweight and efficient centralised routing criteria.

Contribution: Practical solutions that extend WSN lifetime were investigated by addressing the resource-constrained nature of wireless nodes when migrated into SDN, becoming SDWSN. An A-star-based routing algorithm was proposed to extend WSNs in the SDN environment. The algorithm, in addition to finding the shortest path for WSN nodes, benefits also from the SDN controller to obtain fair distribution of traffic to maximise resource utilisation among the nodes. A simulation model of the algorithm was developed, in which it is deployed within the SDN controller. The algorithm is then evaluated against existing WSN energy-saving algorithms.

1.5.2 IoT Failure Recovery in the SDN Environment

Failure detection and recovery is another survivability concern in all types of networks. In this sense, even though SDN is potentially a promising solution for IoT solutions, certain technical challenges must be addressed when deploying IoT solutions based on the SDN architecture. One challenge is the choice of failure detection technique that adapts SDN technology and, at the same time, respects the unique nature of WSNs. Another challenge is the selection of an SDN-native feasible recovery technique under the restrictions of IoT solutions under limited resources.

Contribution: The failure detection problem is addressed, and a comparative evaluation of existing SDN failure detection techniques is presented. The performance of different failure detection techniques is evaluated in an IoT simulation environment. The control traffic overhead and failure detection time are the main criteria in this evaluation. The selection of any particular technique is an application-based decision, which our work contributes towards clarifying. For failure recovery, the performance was measured for various combinations of existing failure recovery techniques that adhere to SDN specification constraints and suit IoT applications. At the centre of this contribution lie the identification of architectural aspects and considerations of `OpenFlow` SDN protocol when applied to a resource-constrained environment such as IoT.

1.6 Thesis Organisation

This dissertation addresses a range of open challenges in the area of survivability in WSNs and proposes embracing network programmability through SDN architecture. Chapter 2 introduces SDN and WSNs and their distinguishing characteristics and highlights their potential. Chapter 3 reviews recent literature on wireless network applications in the

SDN environment. Since the integration of SDN is still a work in progress, we highlight the most relevant work regarding wireless fault tolerance and survivability problems. Chapter 4 addresses the lifetime energy problem in WSNs and presents SDN as a solution for lifetime maximisation. The problem is mathematically formulated and solved then a heuristic solution based on an A-star algorithm is presented. Chapter 5 addresses the error recovery problem in the SDWSN environment, technical challenges are addressed, and then a novel error recovery model is utilised to solve this problem. This work then concludes in Chapter 6 with a summary of the findings and future research directions.

Chapter 2 | Background

2.1 Internet of the Things: An Overview

The Internet of things (IoT) can be defined as the network of computing devices incorporated in different types of objects that send and receive data through the Internet. The IoT allows the transfer of data over a network without the need of human-to-computer or human-to-human interactions. An increasing number of organisations use IoT to automate processes and increase the quality of customer service, among other benefits. The number of interconnected devices is expected to reach more than 34 billion devices by 2021 [27]. According to a BBC Research report [28], both the IoT hardware and service sectors are expected to reach $17.3 billion by 2022 at a compound annual growth rate of 21.7%. These estimations show the high potential impact of IoT on the entire global economy and suggest it will likely disrupt many industries.

The technology and applications of IoT are likely to be one of the major drivers of innovation and investment in the communication sector, likely leading to the introduction of many new services [27]. The IoT has applications in several domains, including biometrics, transportation, industry, agriculture, business, infrastructure, energy, health care, and home appliances. In the next years, it is expected that IoT will continue to change these domains by automating, digitising, digitalising, optimising, and transforming processes, business models, and industries.

2.2 Internet of the Things Applications

The IoT is rapidly penetrating multiple fields such as medicine, agriculture, power systems, and industrial automation. This section briefly discusses some popular applications of IoT to emphasis its importance in the evolution of our world into a smart world [29, 30].

2.2.1 Applications in Environmental Monitoring

Internet of the things have several environmental applications, such as the prediction or detection of coal-mine flooding, forest fires, earthquakes, tsunamis, cyclones, water quality, gas leakage, and volcanic eruption [29]. That this technology helps in the prediction and early detection of these environmental disasters allows the relevant authorities to take precautionary and safety measures to reduce the impact on the environment. Sensors used in environmental monitoring collect data and convey them to the base station through the Internet. For instance, in air pollution monitoring, the chemical reactions that result in pollution are detected using sensors in an IoT system [29]. This system uses an air quality chemical index to compare the obtained information with values on the index to determine the required intervention.

2.2.2 Applications in the Agricultural Sector

Internet of the things applications in the agricultural sector play a crucial role in minimising wastefulness of resources and in conservation of the environment [30]. A WSN collects, processes, and produces important information that can be used to assess favourable conditions such as sowing density, humidity, temperature, pressure, fertiliser, and insecticides levels, among other input needs [31]. The sensed data are then transmitted to the human interface via cloud computing technology or the Internet. Personnel at the main sink examine the data, and if required, take action to ensure that the production cost is low

and the crop yield is high. In the real-time monitoring approach, the data are inspected to determine the status of various environmental and climate conditions that affect the production of agricultural products [29]. To improve the precision of the outcome, the processed data of different topologies, such as throughput, load, and delay, are calculated and simulated. For instance, in crop management, IoT is used to detect pH levels, temperature, and humidity inside the rumens of cows, goats, and sheep [31]. The collected data are then transmitted wirelessly to an exterior receiver node through a condensed measuring probe known as a bolas. The objective of this wireless system is to detect the presence of ailments, such as acidosis, hyperthermia, and blotting [31].

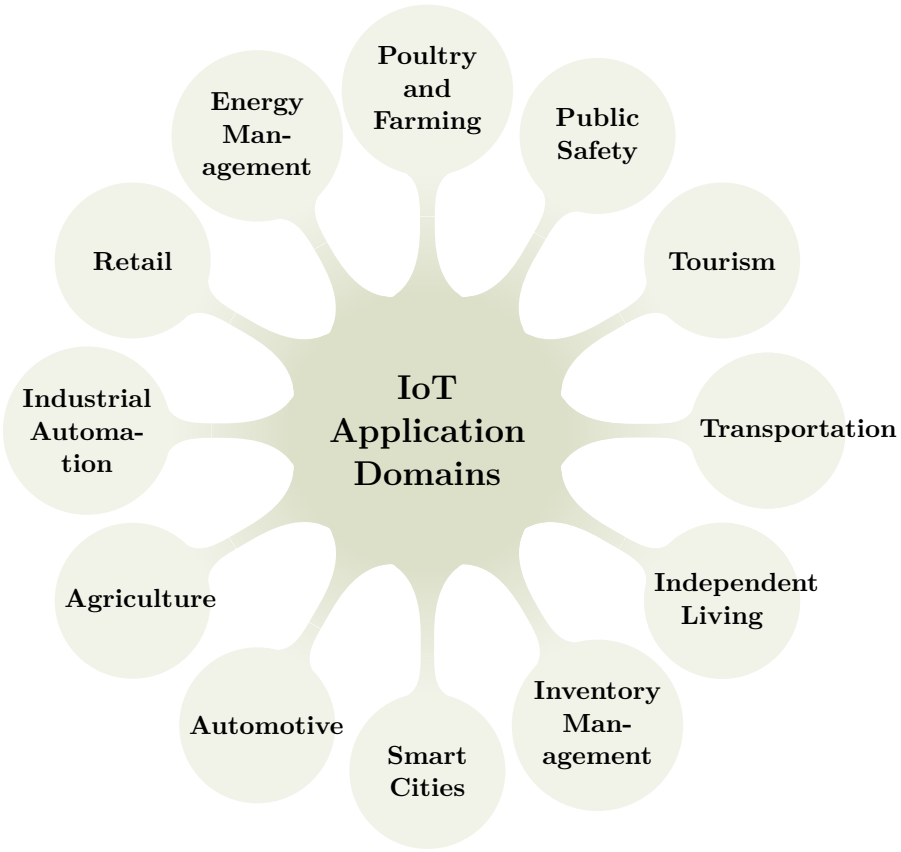

Figure 2.1 – IoT application domains

2.2.3 Applications in the Healthcare Sector

Currently, applications of WSN and IoT have extended to the healthcare sector where the technology is used to detect physiological parameters such as the temperature, pH, and heart rate of patients [29]. Therefore, it is possible to detect various ailments such as heart attack, various heart diseases, Alzheimer, and acidosis. The IoT used in the healthcare sector is designed in a way that enables sensing of various physiological parameters, and the data are then transmitted to the care provider for further analysis and diagnosis [31]. Internet of the things applications have also been applied in telemonitoring arrangements, such as a cognitive sensor network for the elderly. This home-based system comprises an adequate number of cognitive wireless sensors that are used to detect the usage of electrical devices, water flow, and bed usage patterns [29].

Moreover, utilising IoT in the medical field enables real-time monitoring of the health status of a patient. For instance, electrocardiogram (ECG) sensors have been used to monitor patients in real-time who suffer from chronic heart diseases [29]. In this case, the patient wears the ECG sensor, which continuously sends data to the healthcare centre or clinician and allows continuous monitoring of the patient.

2.2.4 Applications in Wildlife

Internet of the things are used in the wildlife sector to track the location of an animal in the wilderness. A sensor in the IoT system can be attached to the body of a wild animal, such as at the neck, a limb, or an ear, to enable identification of the animal's position as well as its feeding and movement patterns. One of the examples is the Zebra Net that is used to monitor the movement of zebras in the wilderness [29]. The Zebra Net sensor is attached to a given part of the zebra's body to monitor the animal's location and position and the type of food consumed by the animal. Typically, a collar-mounted sensor is attached to the neck of the animal, which helps continuously track its movements. In

addition, endangered species such as elephants, rhinos, lions, and other wild cats can be protected using WSN sensors. In this case, sensors are attached to the animals to help determine their exact location all times, which enables the wardens to monitor and control the animals' movements.

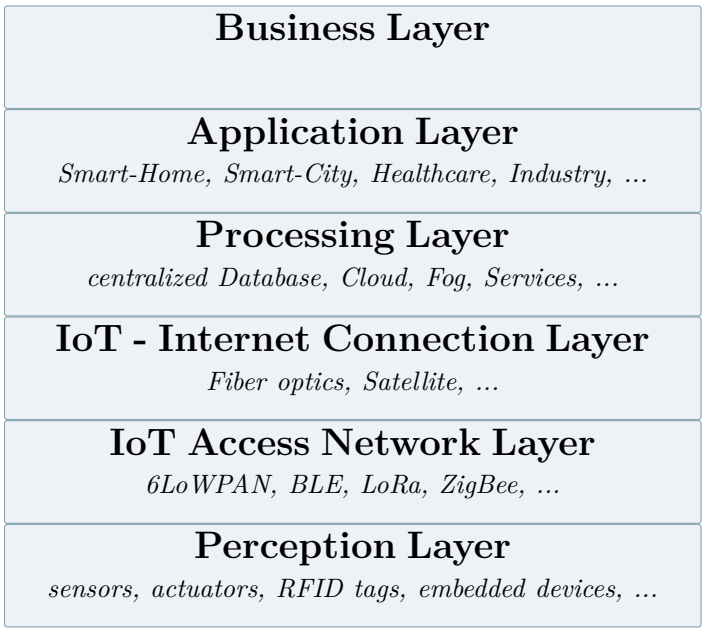

Figure 2.2 – The six layers IoT architecture

2.3 Internet of the Things Architecture

There is currently no universally accepted architecture design. The most commonly used designs include three-layer, five-layer, cloud and fog-based architecture [32]. The three-layer architecture consists of an application, network, and perception layers. The cloud and fog-based architecture consists of transport, security, storage, pre-processing, monitoring, and a physical layer. Compared to the other proposed architectures, the six-layer architecture is considered to be a better and wider view of IoT architecture [32]. It separates the network layer in previous proposals into access network and Internet connection layers. The layers are perception, access network, Internet connection, processing, application, and business.

The perception layer consists of actuators, radio frequency identification (RFID) tags, sensors, and other embedded devices that are usually small form-factors with constrained power sources. These are the sensors that collect data from the environment. Most IoT devices use a battery as the energy source, though energy-harvesting or mains-powered devices exist.

The access network layer is responsible for the transmission of data collected by the perception layer. It encompasses several communication technologies, including Bluetooth Low Energy, Ethernet, and WiFi. Different communication technologies provide different transmission rates and data ranges and have different levels of costs and power consumption.

The Internet connection layer consists of a border gateway or router that connects the inner network to the Internet via communication technologies such as satellite or fibre optics.

At the processing layer, the collected data is processed, analysed, and stored. Storage and processing systems are either distributed or centralised. At this layer, middleware services are provided based on the data that is analysed and processed.

The application layer is where applications in various deployment areas use data obtained from the processing layer.

The business layer is responsible for coordinating with the whole IoT system to extract appropriate data for creating business models and other relevant data that helps in making important business decisions.

2.3.1 Communication Technologies

There are two categories of WSN based on functionality. They include short-range communication technologies and long-range communication technologies [23]. Wireless sensor networks in the short-range communication category are Bluetooth, Zigbee, Bluetooth LE, and Z-Wave. The following list describes these technologies:

2.3.1.1 Short-Range Communication Technologies

Bluetooth was created in 1994 and is based on the IEEE 802.15.1 standard. It uses an frequency-hopping spread spectrum (FHSS) radio transceiver and functions within a diameter of 10m [23]. Its energy consumption is very high compared to other short-distance WSNs. Bluetooth is applied in medicine and mobile technologies.

Zigbee is a short-distance WSN that was created in 2003 and is based on IEEE 802.15.4. It uses a direct-sequence spread spectrum (DSSS) radio transceiver and requires a low amount of energy compared to Bluetooth. It can function within a diameter range of 70m to 300m and is mainly used in monitoring and control [23].

Bluetooth LE, or BLE, is another short-distance WSN that is based on IEEE 802.15.1. It operates within the 2.4GHz frequency band and was created in 2011. Although Bluetooth and Bluetooth LE function within the same diameter range, the latter consumes less power [23]. Thus, Bluetooth LE is more cost-effective than Bluetooth. The technology is also applied in the healthcare and public transport sectors.

Z-Wave is a short-distance WSN that consumes lower energy than Bluetooth. In terms of energy consumption, it is comparable to Zigbee and Bluetooth LE. Unlike Bluetooth and Zigbee, Z-Wave operates on a varying radio frequency between 800–900MHz. Hence, it is less likely to experience interference [23]. It can function within a 100m diameter and is mainly used in smart homes, particularly in combination with IoT.

A wide variety of technologies are used in IoT systems, and a few commonly used technologies are considered IoT enablers. Radio-frequency identification (RFID) forms the part of the perception layer of the IoT architecture and is responsible for object identification. Radio waves are used to transfer the identity of the object in the form of a serial number [33]. Based on power provision, RFID tags can be categorised as active, passive, and semi-passive.

Barcodes encode information through a combination of bars. These are machine-readable codes that contain object-specific information. They are read by lasers or cameras and have become popular for object identification. Quick response (QR) code that is a variation of the barcode has been particularly adopted in a wide range of applications [34].

Near field communication (NFC) can transfer data between objects wirelessly. The object must be close (around 20 cm) [35]. The tags are similar to RFID and contain a small amount of data. The tag can be rewritten. It is an effective way of communicating when devices are close to each other, and it does not require line of sight.

2.3.1.2 Long-Range Communication Technologies

Some long-distance WSNs include NB-IoT or narrowband-IoT, which is a technology using the 4G network. As a result, it has wide and excellent coverage, even underground [36]. The technology can connect several devices to one another concurrently. It uses low energy, and its battery can last up to 15 years. Also, it is not possible to interfere with this technology because it uses licensed frequencies such as 3G and 4G. Also, the installation of NB-IoT is cheaper [36]. However, its low bandwidth hinders the transmission of large quantities of data within a short time.

LTE-M is comparable with NB-IoT. It has a wider bandwidth compared to other LTEs but offers a narrow coverage. In addition, it transmits data at a reduced speed but

more frequently than typical 4G networks [37]. One of the advantages of this network is the ability to provide real-time information.

Long Range (LoRa) is a long-distance WSN designed to connect devices that are battery-powered [36]. It achieves this connection through a local, national, or global network at a very low cost. Consequently, it requires a network point, similar to cellular M2M or WLAN. It can be accessed from a distance of 16km above ground or underground [36]. Unlike NB-IoT, it utilises a licence-free network called ISM, meaning that there is the possibility for interference by other users. Similar to NB-IoT, there is a legal limit for data usage.

Sigfox allows the exchange of small amounts of data and does not require frequent communication. It can be accessed from 40km and consumes low energy; when it is on stand-by mode, two AA batteries can last for two years [36]. Unlike NB-IoT, Sigfox uses a license-free network called ISM, which can lead to interference when a high number of users are on the same network. In addition, data usage by this WSN is legally limited.

2.4 Characteristics of Software-Defined Networking

Software-defined networking (SDN) [9] is a new networking approach that removes the path processing from network devices and allows them to save their computing powers for data forwarding. In SDN, the computation of the path does not occur in the wireless nodes; instead, it happens in an external controller. This approach has been applied successfully in wired networks with significant results. Moving the logic and computations from the nodes to a central controller results in reducing the power consumed by the nodes to calculate the path. In addition, the central controller can provide a clear picture of the network, instead of consuming the power of the nodes by trying collaboratively to find the best route for transmission.

The control plane can be centralised and managed remotely, while the forwarding plane forwards the basic data based on instructions from the control plane. Although there are many challenges associated with the design of an SDN [9] , `OpenFlow` [11] has been a successful implementation of SDN, with its comprehensive yet agile specifications [12] . The novelty of SDN/OpenFlow has enabled researchers to explore new applications [13] that were not possible in traditional networking [14].

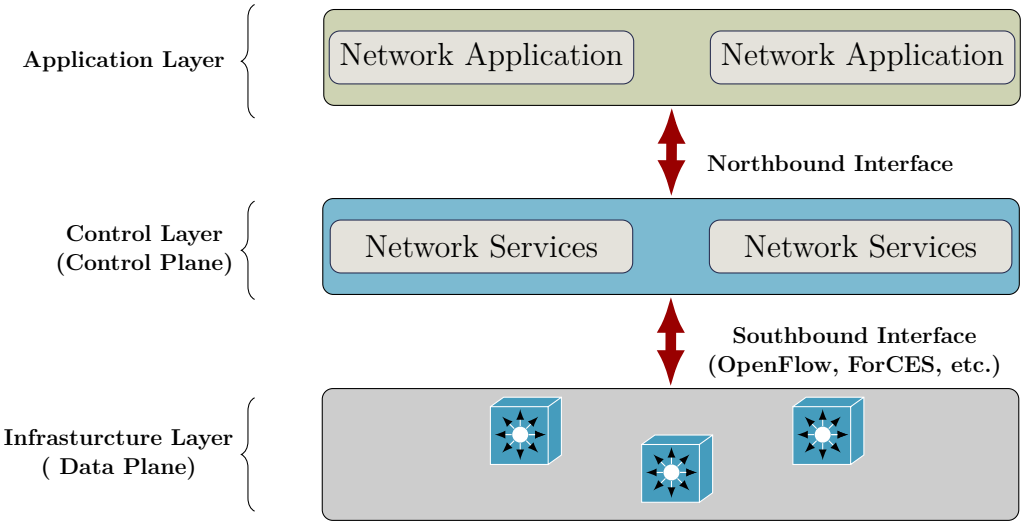

Figure 2.3 – SDN

2.4.1 SDN Controllers

In the SDN paradigm, the controller is the brain of the operations. The data plane depends on the controller's instructions to manage the flow of traffic. The controller translates network policies into configurations for individual network devices to create, read, update, or delete flow entries. These configurations can be set beforehand, proactively. Reactive flow entries are installed when the network device consults the controller to determine the next course of action for traffic flow. Multiple controllers can be clustered for higher reliability and network scalability. The placement of controllers and the quantity depends on the topology and operational requirements of the system. The first SDN controller was NOX, which was developed using C++/Python. Since then, many

SDN controllers have been implemented, mostly using either JAVA, C, or Python and addressing different networking requirements. Table 2.1 lists the most commonly used and reliable controllers. Ryu [40] controller is an open source component-based Python

Controller	Implementation	License	Developer
NOX [38]	C++	GPL	Stanford University
POX [39]	Python	Apache	Nicira
Ryu [40]	Python	Apache	NTT, OSRG
OpenDaylight [41]	Java	EPL	Industry consortia
Floodlight [42]	Java	Apache	Big Switch
Beacon [43]	Java	BSD	Stanford University
MUL [44]	C	GPL	Kulcloud
Trema [45]	C/Ruby	GPL	NEC

Table 2.1 – Different SDN controllers

SDN controller that is used throughout this work. This controller has the advantage of running various controller functionalities and integrations with OpenStack[1] thus, it is widely used in cloud orchestration applications [46][47]. Ryu utilises a first-in first-out priority for events in its application. Ryu is better suited for research applications compared to NOX because of its active development and support for other southbound interfaces other than OpenFlow, such as OF-config [48] and Netconf [49]. However, Ryu's performance is slower than other SDN controllers when deployed in large-scale networks [50][47].

2.4.2 OpenFlow

Put simply, SDN treats network devices as forwarding boxes regardless of the vendor. SDN separates the forwarding process of data packets from the routing process 'control process'. The controller performs the routing and management operations. The controller has a global view of the network layer, because it is located between the application layer and the physical switches in the network. The actual rules are installed in network devices

[1] OpenStack controls large pools of computers, storage, networking and multi-vendor hardware resources in a datacenter.

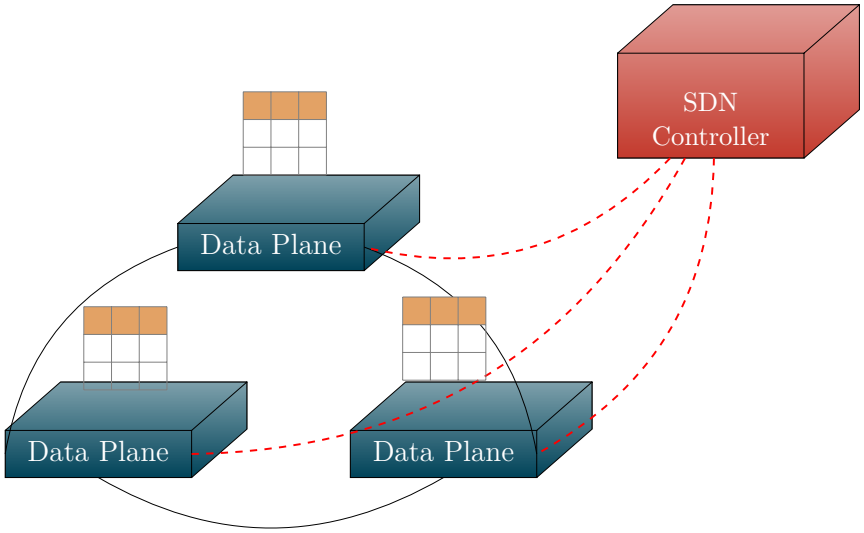

Figure 2.4 – In software-defined networking, OpenFlow protocol is used for communication between controller and switches.

by the controller, usually through `OpenFlow` protocol [12].

The existing prominent implementation of SDN is `OpenFlow`. OpenFlow was initially proposed by Stanford University [51], and it is now standardised by The Open Networking Foundation ONF [24]. `OpenFlow` specifications [12] [16] [17] [18] defines the architecture, messaging protocol. In `OpenFlow` architecture there an `OpenFlow` controller and `OpenFlow` switch which communicate through `OpenFlow` protocol. The controller manages one or more switches, formulates the flows, and programmes the `OpenFlow` switches. The controllers can also operate in the same namespace as the network applications. There are several implementations of `OpenFlow` controllers. They vary in terms of programming language, target users, and learning curve. The prominent ones are NOX / POX, Ryu, Floodlight and OpenDaylight.

2.4.2.1 OpenFlow Switch

`OpenFlow` switches are agents that communicate with the controller and process the controller's messages. An `OpenFlow` switch contains `FlowTables`, `GroupTables`, Ports and an `OpenFlow` channel. To achieve this functionality a software implementation of

the OpenFlow standard, such as OvS software [52], must be installed in the standard switch.

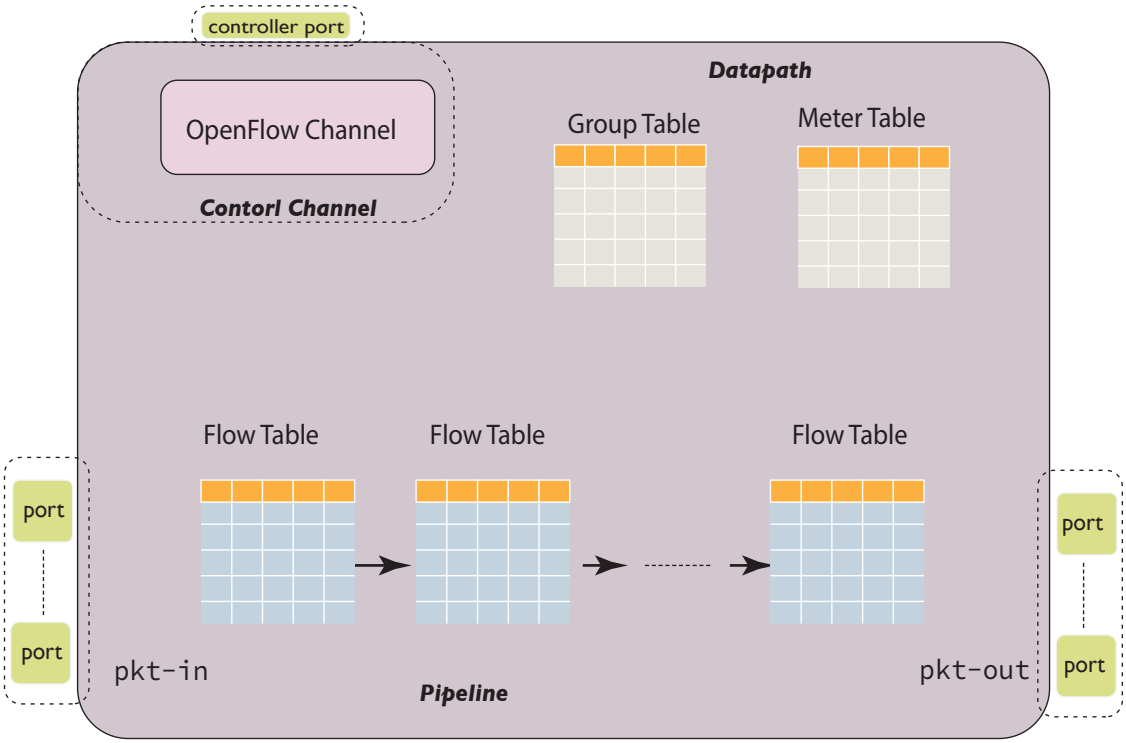

Figure 2.5 – OpenFlow Switch Components

Every entering packet is matched to flows in FlowTables. Flows within FlowTables contain sets of actions that are applied to each packet that matches the rules.

Header Fields	Counters	Actions	Priority

Table 2.2 – FlowTables Fields of OpenFlow Standard 1.0

With the introduction of OpenFlow standard 1.1, Actions were replaced with Instructions. With Actions, if there is a match, the action is performed directly. While in Instructions, more sophisticated operations can be performed according to the instructions which can be: (1) immediate actions , (2) a set of actions, or (3) a change

to the pipeline processing to go to a certain `FlowTable` or `GroupTable`. Additionally, `OpenFlow` 1.1 allows for more tables compared to `OpenFlow` 1.0, and introduces the concept of `GroupTables`. This concept was introduced to perform common networking tasks, which can be of the following types.

- **ALL** a list of actions can be performed, used for flooding and multicasting.
- **SELECT** only one set of actions in the group is executed.
- **INDIRECT** instructions are executed in the next hop.
- **FAST FAILOVER** the live/functioning instruction list is executed.

Match Fields	Priority	Counter	Instruction	Timeout	Cookie	Flag

Table 2.3 – FlowTables fields of OpenFlow standard 1.1

Ingress port	MAC D.A.	MAC S.A.	Ether Type	VLAN I.D.	IP src	IP Prtcl	IP DSCP	L4 src prt	L4 dst prt

Table 2.4 – Possible match fields

Rules or Match Fields Rules are matching criteria based on ingress ports, Ethernet, IPv4, or TCP source or destination addresses. This makes `OpenFlow` make rules for Layers from 1 to Layer 4.

- Input Port (L-1)
- Ethernet (L2-Data)
- IP (L3-Network), subnet mask availability
- Transport Protocol and TCP/UDP ports (L4-Transport)

Actions The measures that takes place once an event occurs in the data-path pipeline. For example, when a packet first arrives, and there is no match, the rule can be 'drop it' or 'consult with the controller'. It is not required for an `OpenFlow` enabled switch to support all actions. The actions available are shared with the SDN controller during the bootstrapping process through the `OFPT_FEATURES_REPLY` message. Examples of actions include:

- Forward/OUTPUT (All, Controller, Local, Table, IN_port, Flood)
- Enqueue
- Drop
- Modify-Field
- SET_NW_SRC , SET_NW_DST

Counters, keep statistical information about flows, such as Received Packets, Received Bytes, and Duration. A closer look at the simple but efficient structure of `OpenFlow` tables reveals that it can provide the `OpenFlow` switch with the capability to function as a switch, router, or firewall. For example, using the field `[L4 Dst]` to filter TCP or UDP traffic, combined with a drop action, we have firewall functionality. Furthermore, by setting the field `[ IP Dest addr]` to forward matched flow to a certain port, the result is router functionality. In addition, utilising counters for traffic size or duration simplifies the work of networking engineers to perform sophisticated quality of service (QoS) requirements.

2.4.2.2 OpenFlow Protocol Messages

The communication between an `OpenFlow` switch and a controller may be one of the following types of messages. The relevant messages are defined below, the `OpenFlow`

specifications contains a comprehensive list [53].

- OFPT_HELLO: This is a symmetrical message to trigger the connection process between the OpenFlow switch and the controller.
- OFPT_FEATURES_REQUEST: A controller-to-switch message created by the SDN controller to request the features of the OpenFlow switch.
- OFPT_FEATURES_REPLY: A reply to the OFPT_FEATURES_REQUEST that contains a list of the switch's features.

The above three messages, in sequence, describe the process of OpenFlow bootstrapping. Then, if successful, the SDN controller installs configurations and further installs proactive flows using the following two messages.

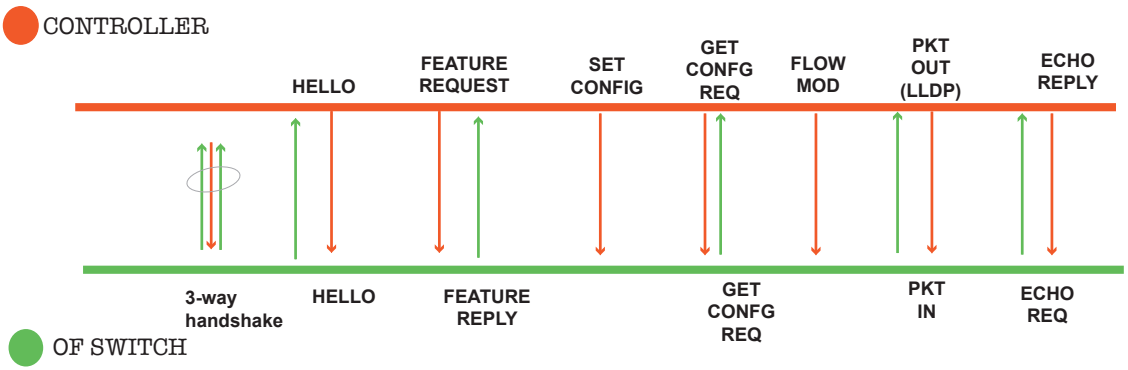

Figure 2.6 – Messages exchanged between the SDN controller and the OpenFlow-enabled switch

- OFPT_SET_CONFIG: A controller-to-switch message to modify the settings of the switch.
- OFPT_FLOW_MOD: A controller-to-switch message generated by the controller to modify the FlowTables of an OpenFlow switch.

Other relevant messages for this study include the following messages.

- OFPT_ECHO_REQUEST: A symmetrical message used to implement a keep-alive of the connection.
- OFPT_ECHO_REPLY: A symmetrical message in response to the previous message.
- OFPT_PACKET_IN: A switch-to-controller message generated by the `OpenFlow` switch on forwarding a packet to the controller.
- OFPT_PORT_STATUS: A switch-to-controller message generated by the switch to inform the controller about a change in state of a physical port.
- OFPT_PACKET_OUT: A controller-to-switch message that permits the SDN controller to inject a packet into the data plane of a switch.
- OFPT_STATS_REQUEST: A controller-to-switch message generated by the controller to query the contents of one of the switch's `FlowTables`.
- OFPT_STATS_REPLY: A switch-to-controller message that is generated in response to an OFPT_STATS_REQUEST message enclosing the requested information about the flows in the switch's `FlowTable`.

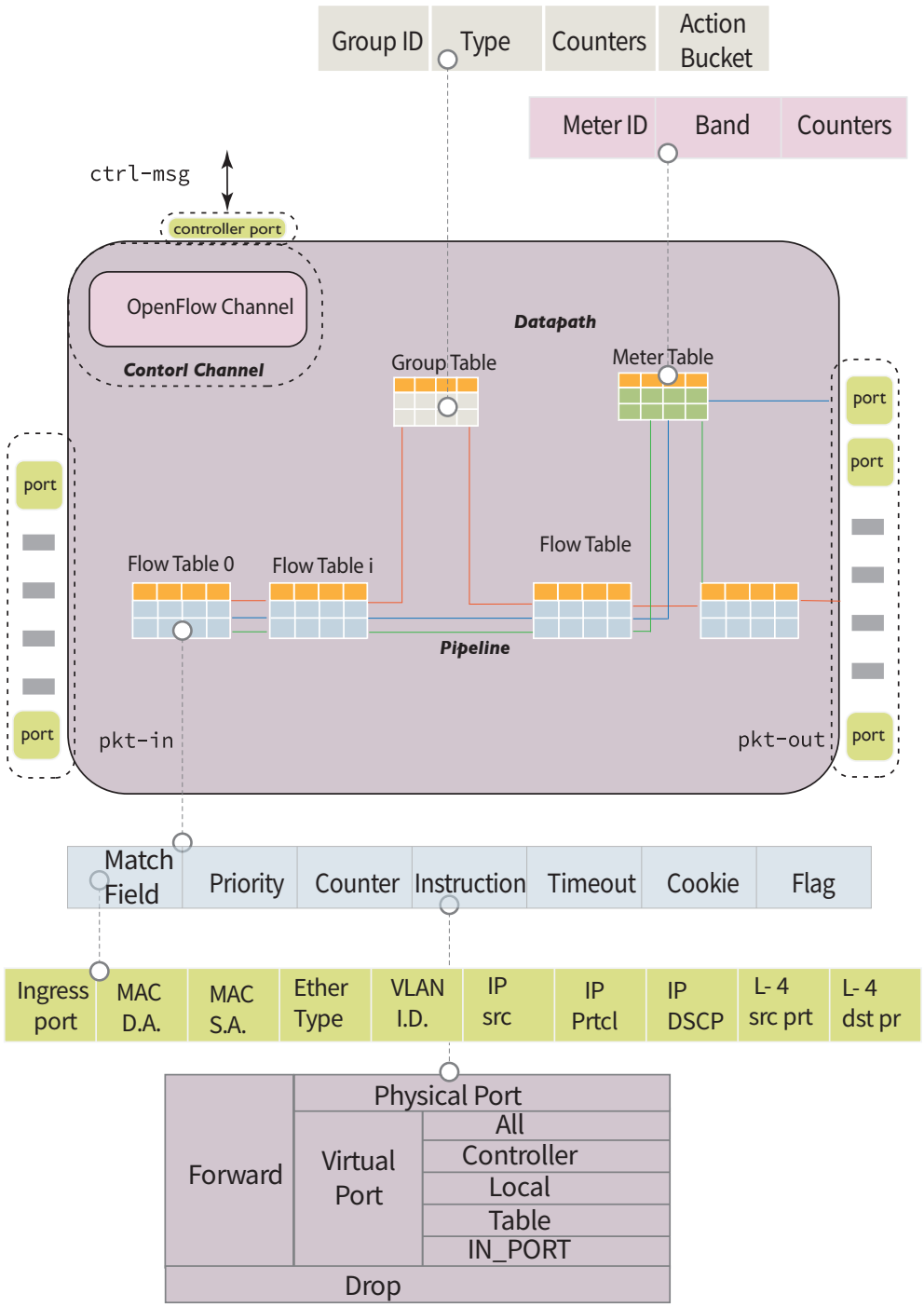

Figure 2.7 – Detailed OpenFlow switch components with the main fields of each table

2.5 Wireless Sensor Networks (WSN)

A wireless sensor network (WSN) refers to a self-configured network, and unlike other networks, it does not require an infrastructure to function [54]. A WSN is typically used to monitor environmental or physical conditions such as pressure, temperature, vibration, pollutants, motion, or sound. A WSN comprises several thousand sensors called nodes, which use radio signals to communicate with each other. Each node contains computing and sensing devices, power elements, and radio transceivers [54]. Wireless sensor nodes cooperate to transmit the data to the base station or the main sink where the data are observed and analysed. The base station functions as an interface between the network and the user. All nodes in a WSN are characteristically resource-constrained, such that they have low processing speed, limited communication bandwidth, and inadequate storage capacity [54]. However, the discovery of new architectures that contain heterogeneous devices and the recent developments in WSN have eliminated these limitations and considerably expanded the spectrum of applications of this technology. Advancement in WSNs has been rapidly increasing, leading to a wide range of applications in areas such as agriculture, manufacturing, environment, and healthcare. A WSN is deployed in a random, structured, or unstructured way depending on the network application and the terrain [54].

2.5.1 Functionality

A WSN system consists of several sensor devices that can compute, communicate, and sense wirelessly. The basic unit of a WSN is the sensor node, which is a tiny low-cost scattering device with low energy consumption and wireless communication and local processing capabilities [54]. A sensor node combines its capabilities to sense, communicate, and compute input to produce valuable information for the user. A sensor node is attached to a battery that acts as the power supply source. A WSN also contains

a central processing unit or a microprocessor that digests the collected data. The processor is attached to a memory chip on which the information is stored. Every sensor node has a radio transceiver or an antenna that allows one node to communicate with its neighbours. Sensor nodes sense and collect data on events happening nearby, which is then processed by the microprocessor and transmitted to the base station [54]. Various components of WSNs have limitations that may hinder the proper functioning of the system. The battery has a limited power supply because it cannot store enough power to sustain the system. The memory chip has a limited storage capacity, meaning that it can only store a small amount of information. Radio transceivers compared to infrared or optical communication are not robust and are not interference-free. The transmission of data can be affected when the sensor devices generate vast volumes of redundant data, and similar data from numerous devices might be aggregated, reducing the number of transmissions [54].

2.5.2 Wireless Sensor Network Standards

The fundamental requirements for WSN include a high level of administration, low cost, reliability, low power and maintenance, and easy deployment. Consequently, various WSN standards such as Zigbee Pro ISA 100.11a, WirelessHART RPL, 6LoWPAN, and LoRaWAN have been developed by companies such as Zigbee Alliance, HART Communication Foundation, and the International Society of Automation [55]. These standards are based on IEEE 802.15.4.

2.5.2.1 Zigbee

Zigbee is a WSN standard developed by Zigbee Alliance. It is applied in a wide range of sectors as a monitoring agent. Similar to WirelessHART, Zigbee is based on IEEE 802.15.4 and has an operating frequency of 2.4 GHz [55]. The standard can form three types of topologies – mesh, star, and cluster – and defines its network layer for various ap-

plication layers and networking capabilities that provide a framework for communication and development application. There are two implementation options for this standard: Zigbee, for smaller networks, and Zigbee Pro, for larger networks [55]. The key features of the Zigbee standard are low cost, long range, low energy usage, robustness, easy deployment, self-healing, and self-organising. All these features are present in WirelessHART. However, Zigbee can only interoperate with Zigbee devices, and unlike WirelessHART, Zigbee lacks frequency diversity and is exposed to security threats. The standard uses a static channel, which increases the chances of interference and delay [55]. Another limitation is the lack of path diversity, meaning that in the event a path breaks down, a new one must be created. Unlike WirelessHART, it is not easy to achieve scalability with Zigbee due to the presence of ad hoc on-demand vector routing (AODV).

2.5.2.2 6LoWPAN Standard

6LoWPAN stands for IPv6 over low power wireless personal area network and was created by the International Engineering Task Force (IETF) based on IEEE 802.15.4 [55]. 6LoWPAN is designed to be applied in sectors that require many sensor nodes to cover a large geographic region using low power, low cost, and few computations. The incorporation of IPv6 allows connectivity to the Internet at low data rates and a low duty cycle. 6LoWPAN has the following benefits: a smaller packet size; easy management of the network due to IPv6; mobility support; reliability; reduction in latency, header fragmentation, and compression; and high scalability as a result of the adoption layer [55]. Some of the drawbacks of 6LoWPAN include susceptibility to link failures, weak security, and interference.

2.5.2.3 WirelessHART

WirelessHART is a WSN standard created by the HART Communication Foundation (HFC). It uses IEEE 802.15.4 to process information automatically, and its operational

frequency is 2.4 GHz [55]. The standard combines a direct sequence spread spectrum (DSSS) with a frequency-hopping spread spectrum (FHSS) to transmit data efficiently. The functionality of IEEE 802.15.4 is extended at the link layer by the addition of 10ms timeslots and the use of the time-synchronised mesh protocol (TSMP). This protocol utilises time division multiple access (TDMA) to reduce the frequency of collisions and allow channel access [55]. The WirelessHART standard can self-organise and self-heal.

Additionally, the WirelessHART standard is robust, highly secure, energy-efficient, simple to execute, and interoperable with other devices manufactured by HART [55]. It also easily achieves scalability using either multiple access points or a WirelessHART Gateway. While the standard has several benefits, it also has drawbacks. For instance, it cannot be interoperated with other standards that are based on IEEE 802.15.4. In addition, WirelessHART has only dedicated links and has provisions associated with shared links. The standard uses a scheduling algorithm called a centralised scheduling algorithm [55].

2.5.2.4 IPv6 Routing Protocol Standard

The IPv6 routing protocol (RPL) for low power was developed by the IETF [56]. The standard is mainly applied in IoT environments. Since its first proposal, the standard has been improved extensively to allow applications in diverse environments and scenarios. One of the areas in which RPL has been extensively applied is the control of congestion [57]. In lossy and low power networks (LLNs), congestion is a problem and can result in a reduction in network lifetime. Some of the benefits of RPL include low energy consumption and low cost.

Chapter 3 | Literature Review

3.1 Introduction

The world we live in today is transforming into a 'smart world' rapidly with the fast-paced technological advancements. The IoT is an important milestone in this context. Everyday devices are connected to each other over a network and are able to communicate with each other and behave intelligently.

A simple working of an IoT system can be described in three steps [33]:

1. Data sensing and communication: The objects collect specific information from their surroundings with the help of sensors and communicate it to the processing centre. The data picked up by the sensors may range from temperature and humidity level to vibrations.

2. Action: The information collected by the sensors is received by the processor, and after being processed, a decision is made regarding an appropriate response.

3. Feedback: The results yielded by the action taken in step 2 are communicated to the administrator.

3.2 Low Power Protocols

There are several standardised protocol stacks used for low power networks. These include narrowband Internet of things (NB-IoT), Long Range (LoRa), and Long-Term Evolution Machine Type Communication (LTE-M). For instance, the authors of [58] supported the use of a stack for lossy and low power IoT networks that make use of the standards proposed by IEEE and IETF. These standards are meant to offer energy-efficient medium access control layer (MAC) and physical layer (PHY) operations for lossy and low power networks.

The routing protocol for low-power and lossy networks (RPL) protocol, is a tree-based approach that is defined by IETF as the standard protocol for low-power and lossy networks (LLNs) [59]. This protocol organises the network in a directed acyclic graphs (DAG) that encompasses more than one destination-oriented directed acyclic graph (DODAG). Each DODAG is used to represent a routing tree and is constructed from objective functions that use routing metrics to calculate the best path between the DODAG root and the nodes [60].

Culler and Berkeley [61] proposed a hybrid routing protocol for LLNs (HYDRO). This approach combines centralised control with local agility, and the network nodes produce DAGs as the default routes of the border router.

Kim, Montenegro, Park, Chakeres, and Perkins [62] proposed a dynamic mobile ad-hoc network on-demand for the 6LoWPAN protocol (DYMO-low). This protocol uses routing messages as control messages and was fully projected for the use over IEEE 802.15.4. The routing messages are not fragmented, and the choice of the best path to forward a message is realised considering the route cost and the link quality indicator value. A different protocol is the Lightweight On-demand Ad hoc Distance-vector Routing Protocol – Next Generation (LOADng) [63]. The LOADng protocol creates a route only

when a node sends a message to another node, and the process of creation is realised through control messages. The route reply message is used by the destination to attend the received route creation requests. In order to create a bi-directional path, an route reply message can require a reception confirmation.

3.3 Energy Efficiency in Internet of the Things

Energy efficiency in IoT is an important factor that must be addressed when considering the large number of energy-constrained devices that are connected to the Internet. In order to increase the life span of different sensor nodes, energy conservation efforts should be made at different levels of an IoT infrastructure [32]. With this in mind, the following section provides a review of energy conservation approaches for IoT devices.

Sensors play a key role in IoT and are a special consideration in energy efficiency solutions [32]. Several models of energy-efficient IoT have been proposed. For instance, Wut, Yang, Li, and Li [64] proposed an energy-efficient model for a physical layer that can also provide deployment benefits and a naïve optimisation principle for energy efficiency.

Al-Kahtani [65] presented a cluster-based sleep schedule model where clusters were formed and devices assumed to have the same energy deployed. Different devices were used as principal cluster heads (PCH) or alternative cluster heads (ACH),-which were used as fault tolerance in case PCH devices failed. In each cluster, some devices are active and others are not, reducing energy consumption.

Ahmad, Asim, Khan, and Singh [66] introduced the concept of green IoT. Which requires specific hardware and software that are designed to consume less energy while maintaining the same level of performance. For example, green wireless networks use sensor nodes with small power and storage capacity and can be achieved by green energy conservation, radio optimisation, and routing techniques. Green data centres are based

on renewable energy resources and use protocols [67].

Al Ridhawi, Aloqaily, Kotb, Jararweh, and Baker [67] proposed a new fog-to-fog (F2F) collaboration approach for simple and complex multimedia service delivery that creates short-term service-level-agreements offered to cloud subscribers while maximising fog profit gains and client satisfaction. The proposed solution consists of a learning mechanism that uses online and offline simulation data to generate guaranteed workflows for new service requests. The evaluation of the model showed positive gains in service delivery at a reduced power consumption for cloud and fog data centres.

3.4 Energy Conservation in Internet of the Things

There are multiple challenges that affect the design of an IoT architecture. Issues such as security, mobility, energy usage, business model, and interoperability must be taken into account.

3.4.1 Energy Conservation Issues

Energy conversation is a difficult task, especially in situations where access to resources is reduced while operational costs increase. There are several issues that must be considered when searching for energy conservation solutions, including analysis and planning, energy engagement, maintenance, and employee commitment and satisfaction. Important variables to be considered are traffic fluctuation, collision listening, overhearing listening, protocol overhead reduction, and idle listening [32].

Traffic fluctuation in the network traffic can cause congestion, and when there is a maximum traffic on the network, even if the network is working at the maximum efficacy, congestion will reach a significant apex level [68].

Collision listening occurs when a node receives multiple data packets at the

same time. In this situation, the received data becomes irrelevant and retransmission is necessary, causing additional energy consumption [69].

Overhearing listening can occur during data transmission when interference occurs with the neighbouring node. This issue is especially relevant for the nodes within reach and consumes a substantial amount of additional energy [70].

Protocol overhead reduction is an important consideration, as the protocol header information consumes significant energy resources. There are several ways to reduce this energy consumption, including cross-layering approaches, adaptive transmission periods, and optimised flooding [71].

Idle listening refers to the fact that energy is consumed even when the node is in idle mode and active and waiting to send data. In order to address this issue, one option is to turn back the sensor nodes from sleeping to active mode after processing a wake-up signal or after a predefined time interval [72].

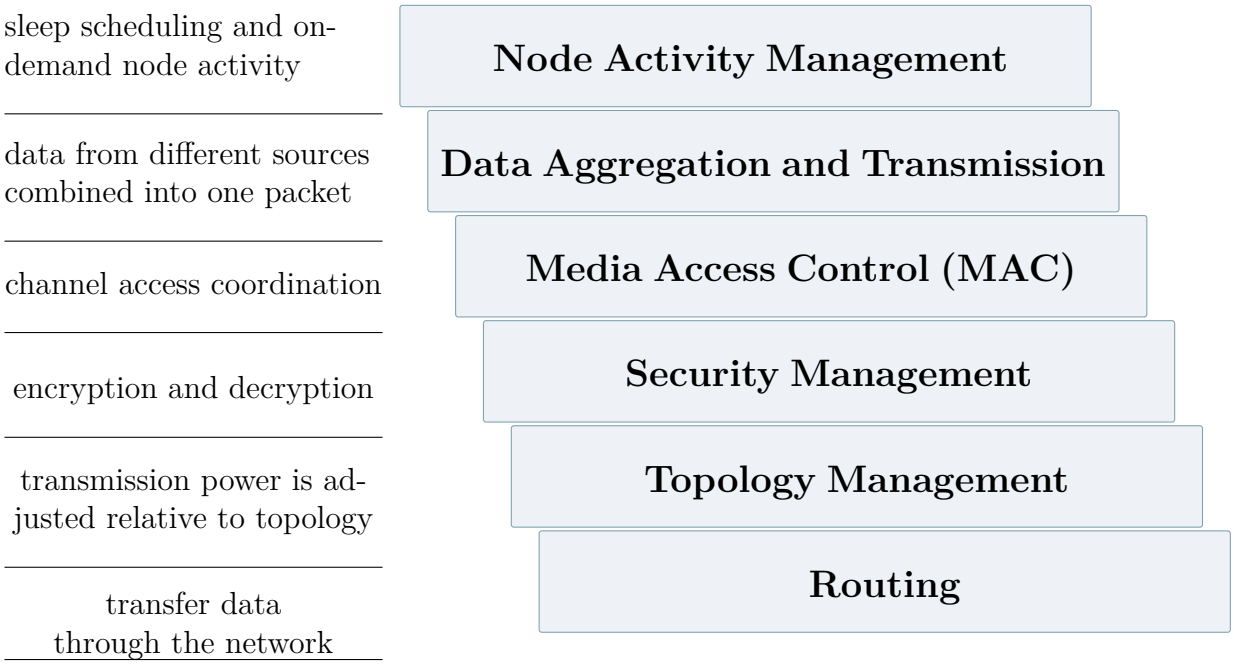

Figure 3.1 – IOT layers

3.4.2 Energy Conservation Approaches

There are several energy conservation approaches that can be used for IoT, and the decision of a specific approach over others will have to be based on multiple variables. For instance, routing protocols can take three forms, proactive, reactive, and hybrid routing [73]. In proactive routing, the routes are routinely updated for each node in the network. In a reactive protocol, routes generated on-request for the node. Hybrid routes combine both approaches [74].

Node activity management can be used to save energy in idle time spans by using sleep scheduling, namely setting a priori sleeping and wake-up timing. On-demand node activity, in contrast, is not scheduled, as the node remains in an active state and has some functionality [75].

Another approach for saving energy is through the design of improved MAC protocols. The MAC protocol describes a set of frame transmission rules and channel access procedure. The MAC standards can operate as non-beacon-enabled or beacon-enabled. The non-beacon-enabled mode is always awake while the latter defines super-frames in which nodes are active just for a small fragment of a super-frame [76].

Another area of activity where energy can be saved is security management. At the present time, there is a need for improved cryptographic algorithms that could be implemented in the application layer of security management. Saving energy in other layers can be challenging, as each layer of IoT requires security measures. For instance, in the perceptual layer, confirmation is fundamental in order to avoid illegal access to the node, while in the network layer, confidentiality and integrity must be ensured at the cost of energy consumption [77].

In order to reduce the node power consumption, several topology management

practices can be used. There are four control types, namely graph-based topology control, Gabriel graph, relative neighbourhood graph, and localised minimum spanning tree. Graph-based control occurs where information about sensor separations and their position can be accessed. In a Gabriel graph, every sensor participating in the network is aware of the sensors located in their proximity and their location. A relative neighbourhood graph is a straight line connecting two points. A localised minimum spanning tree processor works as a power diminished network by using a minimum spanning tree over the network in a dispersed way [78].

Because data transmission is more costly than data processing, one approach is to aggregate data sets inside clusters, where data coming from different sources is consolidated in one packet. This approach helps reduce redundancy and the number of transmissions. When power control is used in the transmission process, it is possible to save a significant amount of energy [79].

3.5 Software Defined Networking

IoT solutions can benefit from the newly emerging concept in networking called SDN. SDN is based on the separation of the control and media (or data) planes: the control plane can be programmed, and the data plane is abstracted. SDN treats networking devices as objects that perform basic forwarding functionality. This approach results in maximising hardware and power utilisation.

Other complex and intelligent calculation tasks are moved to a controller. Using SDN, the controller can operate various vendor devices through standard communication protocol, which leads to easier network orchestration and simplified operation and maintenance. Moving the logic and route computations from the nodes to a central controller results in removing existing complicated calculations of network devices and to easier network orchestration and maintenance because of the centralisation of the configuration

process. For wireless networks, the central controller can visualise a network-wide picture instead of consuming the power of the nodes by trying collaboratively to find the best route for transmission. This chapter highlights recent developments in these fields to address the survivability problem in the IoT.

3.5.1 Survivability in Software Defined Networking

A few recent studies have addressed survivability issues, mostly in wired networks. In[80], the backup path is used once a failure is detected. The optimised path is calculated based on link capacity and congestion constraints. The solution is inspired by the MPLS tagging approach, in which a packet discovers the failure. The authors assume that the packet will detour and make subsequent packets use the backup path. The authors have not implemented the proposed solution, but claim that it could be implemented using OpenState. The problem is formulated to reduce the length of the path of the tagged backup packet, and the model is solved using AMPL-Cplex.

The authors in [81] apply the concept of protection to an existing network by constructing a new network considering the capacity factor. The authors begin by constructing the primary graph, in which each link is assigned a probability of failure, and a primary capacity. Then, a backup network is constructed using the same vertices (nodes) using new backup links. So, those links provide a backup path for the primary links and allocate capacity. The problem was formulated as an ILP for backup capacity provisioning, then solved using simulated annealing to use networks with a larger number of nodes. The assumptions in the work include links being bidirectional, primary links failing with probability, and backup links not failing. However, the authors provide a draft of how to consider unreliable backup links.

In [82], the protection is performed for the links, not the entire network as in the previous work. The primary network is calculated and, for each switch, the set of

OD (origin, destination) is based on capacity and demand. The problem is formulated as an ILP and solved using a heuristic algorithm. The authors performed a simulation in comparison with other methods.

The authors in [83] prove that routing tables can provide guaranteed resilience against a single failure. In [84],a study of an SDN local fast failover mechanism is presented. The authors report the effect of fast failover and the trade-off between robustness and load balancing. Another resilience approach is presented in[85], in which a higher degree of availability and fixable routing control is achieved by moving the responsibility of connectivity to the data plane. The authors introduce the notion of data-driven connectivity (DDC), in which forwarding decisions should guide the packets to their destinations.

A shortest path backup and a greedy algorithm for controller selection are proposed in [45] for wireless mesh networks (WMNs) in an SDN environment. The aim is to extend the network edges and to reduce packet loss. The simulation was based on 20 nodes, but recovery time in case of failure was not documented.

To detect the connection status and to minimise the effect of controller loss, a test bed was implemented recently [9]. In the study, OvS were modified within WSNs to perform recovery in addition to detecting node failure. The authors in [86] propose a method of dynamic monitoring of the entire wired network in the case of flow changes. That is, once a new rule is inserted in the flow table, how long will it take the network to update the affected switches running OvS. In addition, event log function and visualisation were implemented.

Ref.	Recovery Type	Algorithm	Time	media	metric	Cplex	SDN sim.
[87]	tagging	MLIP	30 s<	wired	link capacity	Yes	No
[88]	Link protection	LIP-SA		wired	Link capacity	Yes	No
[89]	Link protection	heuristic	0.4 s <	wired	Link capacity	No	Yes
[90]	Node protection	Shortest		wireless	distance	No	No

Table 3.1 – Results overview

3.6 Software Defined Networking (SDN) Applications in Wireless Networks

Different implementations of SDN in wireless networks have been presented to solve various problems in wireless networks. Research work has dealt with applications intended to achieve a specific goal, such as to improve the network's performance. Other applications include load balancing, rural networks, and wireless sensor networks.

3.6.1 Performance Improvement

Wireless networks share the common goal of utilising the available bandwidth and reducing delay. These two factors affect the performance of the network. With the successful deployment of SDN in wired networks, it has been tested in various wireless networks using both simulation and test-bed experimental approaches.

In WMNs, traffic is transferred in a multi-hop fashion from one wireless router to another, which creates a major problem regarding the client's mobility. When a client moves within a WMN, it is difficult to maintain end-to-end mobility. Current routing protocol solutions, such as the ad-hoc on demand distance vector (ADOV) protocol [91] and the better approach to mobile ad-hoc networking (B.A.T.M.A.N) protocol[92], lack flexibility and do not provide flow-based routing. The first work to address this issue and introduce`OpenFlow` as a solution was a study by Dely, Kassler and Bayer (2011) [93]. Their experiment results indicate that `OpenFlow` is a promising option, with an average

outage of 200 ms during handover.

In [94] , OMNeT++ [95] was used to conduct a comprehensive investigation of WMNs operating with`OpenFlow` technology. The simulation results are compared with traditional open shortest path first (OSPF) routing. The aim of the experiment was to evaluate the performance regarding throughput, end-to-end delay, and packet loss. The results indicate that `OpenFlow` performed worse in terms of throughput when the number of users exceeds 20. However, for less than 20 users, all other performance parameters, including throughput, indicate greater benefits from using `OpenFlow` rather than traditional routing. For example, the end-to-end delay was improved by 47% based on the simulation of different applications over a variety of different users and simulation times.

One of the factors that affects the performance in wireless networks is the handoff process. Traditionally, handoff is performed through the measurement of the received signal strength indicator (RSSI) when the client is moved to the access point (AP) with a stronger signal. This technique does not guarantee an acceptable performance for the entire network. An alternative solution using `OpenFlow` was proposed in [96]. In what the authors call`OpenFlow` Access Point (OFAP), each client uses its service set identification (SSID) as its basic service set identification (BSSID). The former is the network name that is used to identify the network, while the latter is the MAC address of the device that is providing the service. By using SSID as BSSID, the perception of the client's device is that there is only one AP in the entire network. The experimental setup in this work included Kulcloud [1] as the controller and OpenWrt [2] to configure Open vSwitch [52] in the nodes. The performance regarding throughput and delay was measured for the proposed handoff procedure. The experimental results in this work indicate an increased throughput of 26.7% compared with the traditional RSSI, with no

[1] www.openmul.org
[2] www.openwrt.org

interval of disruption during the handoff process. Although this approach solved the problem of seamless handoff, the extra traffic generated/processed is not discussed. In addition, in some applications, security issues could arise from sending the same traffic to multiple devices.

A common problem among routing algorithms on WSNs is the lack of a global view of the network's topology and the high number of advertisement packets. Yuan, Fang and Wu [97] proposed the use of`OpenFlow` with link-state routing to overcome those challenges. The authors used Raspberry Pi as the WSN nodes and installed Open vSwitch to get an`OpenFlow` test bed. ADOV-UU [3] , a version of the famous ADOV routing protocol, was used. The paper does not indicate exactly which controller was used. However, the authors claimed that the new design allowed better utilisation of the available bandwidth. An advantage of this approach is that it provides a fault-tolerant design of the system (i.e., the system can use the co-existing ADOV protocol if the controller fails).

3.6.1.1 Load Balancing and QoS

The design of SDN enables its control plane to oversee all the devices throughout the network; therefore, it is a potential solution, in particular, for QoS and load balancing.

In [98] , a test bed was used to compare load balancing between an SDN-based WMN (wmSDN) and the traditional optimised link state routing (OSLR) [99] protocol. The design of the wmSDN included an Open vSwitch and a POX controller [4] . In this approach, an in-band traffic solution is presented using the same network for data and control traffic, but with different IP subnets. Both approaches have the same time of hand off between two different experimental flow paths. The approach that used SDN provided better performance in distributing the goodput (the useful data rate without

[3] Aodvuu.sourceforge.net
[4] openflow.stanford.edu

TCP/IP header) between the two available paths. The available links/paths are fully utilised because of the `OpenFlow` controller's ability to oversee the available paths and to make the correct decisions for data flow.

A prototype was used to examine load balancing for WMNs by implementing `OpenFlow` technology [100]. A Linux-based x86 platform was used with Floodlight as a controller, and the B.A.T.M.A.N. routing protocol [92] was used for the initial network topology setup. The design of the network incorporated a custom-designed monitoring-tool 'code' that redirected traffic whenever the performance was degraded. This redirection of traffic was performed through an `OpenFlow` controller that pushed certain flow entries into a congested-nodes flow table. The results of the experiment indicate that there was increased throughput when redirection occurred; in addition, jitter was decreased by 25%. The redirection between paths consumes 5 ms, which is a drawback of this configuration if the goal is not load balancing.

3.6.1.2 Rural Connectivity

Software-defined networking could be the answer to many challenges in rural networks, such as low-cost devices and very limited operational budgets. Rural connectivity does not require demanding networks in terms of features. The main requirements are only simple connectivity with low-cost equipment and operations. Herein, SDN is proposed as a viable solution for this type of network. However, since SDN is still an emerging technology, a major concern is the availability of trained technicians during the first deployment or when additional nodes are required.

Hasan et al. [101] [102] propose the integration of SDN concepts to be used in rural networks. The authors posit that using this integration will only cost the enterprise for the setup, after which it will obtain the benefits of easy management. The authors analysed the business benefits of the integration in detail, positing that it would cause a

paradigm shift in rural networks' business.

Another work that advocates this idea of simplifying the operation and management of rural networks through SDN is [103]. The author of this work went further than [101] [102] by proposing architectural and design concepts for such implementation. A test was employed for these ideas that consisted of a POX as a controller, with an OvS installed in the wireless nodes. No loss of data was reported using this technique, but it failed to provide any extra features that might have been expected from a new design for a wireless backhaul. However, the objective of rural wireless networks is to provide service at a low cost for low-performance devices, and this design can achieve those goals.

3.6.1.3 Heterogeneous Networks

One possible application for which SDN has been evaluated in recent research is its use in mixed or heterogeneous network environments, according to [104] and [105]. The first study discusses the possibility of integrating infrastructure-based and infrastructure-less networks in what is called heterogeneous-SDN (H-SDN). The authors posit that H- SDN, when correctly implemented, could solve existing problems regarding capacity sharing and compatibility between different device vendors. Supported by the flexibility of SDN, this solution could enable innovation and interoperability in such networks. However, the second study uses this technique for a noble objective that is disaster resilient. Prototypes were installed in three locations, and each was equipped with an `OpenFlow` switch. In addition, each device was equipped with extra network interface cards (NICs) to support satellite, the third generation of wireless mobile telecommunications technology (3G), and fiber to the home (FTTH) technologies. Trema [5] was used as an `OpenFlow` controller, and the nodes used OvSs. The experiment was conducted to enable the link to be functional in case of a disaster, so the switching time was measured for all possible scenarios. The average switching time between the different technologies was 80 ms,

[5] trema.github.io/trema/

which is acceptable in the case of a disaster.

The use of mixed wireless networking technologies is presented in [5]. The authors' goal was to allow any wireless device to connect seamlessly to any network, irrespective of its technology, whether WiFi, WiMAX, or LTE. The authors propose the use of this architecture using SDN. The technical challenges are addressed, and other economic and regularity challenges are left for future researchers. The results are promising regarding a seamless handover from WiFi and WiMAX. The deployment consisted of two WiMAX base-stations, 30 WiFi access points, and five Ethernet switches (wired). The SDN/OpenFlow configuration consisted of NOX as a controller and used OpenWRT to provide `OpenFlow` switching functionality. To test multiple services in this virtualised environment, a FlowVisor was used to slice the network to allow the co-existence of multiple controllers.

A novel architecture for very dense heterogeneous wireless networks is proposed in [106] by making use of SDN technology. In this type of network, challenges such as control overhead and operational costs are the main concern. In this architecture, SDN is used to provide overlapping between LTE and WLAN, which are connected to the core network.

The advantages of abstracting the forward layer could benefit the case of mixed networks. This abstraction makes the communication between the control and the forward plane vendors and technology independent. Distributed controller architecture can be used to provide redundancy in the case of a failure.

3.6.2 Survivability In Wireless SDN

With the increased interest in Software Defined Networking (SDN), some studies have tried to wight the applicability of its application in WSN. The work of Kahjogh & Bern-

Ref.	Test	Technology	Setup	Controller	Goal
[98]	Prototype	WMN	OSLR	POX	Load balancing
[93]	Prototype	WMN	OSLR	NOX	Performance
[94]	Simulation	WMN	-	-	Performance
[100]	Prototype	WMN	B.A.T.M.A.N.	Floodlight	Load balancing
[96]	Prototype	WLAN	bootstrap	Kulcloud	Performance
[105]	Prototype	Mixed	Manual	Terma	Coverage
[97]	Prototype	WSN	ADOV	-	Performance
[5]	Prototype	Mixed	ipref	NOX	Coverage

Table 3.2 – Software Defined Networking in Wireless Network

stein [107] explored the utilisation of SDN controller to prolong network lifetime under optimal conditions. The authors provided a Mixed Integer Programming (MIP) with tow objectives: (i) To minimises traffic latency and (ii) To maximise network lifetime. The algorithm first evaluates network life and network lifetime with hop counts. Nodes with low energy reserve chose the path with least energy consumption, satisfying the second objective.

Alternatively, if energy is higher than certain threshold, a path with fewer number of hop counts is selected. To evaluate the efficiency of the proposed algorithm, the authors examined the performance of network under the aforementioned objectives separately and combination. The simulation showed 20% decrease in network lifetime due to enforcing hop count criteria. With SDN controller to manage events, balance between hop count and network lifetime can be amenable.

[108] presented tow approaches to solve the lifetime maximisation problem. A mathematical model that is solved using Column Generation and a greedy algorithm. Here coverage is used as a metric for lifetime calculations. The network is considered not functional if it drops under certain coverage level is reached. The paper discusses the applicability of each approach in various scenarios like; network size, flexibility of modifying the coverage criteria. In [109] a column generation approach is used to solve

the lifetime problem with mobile sink.

Energy minimisation of Software-Defined Sensor Network (SDSN), where nodes are equipped with different sensor types was studied in [110]. Each sensor node can activate specific sensing task dynamically based on the task required. In this work, minimum energy sensor activation problem is formulated as Mixed-Integer with Quadratic constrains Program (MIQP). With emphasis on quality of sensing , task-mapping and task-scheduling results are compared to a formulated online algorithm. Investigation results show reduced rescheduling time and control overhead.

A game theory approach was used in [111] to reduce reduce energy consumption in SDWSN. Compared to traditional energy algorithms, SDWSN approach provided balanced energy consumption and prolonged network lifetime.

The authors of [112] developed a cognitive SDWSN prototype for environmental applications. The design offered self-adaptability for environmental changes and reduced control signal overhead resulting in low energy consumption. N

SDN as a solution for reducing energy consumption have been studied in many recent papers. Ranging form Data Center network, Optical Network and Fixed networks. The work of [113] provides an extensive evaluation of the existing energy-centric SDN approaches.

3.6.3 Emulating Solutions for SDN and IoT

The involvement of hundreds of nodes in in wireless environments provides cheap and holistic solution, nevertheless it requires proper planning to ensure a smooth and efficient operation of the system. Network simulation is one way to conduct pre-feasibility study before the system is deployed. Three requirements in a simulator in the case of SDWSN needs to be fulfilled; controller compatibility, wireless functionality support and

extendibility. Once these requirements are available, the simulator can be used to debug and test protocols, evaluate performance and verify the scalability of the system.

One such simulator is developed Ramon et al. [114] for Software Defined Networking emulation, called Mininet- WiFi. In addition to Wi-Fi support, Mininet- WiFi also supports wireless sensor networks technology like 6LowPAN. The goal is enhancing Mininet emulator which lacks the support of wireless channel or mobility modelling. The addition of these features makes a clean extension of the highly reliable Mininet emulator. This is due to supplementation of new classes and abstractions to support emulated links and wireless NICs. Besides, it has the potential of permitting the physical wireless and wired interfaces to integrate with the virtual environment. Thus, utilizing challenging tasks like integrating mobility with SDN environment through single or multiple controllers. An added benefit of using Mininet-WiFi is it's convenient for simulating the handover procedure in a wireless network.

Apart from the initial study, Mininet-WiFi was also implemented in subsequent studies. The work in [115] used a single controller of Mininet-WiFI to run the Software Defined Wired and Wireless Network(SDWWN). It was selected for executing the SDWWN because of its potential in wireless stations and virtualised access points. In this study it was used to emulate wireless services with Ethernet connections.

The work in [116] offers a first use case for security application testing based on Mininet-WiFi. The study addresses a new class of threats to traditional and SDN networks through unauthorised connected devices.

The authors in [117] designed a prototype of a monitoring platform to gather real-time data about the services in the community network"CN" [6] using a gossip-enabled network. The utilisation of Mininet-WiFi enabled an edge-cloud computing environment

[6]CNs are large-scale, self-organised and decentralized communication infrastructures built and operated by the community itself

Reference	Area	Purpose	Controller
[115]	SDWWN	Coverage	NOX
[116]	Campus	Security	ryu
[117]	Community Network	Monitoring	-
[118]	IoT	Security	Floodlight
[119]	SDWN	Association	OpenDayLight
[120]	SDWN	Association	-
[121]	SDWN	Association	OpenDayLight
[122]	VANET	Performance	POX
[123]	IoT	Security	-

Table 3.3 – Summary of Mininet-WiFi Related Studies

and revealed the data dissemination through gossip-enabled network is achieved within minutes.

A study of using Mininet-WiFi on IoT security is conducted in [118]. The study is focused on edge oriented detection and mitigation scheme against DDoS in IoT using SDN.

[119] [120]and [121] studied the association control [7] using Mininet-WiFi as the emulation tool. In [119], Mininet-WiFi is used in research which focused on evaluating the performances of handover association mechanisms in SDN-based wireless networks. This study is pursuing a solution of the simplified network operation and management with taking advantage of Software Defined Networking (SDN), OpenDayLight was used as a controller.

Tarigan et al. [120] designed a dynamic load balancing system that lowers the congested APs load by forcing users located near the boundaries of loaded AP to move to a less-loaded neighbouring AP. The system aims to maximize throughput by moving users to lightly loaded APs and allowing each AP to provide end users with maximum data rate.

[7] Association control is a mechanism that regulates the association between stations and access points in the network

Reference	Area	Purpose	Controller
[124]	SDWN	Handover	POX
[125]	Enterprise	Edge Virtualisation	-
[126]	VANET	GeoBroadcast	Floodlight
[127]	SDWN	Flow Stability	POX

Table 3.4 – Summary of OpenNet Based Studies

Moreover, Mininet-WiFi is used for simulating Vehicular Ad-Hoc Network (VANET) by Indriyanto et al. [122]. In this study, Mininet-WiFi is installed on Ubuntu that ran on a virtual machine. Two Ubuntu servers were installed on VMware: one for running the Mininet Wi-Fi and the other for running the POX [39] controller. The study evaluates performance parameters such as packet drop, delay, and throughput.

[123] evaluated the performance of Mininet-WiFi along with other non-SDN tools. The study evaluated the association and authentication of different wireless emulation tools. The study concluded with an extension module to enable a wireless emulation experiment to span Mininet-WiFi instances on different computer platforms.

Another simulator is OpenNet [124] , that can be used to simulate SDWSN. OpenNet brings the world of SDN through use of Mininet to the world of wireless environment by utilising NS-3. The combination of the two simulation environments, enabled researchers to test and evaluate different scenarios under SDWSN environment [125] [126] [127] [128].

[125] examined new edge virtualisation architecture in WLAN enterprise using SDN approach. OpenNet provided the ability to simulate featuring centralised logical control and provides mobility to clients.

OpenNet running Floodlight controller was used to automatically arrange roadside units (RSUs) in VANET in geographical positions in [126]. OpenNet was used also to evaluate the routing [127], handover management [129] and other applications [128].

Chapter 4 | Lifetime Maximisation of SDWSN

4.0.1 Lifetime of a Wireless Sensor Networks

A network can only fulfil its purpose as long as it is alive. What decides this lifelessness is the choice of proper analysis. Thus, it is imperative to choose the right metric for such analysis so that it suites the intended WSN application. Connectivity, coverage and node availability broadly define the purpose of WSN application.

The later, evaluates a single node lifetime then creates a criteria of evaluating the whole network based on that. A trivial but few practical real life implementation is the last dead node lifetime. That is once all network nodes deplete energy the network is considered dead. Another commonly used approach is defining the lifetime as the time for the first node to die. This plays key role in mission critical applications where all nodes have the same importance. Another variant of the node availability criteria is the time until a fraction of network nodes α are still a live. This definition suits most applications[130].

The lifetime of a single node is determined by the available energy and the energy consumed to perform its operations. Typically a node will consume energy in sensing, communicating and data processing. with the communication part consuming most energy [22].

With a centralised controller like the case of SDWSN, however, connectivity is more applicable. For example, the number of nodes that are able to communicate with the central controller. The node availability criteria is not sufficent in this scenario if the nodes are available and cannot communicate with central controller.

4.1 State of the Art

Many studies address lifetime maximisation of WSN [108] [109]. While there exists some decent amount of work of wired network under SDN environment [113]. There is a scarce in the knowledge base of lifetime in Software Defined Wireless Sensor Network due to the emerging nature of the research field. Below is a highlight of some notable and related research articles starting from solutions in traditional WSN and SDWSN.

4.1.1 Traditional WSN

The appropriate definition of lifetime of WSN that takes into consideration the lack of constant energy nature is discussed in [131] and [132]. [131] provides a definition of *operational lifetime* of sensor nodes to replace the common definition of lifetime of WSN network based on the last node to deplete. It provides a mathematical perspective of the problem, and provides a prove that energy depletion decreases in the order of 1/n of the initially deployed nodes "n". [132] provides an alternative WSN network lifetime definition, *functional lifetime*. Where the lifetime of the network and the amount of data collected depends on (a) the layout of the sensor network, (b) the initial battery capacity on the individual sensor nodes, (c) the characteristics of the sensor data generated at the individual nodes, and (d) the communication costs in transferring.

[108] presented tow approaches to solve the lifetime maximisation problem. A mathematical model that is solved using Column Generation and a greedy algorithm. Here coverage is used as a metric for lifetime calculations. The network is considered

not functional if it drops under certain coverage level is reached. The paper discusses the applicability of each approach in various scenarios like; network size, flexibility of modifying the coverage criteria. In [109] a column generation approach is used to solve the lifetime problem with mobile sink.

4.1.2 Software Defined Wireless Sensor Network

With the increased interest in Software Defined Networking (SDN), other papers have tried to wight the applicability of its application in WSN. The work of Kahjogh & Bernstein [107] explored the utilisation of SDN controller to prolong network lifetime under optimal conditions. The authors provided a Mixed Integer Programming (MIP) with tow objectives: (i) To minimises traffic latency and (ii) To maximise network lifetime. The algorithm first evaluates network life and network lifetime with hop counts. Nodes with low energy reserve chose the path with least energy consumption, satisfying the second objective.

Alternatively, if energy is higher than certain threshold, a path with fewer number of hop counts is selected. To evaluate the efficiency of the proposed algorithm, the authors examined the performance of network under the aforementioned objectives separately and combination. The simulation showed 20% decrease in network lifetime due to enforcing hop count criteria. With SDN controller to manage events, balance between hop count and network lifetime can be amenable.

Energy minimisation of Software-Defined Sensor Network (SDSN), where nodes are equipped with different sensor types was studied in [110]. Each sensor node can activate specific sensing task dynamically based on the task required. In this work, minimum energy sensor activation problem is formulated as Mixed-Integer with Quadratic constrains Program (MIQP). With emphasis on quality of sensing , task-mapping and task-scheduling results are compared to a formulated online algorithm. Investigation

results show reduced rescheduling time and control overhead.

A game theory approach was used in [111] to reduce reduce energy consumption in SDWSN. Compared to traditional energy algorithms, SDWSN approach provided balanced energy consumption and prolonged network lifetime.

The authors of [112] developed a cognitive SDWSN prototype for environmental applications. The design offered self-adaptability for environmental changes and reduced control signal overhead resulting in low energy consumption. Network lifetime was prolonged by more than 50 rounds due to the use of this technique.

4.2 Problem Formulation

A wireless network is represented by a graph $\mathcal{G}(\mathcal{N},\mathcal{L})$. Where the vertices $\mathcal{N}$ of the graph correspond to network nodes. The links between network nodes correspond to the edges/arcs of the graph. There is a bidirectional link $\mathcal{L}$, between a pair of two nodes (i,j) if they are within the transmission range of each other. The set of nodes S_i, contains nodes that fall in the communication range of i. We denote a node that is equipped with SDN controller with $\mathcal{X}$, this node acts as a base station and infinite power supply is assumed. The initial energy of node i is E_i ($E_i > 0, \forall i \in \mathcal{N}$), and the amount of data to be transmitted through link (i,j) is f_{ij}. Other variables are summarised in 4.1.

Lifetime of a node T_i can be expressed as the total amount of energy consumed for sending transmitting and receiving data

$$T_i = \frac{E_i}{e_i \sum\limits_{(i,j)\in\mathcal{L}} f_{ij}} \tag{4.1}$$

In addition, under a flow f_{ij} the lifetime of the network T can be expressed as

the time needed for the first node in the system to deplete its initial energy first.

$$T = \min_{i \in \mathcal{N}} T_i = \min \frac{E_i}{e_i \sum_{i \in \mathcal{L}} f_{ij}} \tag{4.2}$$

Table 4.1 – Variables Used In The Optimisation Model

Notation	Meaning
$\mathcal{G}(\mathcal{N},\mathcal{L})$	*directed graph representing network topology*
$\mathcal{N}$	*set of network nodes*
$\mathcal{L}$	*set of links*
S_i	*set of nodes that can communicate with i*
$\mathcal{X}$	*SDN controller node*
T	*lifetime of the network*
T_i	*lifetime of node i*
r_i	*sensed data by node i*
E_i	*initial energy of node i or battery capacity*
e_i	*energy spent for sending* f_{ij} *by i*
f_{ij}	*amount of flow to be sent through link (i,j)*

Therefore, in order to maximise the lifetime of the network. Routing algorithm that enables the most vulnerable node to depletion must be implemented.

$$\begin{aligned} Maximize\ T &= Max\,(\min T_i) \\ &= Max\left(\min \frac{E_i}{e_i \sum_{i \in \mathcal{L}} f_{ij}}\right) \end{aligned} \tag{4.3}$$

The nonlinear term in (4.3) , can be linearised following the approach of g. The variable $\overline{f}_{ij}$ is introduced to reflect that over time the assigned flow f_{ij} becomes a rate. Hence

(4.3) becomes the following linear program for maximising the lifetime of SDWSN.

$$Max\ T \tag{4.4}$$

Subject to the following constrains:

$$s.t. \begin{cases} \overline{f}_{ij} \geq 0 & , \forall\ (i,j) \in \mathcal{L} \quad (4.4) \\ e_i \sum\limits_{i \neq j} \overline{f}_{ij} \leq E_i & , \forall\ i \in \mathcal{N} - \{\mathcal{X}\} \quad (4.5) \\ \sum\limits_{k \in S_i} \overline{f}_{ki} + r_i \cdot T = \sum\limits_{j \in S_i} \overline{f}_{ij} & , \forall\ i \in \mathcal{N} - \{\mathcal{X}\} \quad (4.6) \\ \sum\limits_{k \in S_i} \overline{f}_{ki} + \sum\limits_{j \in S_i} \overline{f}_{ij} \leq T & , \forall\ i \in \mathcal{N} - \{\mathcal{X}\} \quad (4.7) \end{cases}$$

The constrain in (4.5) reflects the energy consistency, where only nodes with sufficient energy are selected. Constrain (4.7) is the usual flow constrain, the sum of the traffic entering to a node in addition to the traffic generated by the node r_i equals to the traffic leaving that node. The last constrain ensure concurrent flow.

4.2.1 Illustrative Example

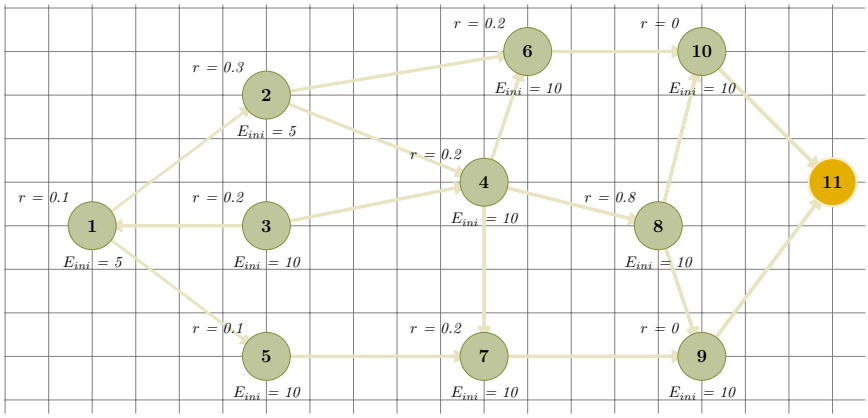

Figure 4.1 – Network Topology of a typical SDWSN

The following network topology represents an SDWSN with 11 nodes, the nature of the topology, and the application in which the network dramatically affects any

:	1	2	3	4	5	6	7	8	9	10	11
1	0	0.1	0	0	0.2	0	0	0	0	0	0
2	0	0	0	0	0	0.4	0	0	0	0	0
3	0.2	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0.1	0.1	0	0	0
5	0	0	0	0	0	0	0.3	0	0	0	0
6	0	0	0	0	0	0	0	0	0.6	0	0
7	0	0	0	0	0	0	0	0	0	0.6	0
8	0	0	0	0	0	0	0	0	0.9	0	0
9	0	0	0	0	0	0	0	0	0	0	1.5
10	0	0	0	0	0	0	0	0	0	0	0.6
11	0	0	0	0	0	0	0	0	0	0	0

Table 4.2 – Flow from sensing nodes to SDN controller

optimisation algorithm.

Node 11 in Figure 4.2 is a base station equipped with an SDN controller. Other nodes are sensing or forwarding the data. The arrow indicates the traffic direction, other parameters are depicted in the figure. This scenario was used for testing the optimisation model (4.4) using AMPL.

The initial energy of node 1 and 2 equals 5, the rest of the nodes have an initial energy of 10. The generated data of each node is represented by r. For example, node 3 generates 0.3 data frames.

AMPL [133] is a reliable algebraic modelling platform for optimisation problems. The intuitive syntax of AMPL allows many mathematical optimisation problems to translated to a model file ".mod" with lucid syntax. With sufficient data ".dat" supplied by the user to support the model, the objective function, primal and dual variables at the solution point can be straightforwardly displayed. AMPL relies on existing off-the-shelf solvers, like CPLEX or MINOS, to perform the pre-solve functionality of the linear program. The network lifetime was 11.11 time units.

The results in Table 4.2 are of the MILP inside the AMPL/Cplex model of the mathematical model. Flow from the sensing nodes to the SDN controller are the non zero values of table 4.2. The link between node 3 and 1 carries 0.2 data frames, and the link between node 2 and node 6 carries 0.4 data frames. These routing decisions are the result of the mathematical model, which optimises the network lifetime.

One drawback of the mathematical optimisation model is that it traces all possible solutions to find the maximum lifetime for the network. This leads to the need of advanced processing capabilities, which does not exist in WSNs. Hence, a heuristic algorithm is a possible alternative solution in this situation.

4.3 Proposed Algorithm

4.3.1 Assumptions

To make the model more generic, we assume that all SDWSN nodes have the same importance and are all equipped with the same computational and energy resources. The topology is assumed to be flat, that is, there are no superior nodes, cluster head or hierarchical structure. Because it is a characteristic of the absolute majority of standard existing nodes, we assume that all SDWSN nodes are communicating with fixed communication power and lack the ability to adjust their transmission power. We also assume that they are immobile once they are deployed.

The SDN controller, on the other hand, is assumed to have unlimited power supply that is constantly charge with reliable energy source. The controller is responsible of forming the topology and managing the recovery process by selecting the best candidate in the case of failure. The SDN controller is placed in the centre of the deployment field. The communication form the controller follows In-Band mode, where nodes forward control messages from neighbours and a direct connection to with the SDN controller is

not required as long as a neighbouring node has an OpenFlow session running with the controller. In the in-band mode, switches do not need an extra physical port for control traffic. OpenFlow defines a virtual port in the SDWSN switch called local port, which enables remote entities (e.g. controller) to interact with the switch via an OpenFlow network (in-band mode)[53].

Tow end nodes are considered connected if they can communicate with each other, send/receive messages to/from each other. So, the link between the nodes are assumed to be bi-directional. It is also assumed that the location information is known during the initialisation process.

4.3.2 A-star Algorithm

A-star is a path finding algorithm that tries to reduce the total number of states explored and is successfully deployed AI applications. A-star is guaranteed to find to find the optimal path by incorporating a heuristic estimate of the cost to get to the destination node from a given starting node.

A-star maintains tow lists, OPEN list and CLOSED list. Nodes that need to be examined are kept in the OPEN list, while the CLOSED contains nodes that have already been examined. It is initialized by the OPEN list containing the start node, and an empty CLOSED set. Every node is evaluated based on cost plus heuristic function $f(n)$. Firstly, the cost of getting from the initial node to n is stored in $g(n)$. In our implementation this corresponds to the ratio of initial energy over the residual energy $g(n) = E_{ini}/E_{res}$. Along a sequence of nodes, $g(n)$ becomes the sum of the previous costs. In addition, the shortest distance for getting from current node n to the SDN controller is $h(n)$, which uses the Euclidian formula:

$$d(i,j) = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{4.8}$$

setting the second function $h(j)$ to zero, simplifies A-star functionality to Dijskrats Algorithm.

Moreover, the evaluation function $f(n)$ which combines the tow cost functions; $f(n) = g(n) + h(n)$, is used to evaluate the best route passing through node n. The value of $f(n)$ is maintained in OPEN priority queue list.

A-star has a main loop that gets the node n with the lowest $f(n)$ from this list. If the destination is not found, then it examines the successors of n by placing them in the priority queue. The loop terminates once the destination is reached or the priority queue is empty.

After collecting the fundamental information, the controller applies the following energy model as a secondary function for $h(j)$

4.3.3 Energy Model

The total energy $E(d)$ for transmitting data from node i to another node in the network at distance d follows the following formula [22] :

$$E(d) = E_{TX} + E_{RX} = \max(E_{min}, \beta d^{\alpha}) + E_{RX} \tag{4.9}$$

To maintain reliable communication, the transmission energy increases with distance increase. Here β is the power required to communicate for one meter distance, $\alpha \geq 2$ is the loss factor due to propagation. However, if no transmission takes place, a minimum energy is assumed to always be radiated E_{min} even in the idle case.
In a square region, where WSN usually deployed, the analysis in [22] produced the fol-

Algorithm 1: Lifetime Extension for SDWSN

```
enqueue (OPEN,start)
while OPEN ≠ ∅ do
    if d(n,n′) ≤ d_trans then
        κ_n ← n′
    end
    dequeue ( OPEN, n_best)
    enqueue (CLOSED , n_best)
    if n_best = 𝒳 then
        foreach n′ ∈ κ_n and n′ ∉ CLOSED do
            successor(n′) ← n
            h(n′) ← √((x_n′ − x_n)² + (y_n′ − y_n)²)
            g(n′) ← E_ini/E_res
            f(n′) ← h(n′) + g(n′)
            if n′ ∉ OPEN then
                enqueue (OPEN , n′)
            else
                successor(n′) ← n_best
            end
        end
    else
        foreach successor(n′) do
            update FlowTable of n′
        end
    end
end
```

lowing average power consumption per node i.

$$E_i^{AVG} = \tau\,\eta^*\,r_i\;d_{(i,c)}\left(\frac{1}{3\sqrt{2}}+\frac{1}{12}\log\left(\frac{\sqrt{2}+1}{\sqrt{2}-1}\right)\right) \tag{4.10}$$

where τ represents the time for transmitting a single bit of data, r_i is the data generated from the node. The model [22] incorporates the location of node as a parameter that is $r_i(x,y)$, where (x,y) the location of the sensor node. The distance from node i to SDN controller is denoted by $d_{(i,c)}$, and η^* is the optimal efficiency watt-per-meter

$\eta^* = \frac{E(d)}{d^*}$ and d^* is given by

$$d^* = \begin{cases} \left(\frac{E_{min}}{\beta}\right)^{1/\alpha} & \text{when } \frac{E_{RX}}{\alpha-1} < E_{min} \\ \left(\frac{E_{RX}}{\beta(\alpha-1)}\right)^{1/\alpha} & \text{when} E_{RX} > 0, \frac{E_{RX}}{\alpha-1} \geq E_{min} \end{cases}$$

For each node that passed the setup phase with no flow entries, the controller uses A* pathfinding algorithm. Utilising formula (4.10) as $h(j)$ and Euclidean formula (4.8) to the function $g(j)$ sets the A* Algorithm path finding criteria that maximises network lifetime.

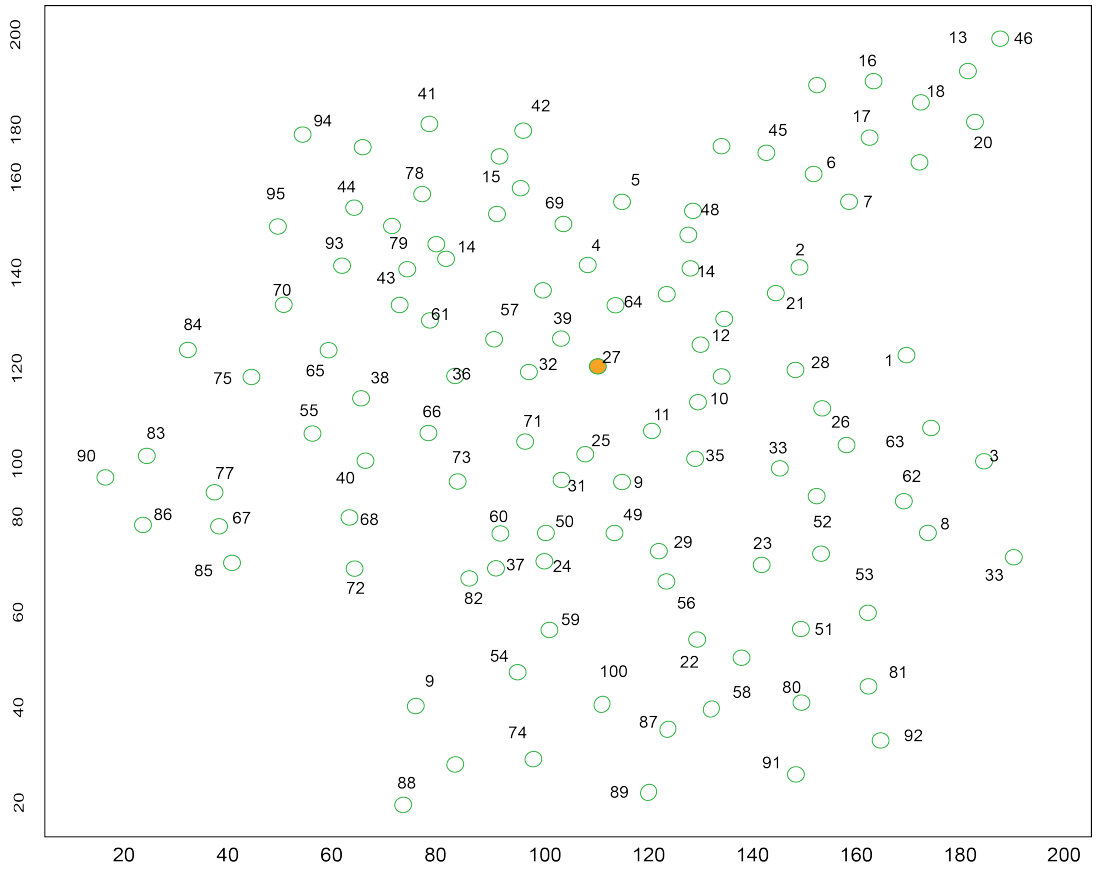

Figure 4.2 – 100 Random Nodes Network

4.4 Simulation

The algorithm was tested in a network of 100 randomly generated, with nodes equipped with the same amount of energy of 5J. The SDN controller was selected to be in the middle of the network, and the traffic generated by each sensor node was four packets per round, with each round a random sensor is selected. The distance between the nodes is set to a minimum 20m, with simulation area of 200m x 200m. Table 4.3 summarises the rest of the simulation parameters.

4.4.1 LEACH protocol

The simulation is compared initially with a protocol called low-energy adaptive clustering hierarchy protocol(LEACH). LEACH protocol puts the multi-hop nature of WSNs into service. The LEACH protocol divides the network into clusters of a hierarchal structure. Sensor nodes are grouped into clusters, and one sensor node is nominated as the cluster head of each cluster. This arrangement is performed to increase energy efficiency because only cluster heads will communicate with the base station. Data is gathered from all the sensor nodes within the cluster, then forwarded to the cluster head. Data is then compressed before being sent on to the base station [17]. A distinctive character of LEACH compared to other algorithms is the ability to self-organise and adapt to the change of nodes' energy. The process of selecting a new cluster head is performed each round. Here, a randomised algorithm is used to decide which node becomes the next cluster head [18]. The underlying assumption is that all nodes are powerful and capable of reaching the base station, but energy is saved by restricting the communication of all nodes except one.

Communication with cluster head uses Time Division Multiple Access(TDMA); where sensor nodes transmit sensed data consecutively. In TDMA, the cluster head node controls the time limit and order in which nodes can transmit. This channel access

technique limits the radio usage of nodes as they restricted from accessing open radio link aside from their designated time slot. LEACH protocol also minimises interference between clusters by allocating a different code for each participating node. Through the utilisation of a conventional communication technique known as code division multiple access (CDMA), which allows multiple sensor nodes to use the same wireless channel at the same time.

The operation of LEACH can be split into a setup phase and steady phase. Setup phase involves the formation of a cluster of nodes and specifying a cluster head. Cluster head node is revealed each round through an election process. The decision to selecting a cluster head is available for all participating nodes and is based on specific probability parameters. The algorithm produces a threshold value, and then participating node selects an arbitrary random value between zero and one. If the threshold value is higher than the chosen arbitrary value, the node becomes a cluster head. Rate of cluster heads of the network determines the threshold value along with the number of times the node has previously become cluster head. After that, cluster head nodes notify other nodes by broadcasting a signal to all the other nodes. Upon receiving this broadcast signal, a regular node forms a cluster with the cluster head having the strongest signal. The reason for this selection is that it will ensure the node will be consuming less amount of energy in transmission to a closer and strong cluster head.

The last process in the setup phase is scheduling. Cluster head designates time slots to all the nodes in its cluster, provided that these nodes become only active during their timeslots.

After the scheduling is completed, the LEACH protocol enters the Steady Phase. During this phase, data is forwarded to the cluster head form all the other nodes in the cluster as per their allocated time slots. Finally, the cluster head then gathers all the data received and conveys it to the base station [19, 20].

Parameter	Value	Unit
Network Area	200x200	m*m
Number of rounds	500	round
Number of sensor nodes	100	nodes
Transmission radio range	80	m
Initial energy	5	J
Number of transmission packets	4	packet/round
Message size	500	bit
α	2	
E_{RX}	10^{-4}	J
E_{min}	10^{-5}	J
β	10^{-8}	J/m/b

Table 4.3 – Simulation Paramters

4.5 Results and Discussion

The main focus in terms of results analysis is focused in directed to how the introduction of SDN technology in Wireless Sensor Networks can improve the network lifetime. Our analysis indicate that SDN improved network lifetime by 16% compared to traditional LEACH. This is due to the fact that SDN technology requires less number of control messages and fewer processing power is consumed due to the fact that route calculations are performed in the SDN controller rather than in sensor nodes in traditional routing. With the different network lifetime definitions mentioned earlier, the observations as follow. The first node to deplete energy in SDWSN is very fast in the SDWSN compared to traditional routing. Thus the applications like intrusion detection might not benefit from the scenario in this work. Alternatively fewer number of nodes could be used for similar applications.

However, considering network lifetime definition as 50% of network nodes to deplete, the new algorithm offers promising potential. Here, network nodes had gained an impressive 22.6% the accumulate to the network lifetime. From figure 4.3, the SDN approach controller tries to stabilise the network at the beginning of the first nodes to

die, this is due to the use of A-star algorithm in finding much stable paths. When 30 nodes are dead, the SDN controller provides a graceful network outage and energy consumption drop. Before that point, nodes that exhausted are the ones with more number of connections to neighbouring nodes and much closer to the SDN controller. It can be noticed that the at some parts, nodes are still having enough energy but not feasible route to the controller. This can be addressed by the deployment of multiple controllers.

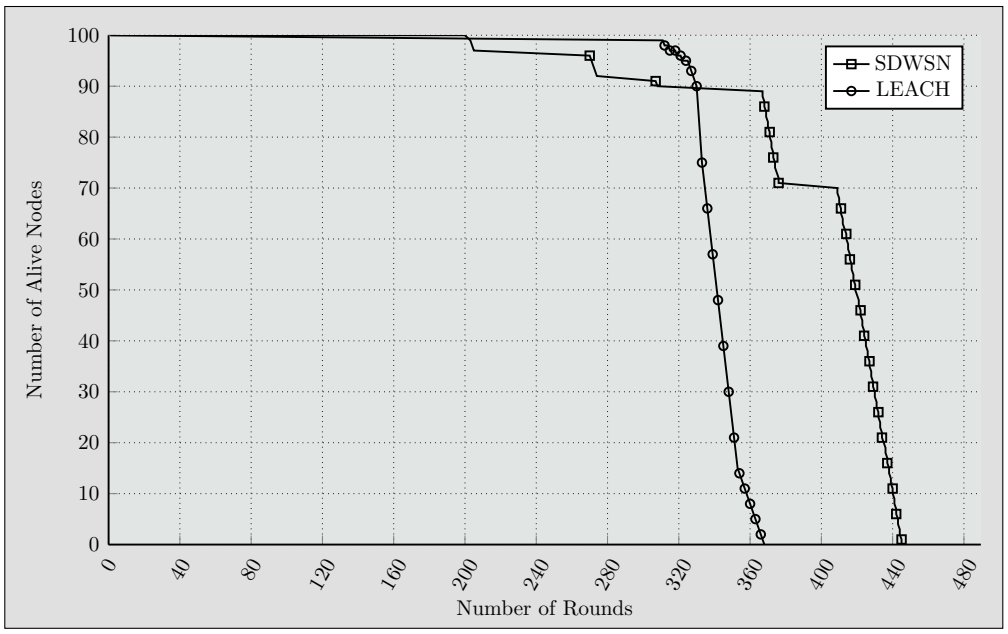

Figure 4.3 – A-star based SDWSN lifetime compared to LEACH with random network consisting of 100 nodes

The distribution of energy in LEACH protocol, is based on selecting cluster head with pre-assigned probability. In our simulation the percentage of nodes with advanced power is 50% and probability of 20%. This results in distributing energy load among network clusters. However, this also causes more control traffic between sensing node and cluster head and the addition of traffic aggregation. Figure 4.4 shows the effects of this approach compared to our proposed solution.

Initially the SDN controller collects network topology causes more energy con-

Characteristic	Value
Number of routes generated by the algorithm	436 route
Average number of hops in each route	5
Average energy consumed by route	1.797 J
Route with maximum energy consumption	3.856 J
average energy dissipation per node	0.363 J

Table 4.4 – Results overview

sumption. After that each node has `FlowTable` with destination stored and the only overhead remaining is to forward traffic. Wherever a node is depleted, `FlowTabless` are updated accordingly following the In-Band mode process through the forwarding plane. In contrast to traditional routing, this does not cause sharp increase in energy consumption with nodes performing the computations and and maintaining the route.

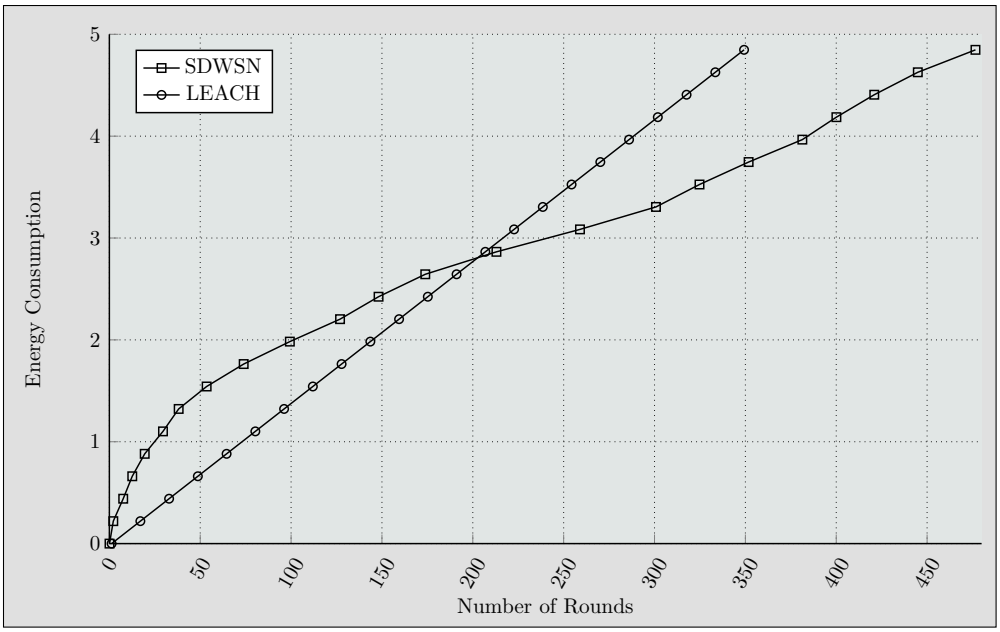

Figure 4.4 – Energy dissipation of SDWSN compared to LEACH with random network consisting of 100 nodes

4.6 Conclusion

Network programability provide a promising future for solving many network problems inherited from traditional networking. In this work SDN controller is incorporated with WSN to manage and oversea the network. The In-Band mode enables wireless nodes to communicate with the controller through their neighbouring nodes once a flow entry requires modification. Other than this case, the forwarding "data" plane is used for its original forwarding function.

This approach might not suite all Internet of the Things (IoT) applications, however a vast majority of applications will gain vast improvement in terms of energy and centralised control. In particular, applications where the number off nodes and their deployment area are on the scale of hounders of meters. In theory, the approach can be extended to include more than one SDN controller, which will further be investigated in future work.

Simulation results suggest that A-star based routing within SDWSN can increase network lifetime by using residual energy of nodes as the heuristic part of the algorithm. Compared to traditional routing, this approach increased network lifetime by 22.6% in randomly generated network . By carefully incorporating advanced nodes or within grid network, our results showed this figure can increase dramatically.

This centralised routing scheme provides a graceful network outage and energy consumption drop. Due to the balancing of power use in the network, wireless sensor nodes deplete their energy sources at approximately the same time. This is highly beneficial since all nodes can be recharged or replaced simultaneously, instead of of constantly monitoring and servicing individual devices.

Chapter 5 | Failure Recovery in SDWSN

5.1 Overview

Internet of the Things (IoT) is the anticipated new wave that will enable everyday things to be connected to the internet and change how we live and work. Once implemented IoT will bring innovative industries and services to light. Since IoT is merely based on Wireless Sensor Networks (WSN), existing limitations in this type of networks have to be resolved. Major drawbacks in WSN like rapid energy depletion and wireless unreliability pose major challenge for WSN. A new trend in networking that could offer a remedy to some of these dilemmas is virtualization and in particular Software Defined Networking (SDN). In SDN environment, computational functionalities of the network nodes are moved to a central controller. Consequently, network nodes are left with minimal required functionality providing them with more power resources to utilize for their fundamental data forwarding functionality. This new technology implementation proven to be a success in wired networks, both in saving resources (human and machine) in addition to delivering cross vendor operability and many other benefits.

Inspired by its success in wired environments, many researchers investigated the feasibility of incorporating wireless devices in SDN environment [87] [88]. Numerous encouraging results in WMN have been obtained both in Simulation [89] and test bed environments [90]. Additionally, other studies have considered the practicality of integrating Wireless Sensor Networks, WLAN and cellular networks [88].

In all of the above wireless implementations, the environments work infrastructure-based fashion. That's due to the existing limitations of SDN of only supporting Ethernet type connections. However, the faultiness of such devices, more precisely the behaviour of the node whilst isolated, haven't been considered reasonably. Survivability is underplaying character of wireless networks, which is not always caused by node faultiness rather than the nature of the wireless environment. Nonetheless, wireless sensor nodes usually characterised by the deficit of long lasting energy source as well as unreliable communication channels, which usually adds to the severity of service disruption and instability.

This part of the research investigates practical solutions that address the disruption-prone nature of wireless nodes when migrated into SDN, i.e. SDWSN. While faultiness is a broad area for investigation in wireless environment, we only consider the factors that probably cause disruption the whole system. In particular, our goal is to provide the means for wireless nodes to act independently (1) in case of an SDN controller absence or routing protocol failure (2) path or link protection compared to restoration in the case of node failure, (3) in addition to providing selecting appropriate beaconing mechanism among the nodes in these situations. The research will help shaping critical design and implementation considerations of disruption tolerant Wireless Software Defined Networks.

5.2 Failure Detection in SDWSN

Data traffic and control traffic can be affected by by failure in the data plan, that is node or link failure. This failure results in (1) preventing new services to established due to loss of control signal; and (2) disruption to the services due to the loss of data traffic. There are two primary error detection techniques that can be employed in SDWSN to identify failure. Loss-Of-Signal (LOS) is a technique for detecting failure in any forwarding port and is usually used in reactive recovery techniques "Restoration". On the other hand, for detecting failure in paths Bidirectional Forwarding Detection [134].

In the case of Loss-of-Signal (LOS), a node that operates under OpenFlow depends on the `PORT_STATSUS` flag to detect port failure. In the event that a port is not functioning, "`port down`" event notifies the controller of the failure. Compared to BFD, this technique notifies the controller about only one end and utilises port failure and the node still has a connection to the to the controller. In addition, this technique depends on failure declaration instead of echo timeout signal.

Bidirectional Forwarding Detection [134] is lightweight protocol, that operated in the data plane and does not require communication to the SDN controller, i.e., the control plane. According to the BFD specifications [134], there are two modes of operation; Asynchronous and Demand modes. The earlier requires periodical exchange of messages at a fixed rate. While the later suits other applications where one end node can request the corresponding end node to activate or deactivate the transmission control messages at any time. Further Asynchronous mode can be Echo or Non-Echo.

BFD implements an echo and control message system to identify the availability of the link between two nodes. BFD session is initiated as most protocols with handshake process after that each node exchanges control message with the other end of the link to indicate the liveness of the link between them. Once the session established, the tow nodes exchange session status messages. BFD **control messages** are required when the operating in Non-Echo mode. Alternatively, if the Echo mode is used, as the name suggests echo messages are used instead of the control messages to check the status of the other interface.

In BFD, failure detection time T_d depends on two factors: (1) the detection time multiplier M and (2) transmit interval T_i. The earlier is the number of lost control messages before the failure is detected by the end node, and the latter represents how

frequent the messages are exchanged. To prevent incorrect status report, traditionally M is assigned a value of 3. The transmit interval T_i,on the other hand, is less than or equal to the Round Trip Time (RTT) of the link. Moreover, a 25% time jitter is usually introduced to accommodate for packet synchronization. Hence, the minimum BFD transmission time is:

$$T_i \text{ min} = 1.25 \cdot \text{RTT}$$

The early versions of OpenFlow specifications didn't address the failure problem. However, since OpenFlow v 1.1 Fast Failover Group Table was introduced [53]. OpenFlow adopts BFD, however it was partially implemented inside Open vSwitch where only Asynchronous mode can been used.For BFD to work for SDWSN a local port has to be reserved within SDWSN and controller has to initiate the BFD process. With the controller sending BFD message, SDWSN node can run BFD session in its local port.

These BFD process takes place after OpenFlow bootstrapping process, OFPT_HELLO , OFPT_FEATURE_REQUEST and OFPT_FEATURE_REPLY. In the case that BFD detects failure, the ingress node declares the working path is faulty and recovery process is initiated based on OpenFlow rules.

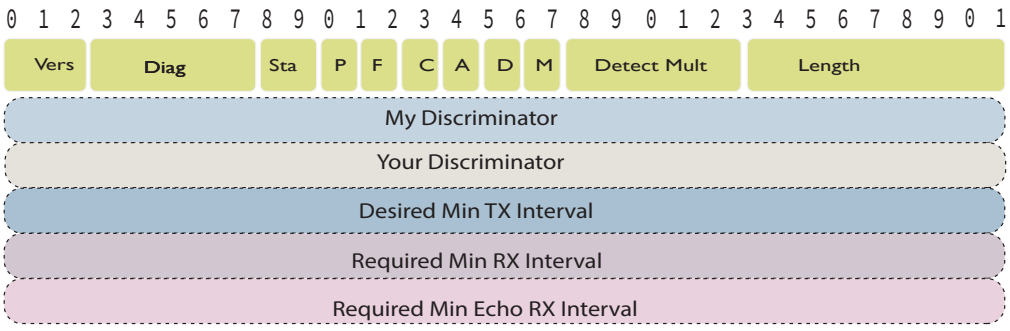

Figure 5.1 – BFD Control Frame

The BFD control packet consists of two parts: (1) a mandatory part and (2) an

optional authentication field. The optional field have different field formats depending on the authentication type.

Figure 5.1 shows different fields of BFD control packet. It contains certain flags such as "A" which indicates the BFD session requires authentication. The "P"and "F" flags are used for polling information independent of the other end and can't be set to 1 at the same time. The "D" flag is used when the node wishes to operate in in Demand mode instead of the default Asynchronous mode. `My Discriminator` field is 32 bits identifier sent to the other end or "0" if it doesn't know the value. The other fields can be negotiated between the two BFD parties based on their capabilities. The `Required Min Rx Interval` field serve in indicating the minimum interval for receiving BFD packets in microseconds to force the other BFD party to the required time. `Desired Min Tx Interval` informs the other end about the desired transmission rate.

The other type of messages, Echo, can be implemented differently. A common approach for this simple type of messages is to include a node id and timestamp in addition to other possible implementations[134].

5.3 Failure Recovery Techniques

A common drawback in wireless networks more than wired networks, is network failure[135][136] . Once it takes place, a network recovery technique is required. One mechanism to recover a lost connection before it happens is called protection. If the connection is restored after failure takes place it is called restoration. [137]

An important related feature in resource recovery design is whether it is link or path recovery. For the earlier, only the failed link is recovered. While in the latter, the end-to-end flow that depends on a failed link is re-established.

When a failure takes place, a choice of link or path reestablishment is an impor-

tant decision for a resilient network design. In the case of link recovery, only the failure from one node to the other end node is considered and resolved. In the other hand, path reestablishment takes in consideration the full end to end path and recovery process includes all failed nodes.

Figure 5.2 illustrates the difference between path recovery and link recovery mechanism. For demand $\langle 1,11 \rangle$ the flow is established on path $1 \rightarrow 2 \rightarrow 4 \rightarrow 8 \rightarrow 10 \rightarrow 11$. If the link 4-8 fails, then with link recovery scheme, the traffic will be routed around the failed link 4-8, e.g., the path 4-6-10. With path recovery scheme, however, a new end to end path for the demand $\langle 1,11 \rangle$ is established along the path $1 \rightarrow 5 \rightarrow 7 \rightarrow 9 \rightarrow 11$.

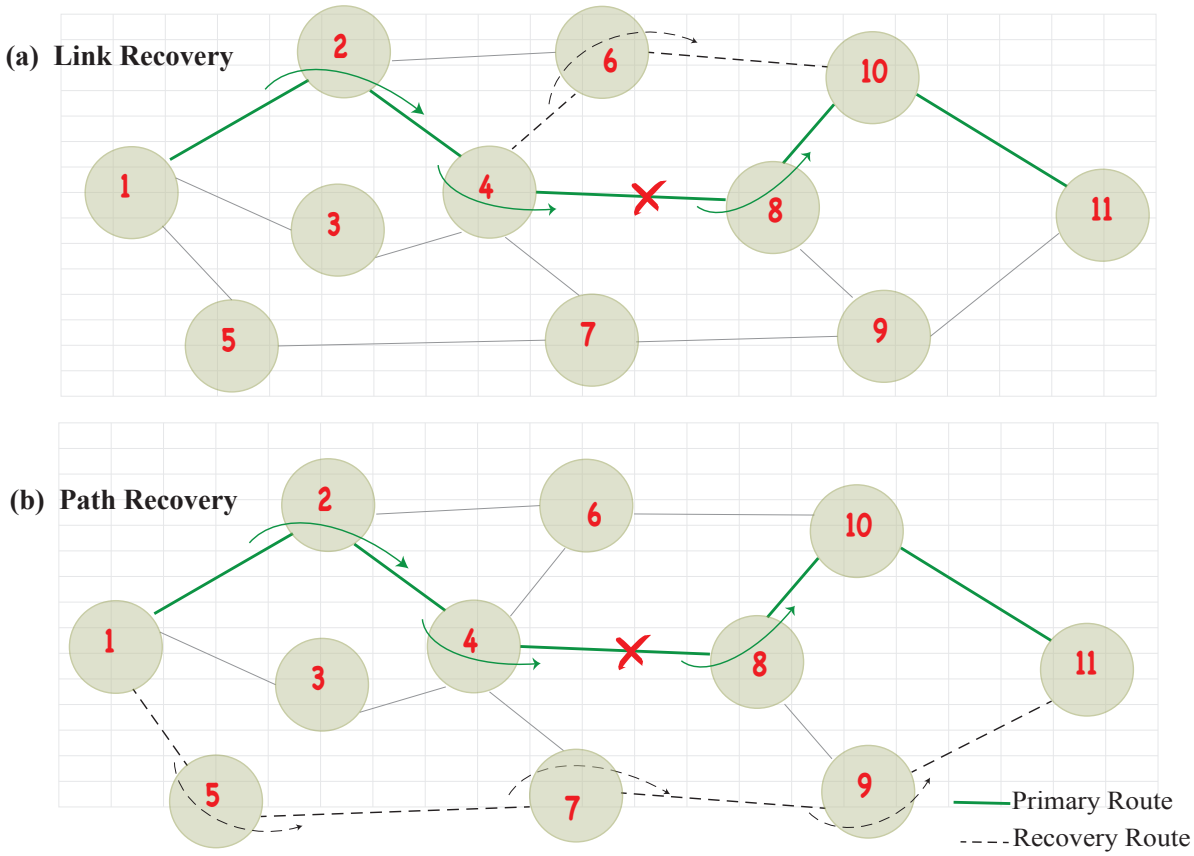

Figure 5.2 – Path Recovery vs. Link Recovery

Conventionally there are two types of network failure recovery in networking literature, namely Protection and Restoration[138] [139]. In path/link protection, resources are pre calculated in advance at set-up time and recovery measures and actions are considered early in the design time. These actions can be either, reserved link capacity or backup link/path establishment. If a failure takes place, then the effect link/path is recovered with the initial predefined actions.

In the case of restoration scheme, only when a failure takes place then reactive recovery process takes place. The control starts by calculating the backup path/link using existing information of the network. Traditionally, protection is a preferred recovery scheme for link failure and restoration is a more suited solution for path failure.

5.3.1 Fast Failover Procedure in Software Defined Networking

The fast failover is implemented by the controller and nodes can . The controller continually learns network-wide topology information. Utilising to the collected information, the SDN controller computes the route for each source and destination pair. The controller then constructs the appropriate FlowTable and GroupTable entries according the its preinstalled configurations. In the case of a node (OF switch) failure, the node can locally redirect the effected flow to a different live port. If configured appropriately, OpenFlow failover procedure can dramatically reduce failure recovery time.

As described in Chapter 1, a distinguishing feature of OpenFlow starting from version 1.1 switch architecture is tables; FlowTables and GroupTables. A FlowTable contains flow entries. With each entry having (1) match fields: that accurately defines the flow, (2) counters: that keeps track of flow statistics, and (3) instructions: which defines how the data will be treated (forward, drop or go to another table).

When PKT_IN is received, the OpenFlow switch compares it against the flow

entries within its FlowTable. If a matching entry is found, the corresponding instruction is performed and counters incremented accordingly. In the case that a flow is not found, a table-miss flow entry is created to handle the packet. This is usually by informing the SDN controller through PACKET_IN control message, and waiting for response.

With the introduction of instructions instead of actions and GroupTable starting form OpenFlow 1.1 [53] , advanced processing methods can be utilised. With actions if there is a match, the action is performed directly. While the instructions, more sophisticated operations can be performed according to the instructions which can be: (1) immediate actions, (2) a set of actions, or (3) a change to the pipeline processing to go to GroupTables.

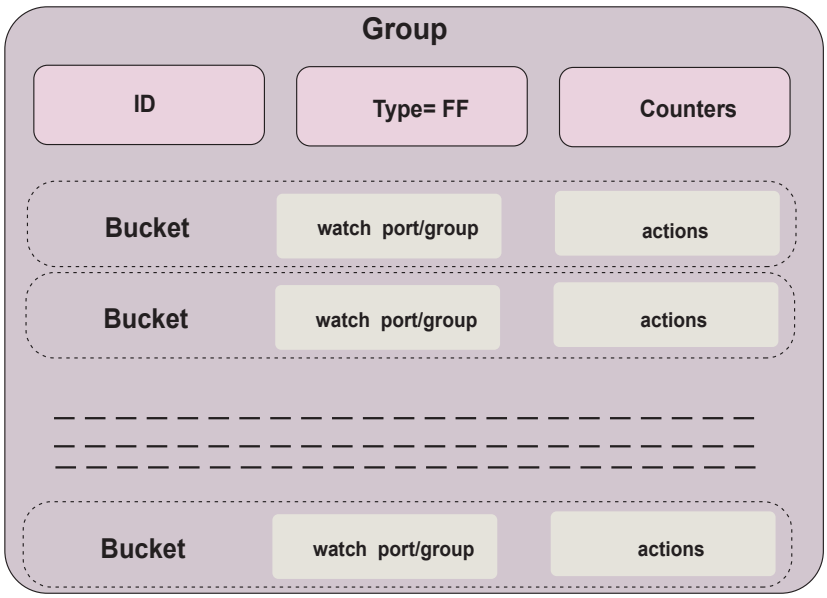

Figure 5.3 – Group Tables of OpenFlow

The GroupTable contains several group entries. Each group entry has a distinctive group ID, a group type, and a number of action buckets. Action buckets is an OpenFlow term that resembles a list of actions that may performed sequentially. For executing a specific group entry in GroupTables, the flow entry sends the packets to a

group entry containing a specific group ID. GroupTables can be of any of the following types: (1) ALL, where it is allows to perform a list of actions, used for flooding and multicasting, (2) SELECT with this type only one set of actions in the group is executed, (3) INDIRECT where the instructions are executed in the next hop, or (4) FAST FAILOVER type that allows the first alive action bucket of instructions to be executed. In our work, FAST FAILOVER group type is used to switch the flow in the case of failure by providing alternative paths once failure is detected.

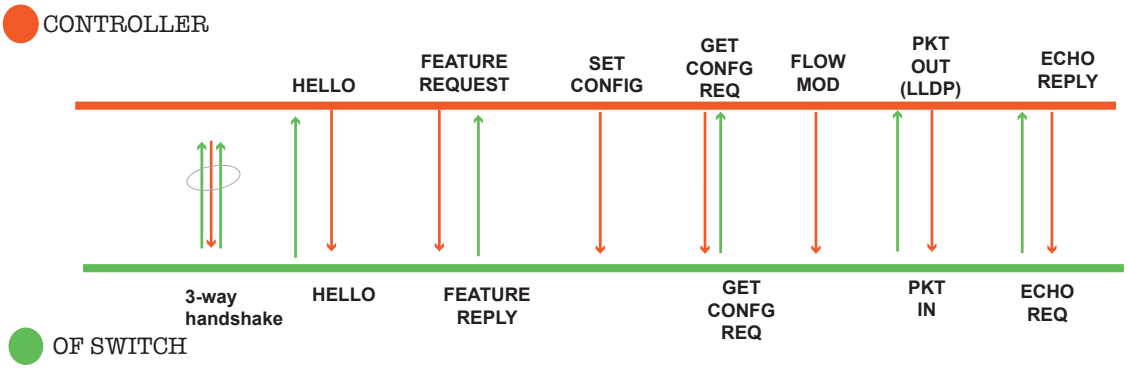

Figure 5.4 – Messages Exchange Between SDN Controller And OpenFlow Enabled Switch

The discovery process uses Open Flow Discovery Protocol " OFDP" that is just an instantiation of Link Layer Discovery Protocol " LLDP". LLDP is layer 2 protocol that transmits the device information and ports to all neighbouring networking devices. The information is sent at a fixed time interval as an advertisement message in the format [Type, Length, Value]. Basic information such as *hostname*, *description*, *port name* and others. Once received by a neighbouring OpenFlow-enabled switch, it updates its OFDP table and forward this advertisement message out to all other ports. [140]

There are three compulsory TLVs; Chassis ID, Port ID and Time To Live. Chassis ID is a unique switch identifier. It also conations other optional TLVs showing in

Each OpenFlow-enabled switch forwards these advertisement messages as

Prem.	MAC D.A.	MAC S.A.	Ether Type	Chassis I.D.	Port I.D.	TTL TLV	OPT. TLV	End OFDP	Frame CHQ

Table 5.1 – OFDP/LLDP Frame Structure

OFP_PACKET_IN to the SDN controller once the the controller broadcasts OFP_PACKET_OUT. Upon receiving all OFP_PACKET_IN from network nodes, the controller can now have a global view of the network devices. Based on this information and utilising a suitable routing algorithm, e.g. Dijkstra, the controller can construct OpenFlow tables for all participating nodes 5.5.

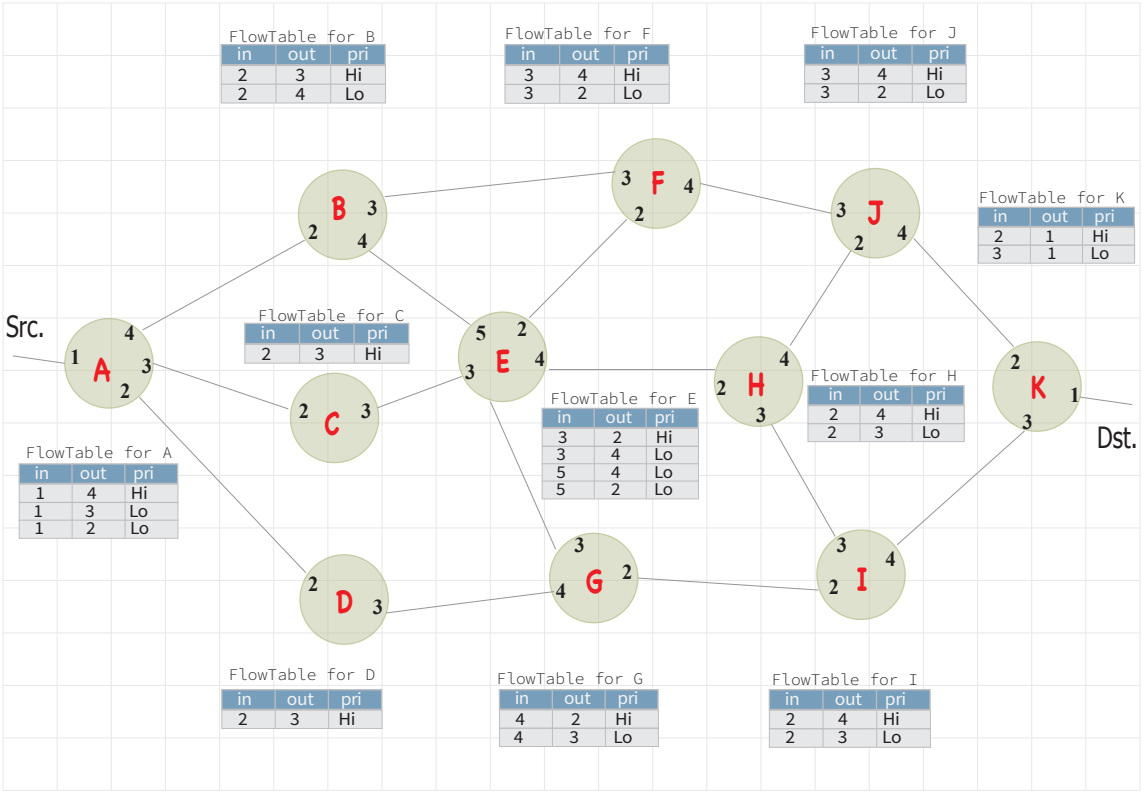

Figure 5.5 – FlowTables For A Sample Network With Input And Output Ports And Priority For The Flow From Source To Destination

After the controller receives packet out , and using OFDP. The controller constructs primary path using FlowTable and backup path for each flow using FAST_FAILOVER group. For each <source, destination> pair the SDN controller sends these flow entries for the effected nodes (OF switch). To achieve this, the controller installs tow action

buckets in the group entry of the GroupTable of type FAST_FAILOVER. The first action bucket for the primary and the second for the backup path. The previous steps are depicted in 5.6 . After receiving the port and other information through OFDP, the controller calculates the path from source to destination. In this example, the primary shortest path is calculated by the controller to be A→D→G→I→K .

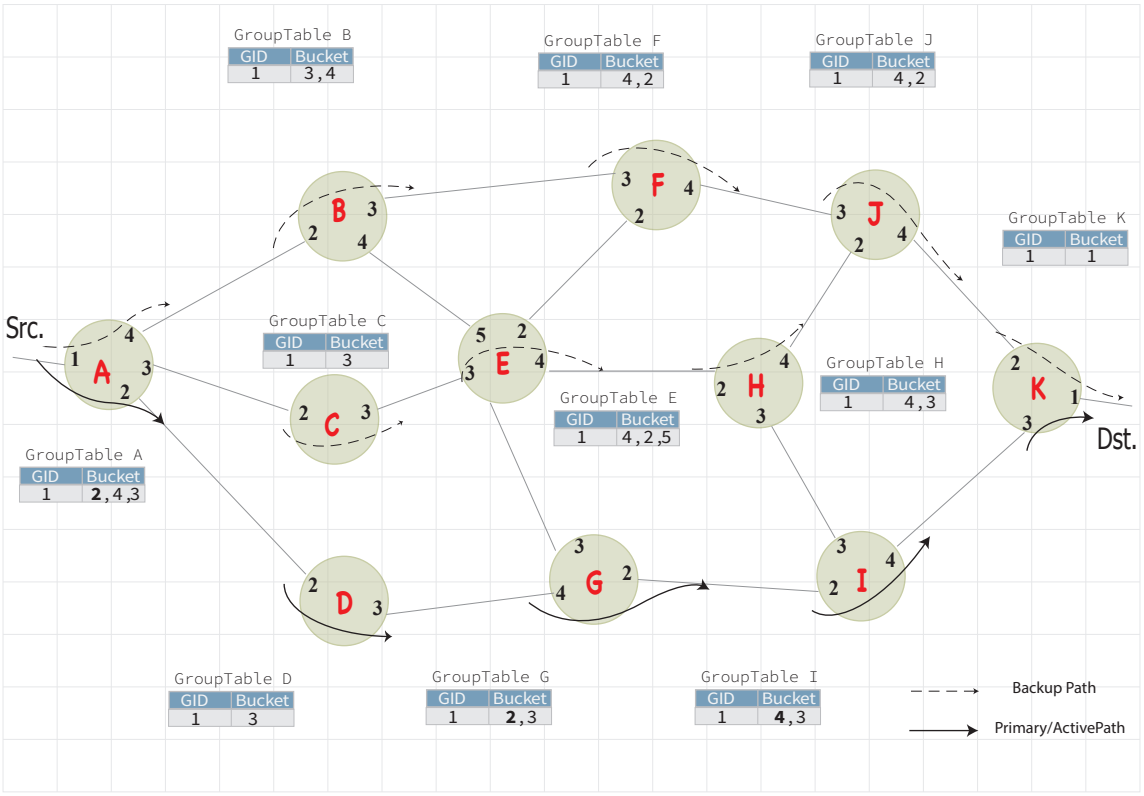

Figure 5.6 – FlowTables For A Sample Network With Input And Output Ports And Priority For The Flow From Source To Destination

Afterwards the controller computes the following backup paths A→B→F→J→K and A→C→E→H→J→K for any possible failure in the link <A,D>, in this example. Because node D has only one action bucket, there are no alternative paths to be diverted from that node. However, node A has 3 action buckets to choose from in the case of failure of the link <A,D>.

Figure 5.7 illustrates the utilisation of the backup path A→B→F→J→K if a failure takes place at link <A,D>. Once PORT_DOWN signal is detected at port number

2 of OF switch A, the traffic is instantly diverted to node D through port 4.

Match Fields	Instruction
IP_src:*src.ip* and IP_dst:*dst.ip*	GID1

Table 5.2 – FlowTable entry for Node A, Directing The Pipeline To Execute The Instructions In The GroupTable

The combination of of FlowTables and GroupTable to achieve this result is summarised in 5.2 and 5.4 respectively. It shows the flow entries for both FlowTable and GroupTable for node A, which is OpenFlow switch. The FlowTable entry for traffic comping from Src. and heading to Dest. is directed to go to `GID1` where further instructions are executed sequentially based on their liveness status.

Group Identifier	Group Type	Action Buckets
GID1	Fast Failover	Watch: A_1 port; Outport: A_2
GID1	Fast Failover	Watch: A_1 port; Outport: A_4
GID1	Fast Failover	Watch: A_1 port; Outport: A_3

Table 5.3 – GroupTable entry for Node A, Prioritising The Flow Using Primary Path As a First Entry and The Rest As a Backup Route In The Case Of Failure

When the first instruction in the action bucket becomes inactive, the second action in the actions buckets is triggered. The process continuos for the rest of created table entries.

Group Identifier	Group Type	Action Buckets
~~GID1~~	~~Fast Failover~~	~~Watch: A_1 port; Outport: A_2~~
GID1	Fast Failover	Watch: A_1 port; Outport: A_4
GID1	Fast Failover	Watch: A_1 port; Outport: A_3

Table 5.4 – A Failure In Port 2, The Next Active Entry In GroupTable Of Node A Is Activate

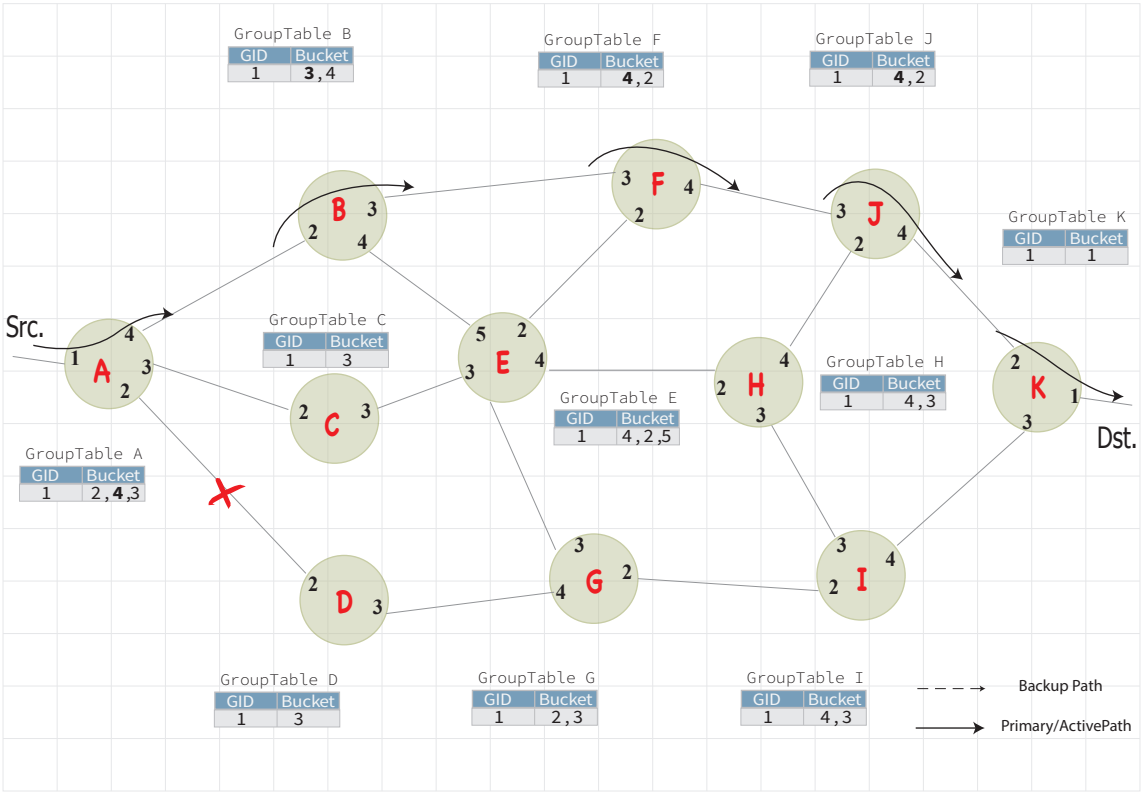

Figure 5.7 – FlowTables For A Sample Network With Input And Output Ports And Priority For The Flow From Source To Destination

5.4 Model and Simulation

The choice of an emulation environment to examine the earlier discussed failure issues, section 5.3 Failure Recovery Techniques and 5.2 Failure Detection in SDWSN, is fundamentally based on the ability of the simulation environment to embrace the new releases of OpenFlow specifications [11] [53][17][18] and Open vSwitch [52] that addressed these problems. Other fundamental features must include support of physical wireless interfaces, 6LowPAN WSN nodes, integration with mininet [141][142], and active support of Open vSwitch [52] are key elements of selecting an emulation environment from the solutions discussed in subsection 3.6.3 Emulating Solutions for SDN and IoT. The topologies were tested Mininet-WiFi which enabled the connection of 6LowPAN sensor nodes to an 802.11 wireless nodes. The sensing nodes either collect data periodically or in the case of an event takes place. Sensing nodes were acting as hosts and is attached to every edge wireless Open vSwitch node. This imitates a typical IoT application where devices collaborate to perform unconventional tasks.

In the experiment, WSN nodes, are equipped with OpenvSwitch (OVS) [52] , which enables the nodes to act as OpenFlow switches and communicate through OpenFlow protocol with the controller. OVS is production quality switch a multilayer software switch licensed under the open source Apache 2 license. OVS supports standard management interfaces and allows the programability of forwarding plane through the control plane. Ryu SDN controller [40] was chosen as the driving controller for the SDWSN network. Ryu provides software components with well defined API which makes the creation of control and management an easy task. Wireshark [143] , was used for analysing OpenFlow protocol the traffic and providing statistical data about the message exchange, protocol, size and duration. The whole system was test under Ubuntu 14.X in a virtualised environment under Parallel software for MacOS. The goal of the experiment is to compare protection and restoration techniques as a solution of WSN in an SDN environ-

ment. To test the protection mechanism, the steps in chart 5.8 were performed inside

Software	Function	Version
Open vSwitch [52]	Virtual SDN Switch	2.5.1
Ryu [40]	SDN Controller Platform	2.7.3
Mininet-WiFi[114]	Wireless SDN Network Emulator	2.3
Parallel	x86 Virtualisation Software	10.4
Linux (Ubuntu)	Guest Operating System	14.04.5
Python	Programming Language	2.7
Wireshark [143]	Netwrok Analyser	2.2.2

Table 5.5 – Applications Used In SDWSN Failure Recovery Implementation

Ryu controller. The controller broadcasts OFP_PACKET_OUT message enclosing OFDP packet for every SDWSN switch. Every switch that receives this packet forwards the OFDP part to all of its active physical ports. Once received by a participating switch, it responds by sending OFP_PACKET_IN message. The OFP_PACKET_IN is sent to the controller to inform it about how this message is received. Upon receiving this information, the controller now has the information needed to link two switches, to which port and other information. We use PKT.ETHERTYPE = 0X88CC and PKT.INPORT = CONTROLLER, to filter OFDP packets in response to the initial OFP_PACKET_OUT message. After that we use a shortest distance algorithm to establish the route from source to destination. In the case that the messages received are not enough to establish a route, the controller populates the participating ends to forward flow to OFPP_TABLE virtual port, that will treat it as per the flow already exist in legacy switching. On the other hand, if enough information is collected through the OFP_PACKET_OUT message, the controller populates the participating SDWSN nodes with the corresponding flow entries. In the case that the node has one active port (edge=1), the instructions is limited to a FlowTable entry. Whereas if there is more than active port, the controller provides protection for the node. By first creating a GroupTable entry with the first shortest path. In addition, the controller also, creates a flow entry with an instruction to go to that particular GroupTable. The GroupTables is set to type= FAST FAILOVER. In our

implementation, the controller creates entries equal to the number of edges linked to that particular node. In addition, BFD is used for detecting faulty links.

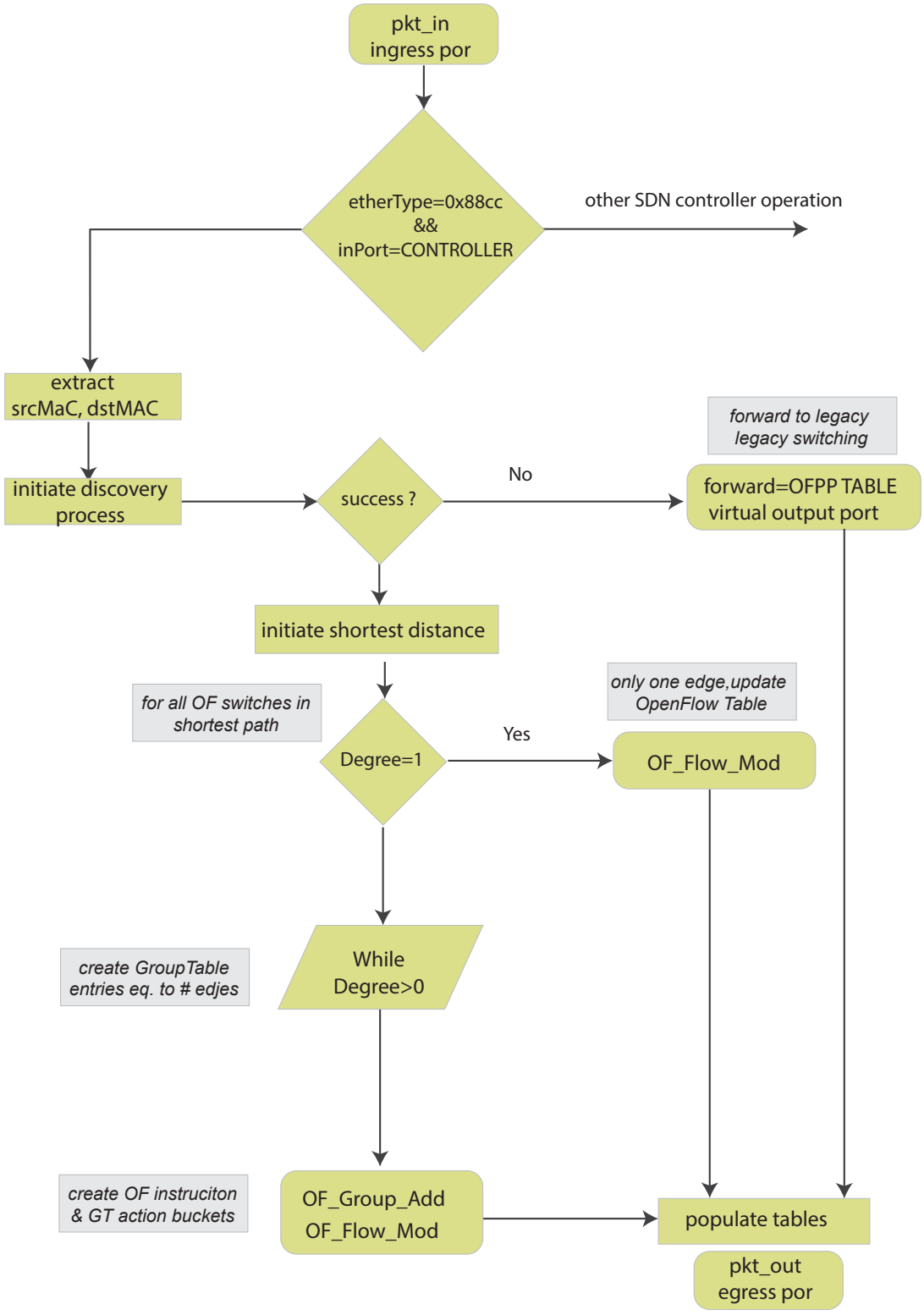

Figure 5.8 – Flow Chart For Protection Process Inside The Controller

In contrast to protection, in the case of restoration the SDWSN node reactively contacts the controller after the failure takes place. The restoration process starts by a first stage of detecting failure through Bidirectional Forwarding Detection (BFD) or Los of Signal (LoS), the later is used in this work. The second stage is from the SDWSN node to contact the controller via OFPT_MULTIPART_REQUEST and OFPT_MULTIPART_REPLY. The controller then reactively calculates an alternative link and updates the corresponding flow entries in the participating switches. Normal routing operation is then performed once the new flow entries are installed.

5.4.1 Evaluation

To evaluate the different recovery techniques in the new Software Defined Wireless Sensor Network environment, the topologies in 5.9. Topologies **a**, **b** and **c** have the same number of nodes (Open vSwitches) with varying number of links and node degrees. Node degree is the number of links to each node, which could affect the number of backup links in the case of failure. Topology **e** is the largest with 59 nodes and 148 links and nodes in this configuration has between 3 and 7 links to connect them with other nodes. Table 5.6 summarises the key differences between each topology.

The choice of these topologies is driven by the fact that the growth in the number of nodes and node degree has influence on failure recovery performance. For example, the increase of number of nodes from topologies a, to d and e; increases SDN control overhead. This in turn, affects the recovery time and increases the probability of packet collision and retransmission. The change in node degree in b and c, has direct influence on the available alternative links and the number of GroupTable entries in the case of protection mechanism. These simple and easily maintained topologies are adequate for evaluating the scope of this proposal, however other alternative topologies are possible.

Topology	Nodes	Links	Node Degree		
			Min	MEAN	Max
a	29	67	3	4.64	7
b	29	64	2	4.46	7
c	29	55	2	3.57	5
d	44	93	2	4.24	8
e	59	148	3	4.72	7

Table 5.6 – Topologies

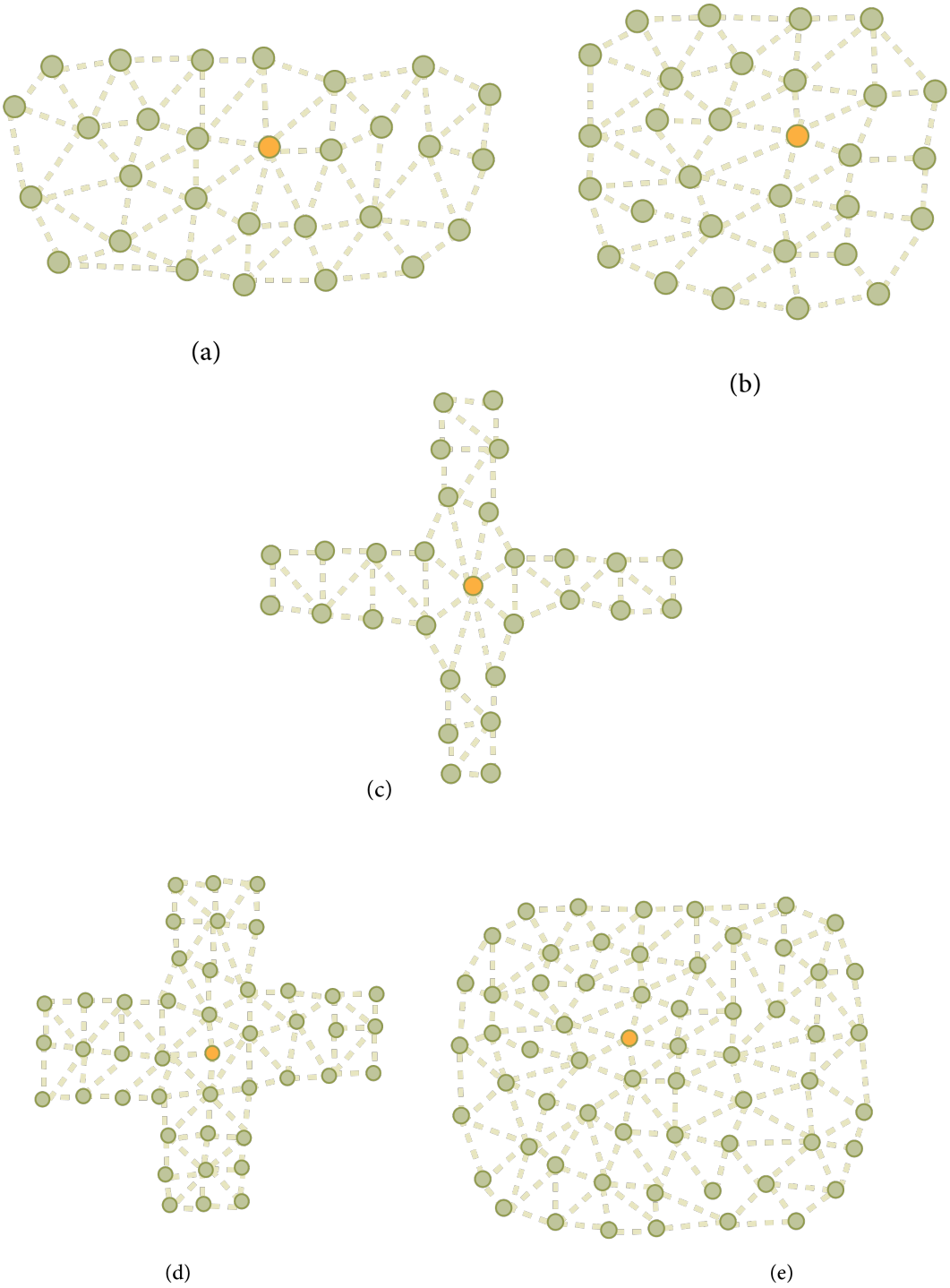

Figure 5.9 – Topologies Used for evaluating Recovery Schemes in SDWSN [1]

5.4.2 Evaluating Recovery Time

Analysis

The failure recovery time can be modelled as explained in [144] [145], which can be extended to the following model in the case of Software Defined Wireless Sensor. The process starts when a failure is detected at time T_D, after that a notification message is sent to the controller that takes time T_{prop} to reach the SDN controller. The controller then spends T_S to search for the effected flow and T_{calc} to calculate a new path. Then, T_{FM} is taken to created and send FLOW_MODE message. Then OvS switch will take T_U to update the new flow entry.

$$T_{Recovery} = T_D + T_{prop} + T_S + T_{calc} + T_{FM} + T_{prop} + T_U$$

Because we are using Dijkstra algorithm, the time taken to calculate the path T_{calc} is of time complexity $O(n^2)$ where n is the number of nodes. The measured average time for the controller to calculate alternative paths for the topologies in 5.9 is shown in 5.10.

In the case of protection, the recovery time model is the detection failure time T_D and internally updating the flow T_U with the OvS switch.

$$T_{Recovery} = T_D + T_U$$

Simulation

To evaluate the average recovery time for the protection mechanism. The topologies in figure 5.9 were installed for each network topology in Mininet-WiFi simulator. Then the a random link failure was performed to the existing working path, through "`ovs-ofctl`" utility by using the `#sudo ovs-ofctl mod-port` command. For each topology the experiment is repeated 10 times. Wireshark [143] was used to record the start of the failure of the link and the establishment of the flow again. Once a new flow entry is

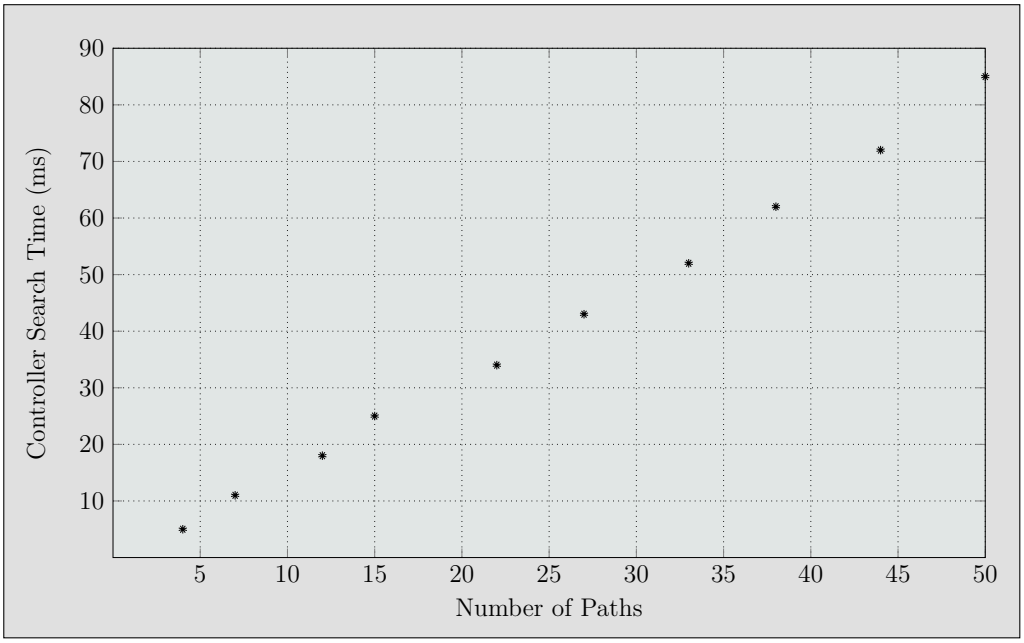

Figure 5.10 – Time Required by Controller In Finding Alternative Path In Restoration Operation

activated from `GroupTable` this ends the failure time for the link.

Similarly the process of examining the restoration process, was repeated with the same topologies. Here Open vSwitch had to consult the SDN controller to evaluate a new working path. Once the traffic resumes to arrive to the destination, this records the end of the failure time. The utilisation of GroupTables in the protection recovery technique provided smoother recovery. The recovery time for the first three topologies was almost the same of around 59 ms. Topology d and e had an average recovery time of 112 ms, and 145 ms respectively in protection technique. The recovery time for the same topologies using restoration is considerably higher. When using BFD for protection using normal `Asynchronous Echo` mode, the recovery process is performed faster than the `Asynchronous Non-Echo` mode. This is due the active echoing performed periodically in `Asynchronous Echo` BFD. The major part of the delay, however, takes place during the communication between the the tow end nodes in BFD. In LoS that is used in restoration, the delay is caused by the control message traversing from the node

detecting the failure to reach the controller and back with average time consumed for 200 ms delay. The remaining time is for the control to computer the alternative route and update flow entries.

The two BFD modes produced similar recovery time for the protection scheme. The high delay in the last topology the recovery time of the restoration technique is due to the large number of links the message has to traverse and higher node degrees. The controller takes more time to install several flow entries in their respective nodes for the new alternative route from source to destination. An average recovery time for topology e of 760 ms is very high and is not acceptable in common time-critical WSN applications [146][130].

From recovery time perspective, the protection technique that depends on OpenFlow GroupTables had a faster recovery time from network failure. This due to the use of preexisting calculated paths within the node and controller-free path reconstruction process.

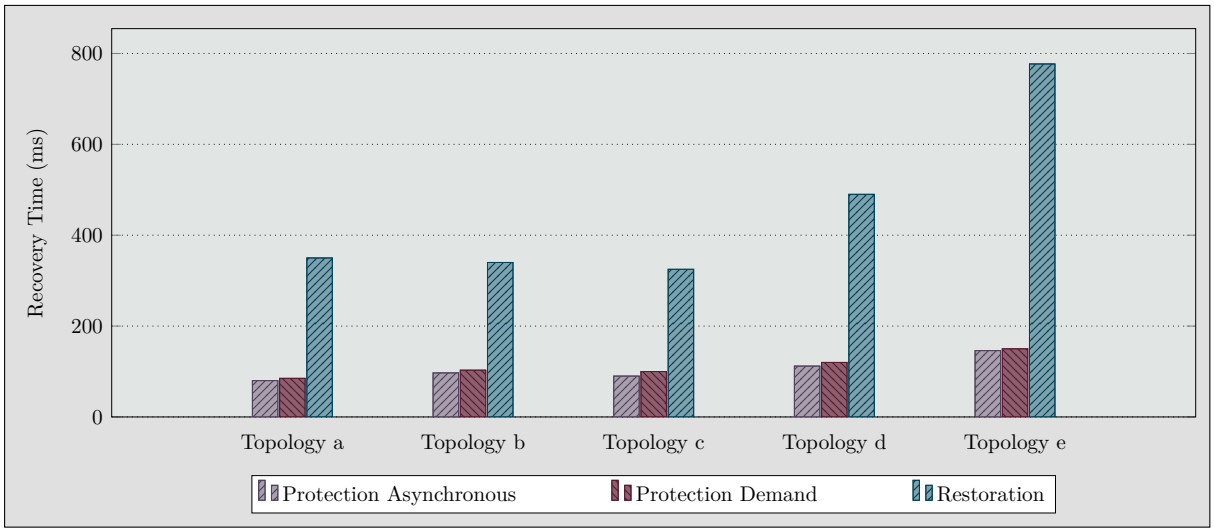

Figure 5.11 – Average Recovery Time

5.4.3 Evaluating Failure Detection Traffic

To evaluate the behaviour of control traffic in protection and restoration. In this evaluation end nodes send collected data to the the controller node in topology **c** . In this test, nodes start communicating and after specific time slot we break one of the links used as a primary path for the traffic. The control traffic is then captured in SDN controller for evaluation.

Restoration

The OpenFlow session hello messages exchanged from the nodes in the topology with the controller at the 12 second of the experiment. At the beginning, the Open vSwitch learns the path from source to destination, this is represented in the initial spikes in the graph 5.12 [22 seconds – 49seconds] of the experiment. This is followed by normal traffic from source to destination, a one second interval gap is used to avoid overloading the SDN controller. This is followed by periodical smaller spikes from the SDN controller. These are the echo messages to check the aliveness of the links in the SDN controller. At the mark of 120 second, a link is disabled in the working path of the SDWSN network. When this failure takes place, the Open vSwitch sends a failure notification message to the SDN controller. Because the restoration process requires the controller to perform the recovery process, this is indicated by a large volume of traffic around 120 second. This large spike in the figure 5.12, due to the control messages of the path reestablishment. These are FLOW_MODE that modifies the flow entry of the effect flows. In addition to the response to these notification messages of the FLOW_MODE.

Protection

Similar to the restoration initial process, the path learning process in the protection have 28 spikes at the beginning of the simulation. In the protection scheme, however, the traffic is higher in volume compared to the restoration. This is caused due to the fact

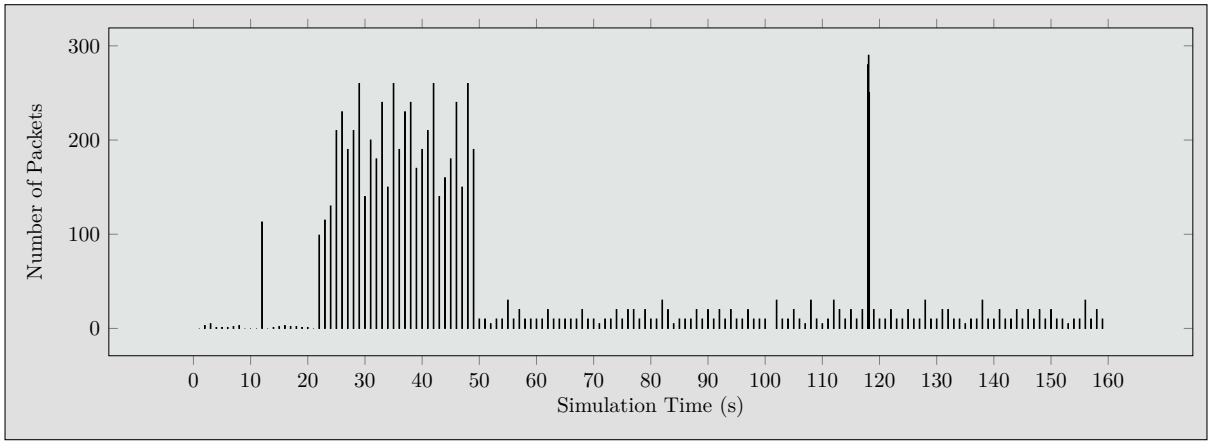

Figure 5.12 – Traffic Behaviour In Restoration Scheme

that the controller establishes a primary path and a backup path from the SDN controller and store the flow entries in addition to the GroupTable entries. The second wave of high spikes from 45 seconds to 68 seconds is beginning of the BFD session for each working path. This is followed by periodical smaller spikes from the SDN controller. These are the echo messages to check the aliveness of the links in the SDN controller. Similar to the previous restoration experiment, the same link failure process was performed at the 120 second in the protection mode. Since the Open vSwitch is not required to contact the SDN controller, the controller does not take any action. This is reflected in the lack of any significant spikes of the SDWSN network at the 120 second mark in 5.13 where link failure was performed. Here the node only uses the pre-installed alternative path.

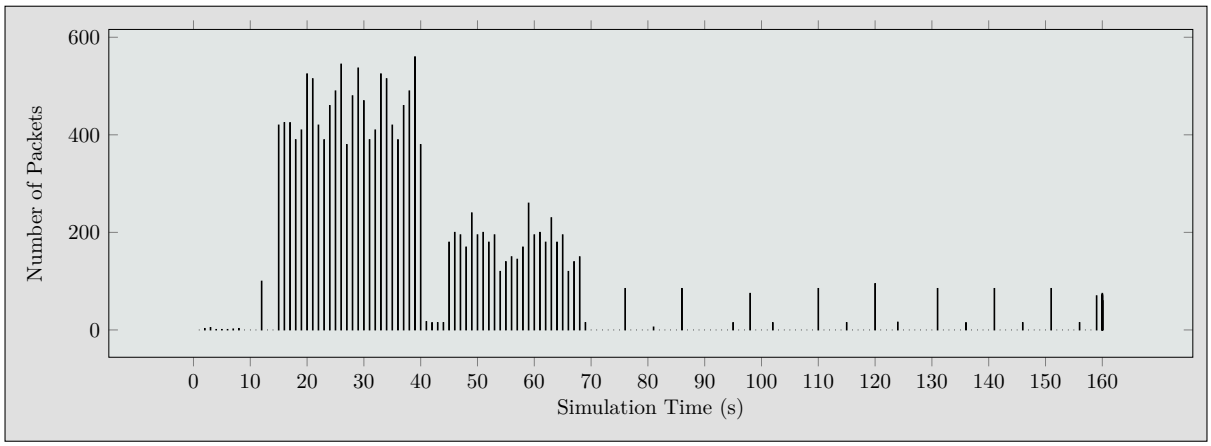

Figure 5.13 – Traffic Behaviour In Protection Scheme

5.4.4 Evaluating Number of Control Messages

The control traffic overhead is main energy consumption element that exhaust WSN nodes' power. Because the other type of data is forwarded without processing. The control messages during the discovery process include OFDP which takes place after the HELLO and FEATURES_REQUEST. For example, Topology **a** has 29 nodes and 67 links, OFDP requires 67 LLDP PACKET_OUT messages. The size of each of these OFDP messages is 160 bits. The same is required for the PACKET_IN messages from SDWSN to the controller.

Each link of the network will have an BFD session established with 24 bytes BFD Control message length for both `Asynchronous Non-Echo` and `Asynchronous Echo` in the case of protection. However, only 12 bytes BFD echo message is required as keep alive messages during the session between each participating nodes in one link through the path in `Asynchronous Echo`. In the case of `Asynchronous Non-Echo` the tow end systems keep exchanging the same control message of size 24 bytes.

For the restoration scheme, each SDWSN node exchanges keep alive message periodically with the controller. This takes place through ECHO_REQUEST and ECHO_REPLY each containing timestamp, bandwidth and liveness liveness information. Once a failure takes place the controller will send FLOW_MOD to each SDWSN in the new alternative backup path.

The use of keep-alive messages form nodes to the controller informs about the node status and once a node depletes energy, the controller can update the path of the effected nodes following restoration scheme. In the case of BFD the tow nodes exchange periodical ping messages to keep the communication alive. Once BFD signal is not received, the SDWSN can utilise group tables to choose the highest priority rule through protection mechanism. The shortest that a BFD entity can recieve control mes-

sages is `min_rx`. The default `min_rx` in Open vSwitch is 1000 milliseconds [52][134] . This is used in the experiment for the BFD `Asynchronous Echo`. For the BFD `Asynchronous Non-Echo` the time is set 1000 milliseconds for the smaller echo messages although it can be set to lower rates for achieving more aggressive error detection.

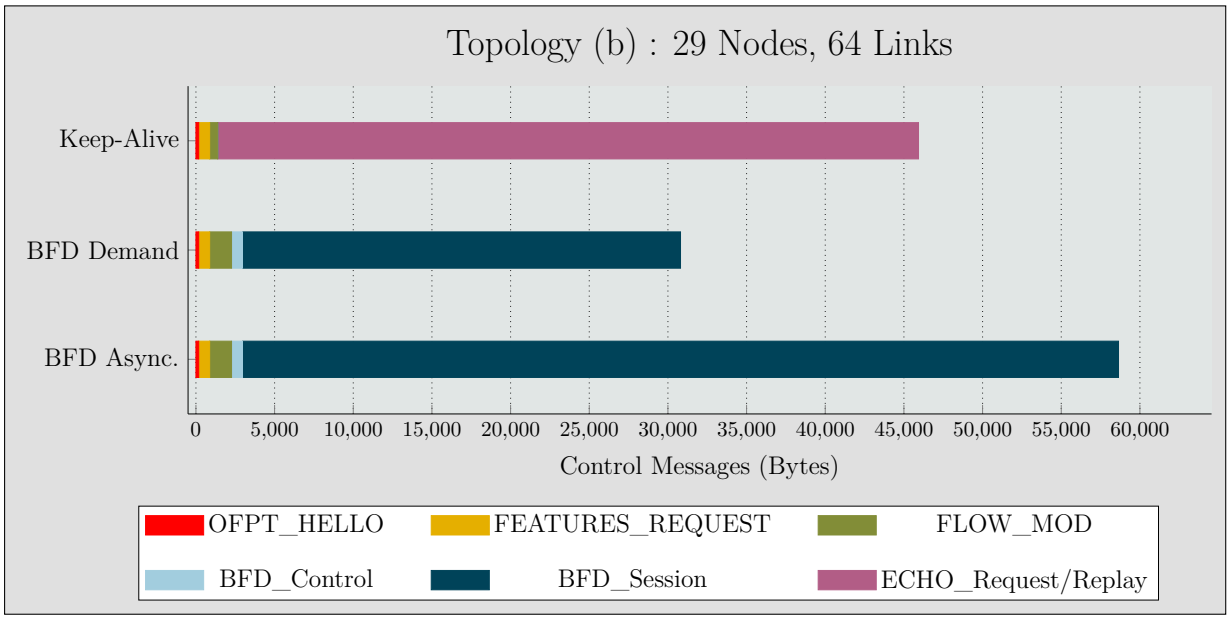

Figure 5.14 – Control Traffic In Topology a

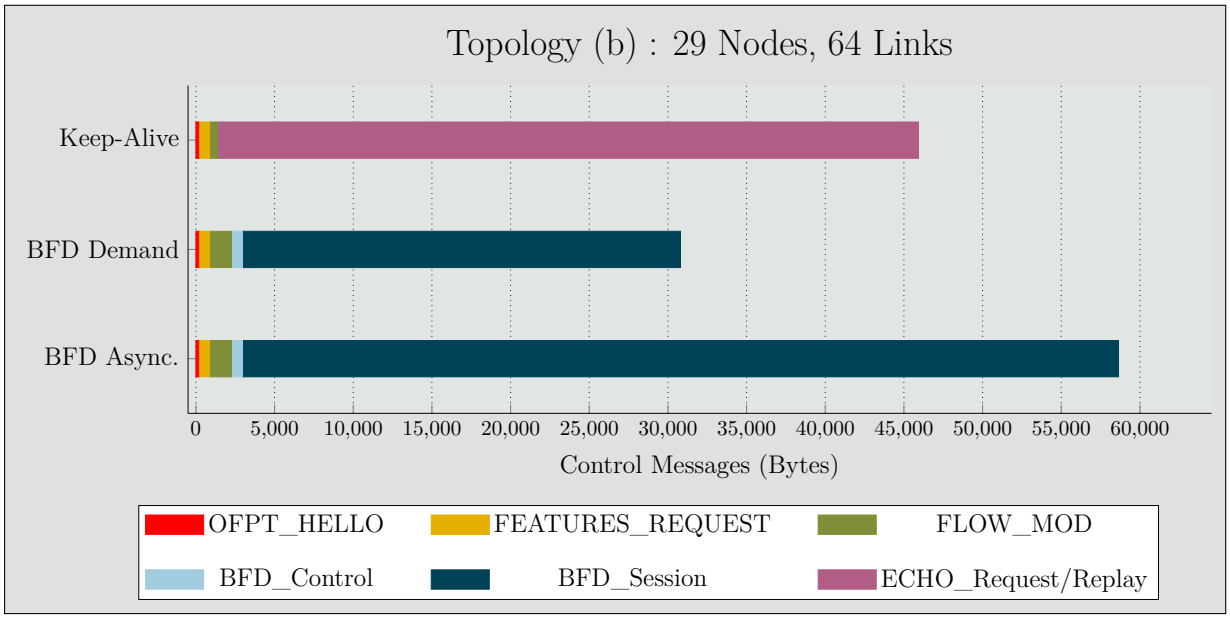

Figure 5.15 – Control Traffic In Topology b

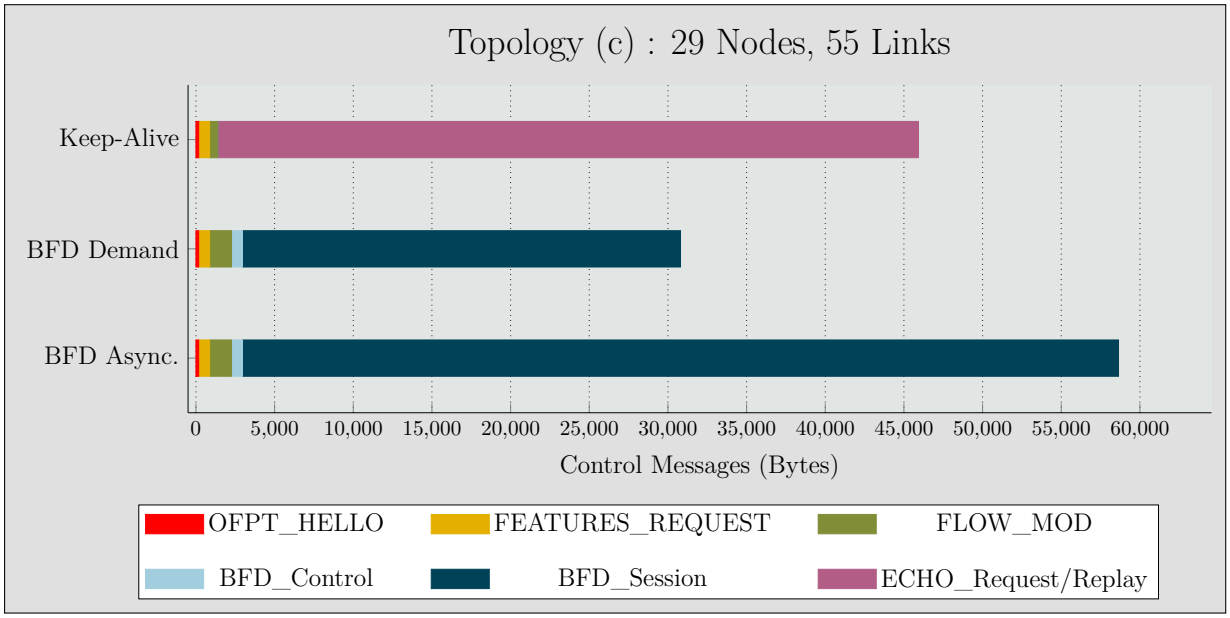

Figure 5.16 – Control Traffic In Topology c

The use of BFD `Asynchronous Echo`, as shown in Figures 5.14 5.17 and 5.18, provides huge savings in the amount of control traffic compared to the normal BFD and `Keep-Alive` schemes. This is due to the smaller message size.

The control traffic of the OpenFlow session through OFPT_HELLO, and the *features_request* has no major effect on the produces traffic. The BFD in the two examined modes shows a great advantage in utilising the `Asynchronous Echo` mode. The control traffic is reduced from 0.056 megabyte to 0.028 megabyte in Topology a 5.14. The ECHO_REQUEST and ECHO_REPLAY messages used in the restoration scheme consume 0.045 megabyte in 5.14.

Similar results are noticed in 5.17 where the `Asynchronous Echo` mode spares almost half of the control traffic due to increasing the frequency of the BFD session messages from the default 1000 milliseconds. The Keep-Alive scheme, on the other hand, maintains the same behaviour compared to BFD in `Asynchronous Non-Echo` mode.

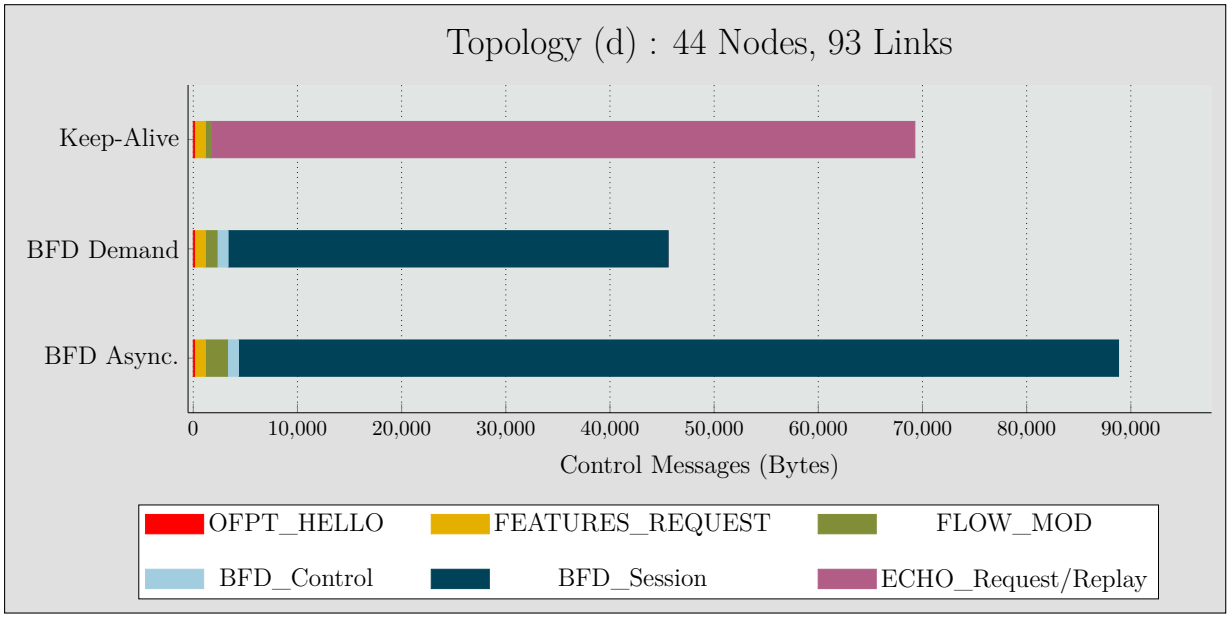

Figure 5.17 – Control Traffic In Topology d

The difference of the control traffic between the tow schemes is 17 kilobytes. The gain of using the BFD `Asynchronous Non-Echo` mode is $\backsimeq$ 42 kilobytes.

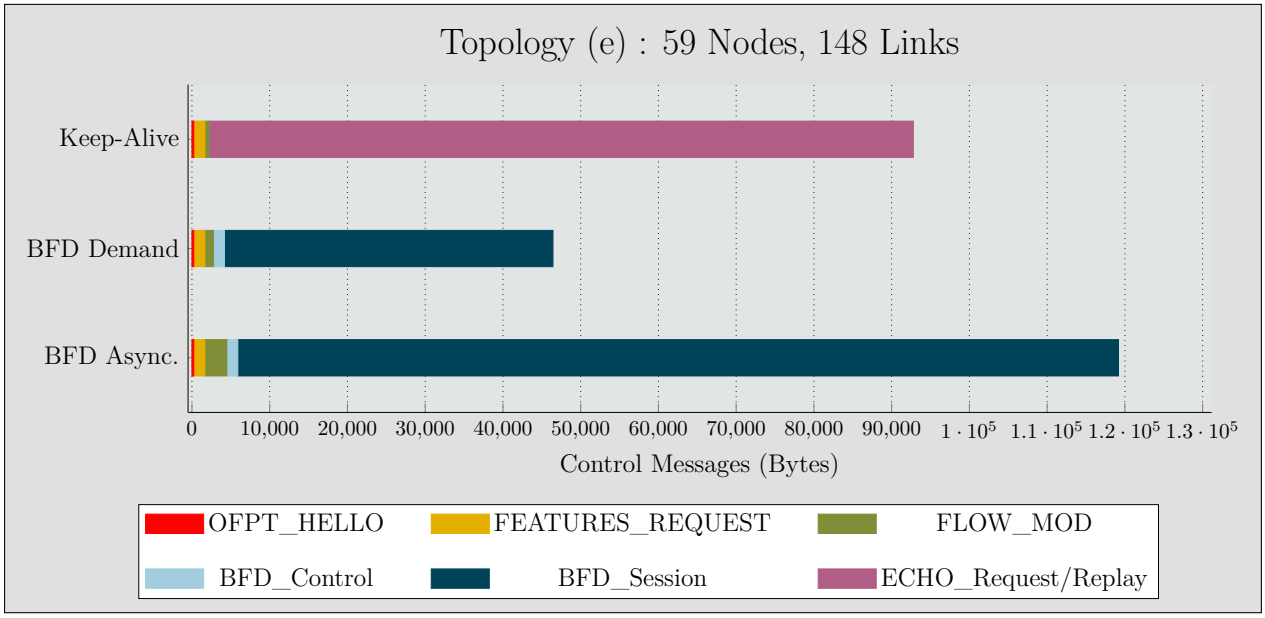

Figure 5.18 – Control Traffic In Topology e

5.5 Conclusion

In this chapter, the feasibility of SDN was examined as a possible solution for recovering WSN in the case of failure. Three different failure detection mechanisms, namely Keep-Alive, BFD `Asynchronous Echo` and BFD `Asynchronous Non-Echo` mode were examined. Simulation results show that adopting `Asynchronous Non-Echo` mode for WSN/IoT in SDN environment reduces control traffic for failure detection by 27-48%.

The protection mechanism provided faster recovery time compared to restoration. On average, protection using GroupTables of OpenFlow protocol, has up to 8 times faster recovery time than restoration, and fewer signalling control traffic over time. However, the choice of `Asynchronous Echo` or `Asynchronous Non-Echo` mode had negligible effect on the recovery time. The only only drawback of the protection scheme is the required amount of memory to store flow entries of both the `FlowTable` and `GroupTable`.

The second experiment was carried out through monitoring the SDN controller traffic. The results showed the traffic behaviour of BFD protocol has less overhead on the SDN controller than Keep-Alive messaging.

The final experiment categorised the control traffic and deeply examined the size of each type of control messages and its traffic volume. This experiment examined not only the SDN controller messages but also the traffic between the two entities participating in an BFD session. The drawback of the BFD in `Asynchronous Echo` mode was the noticeably larger than Keep-Alive (ECHO_REQUEST and ECHO_REPLAY messages). However, signalling in the case of BFD in `Asynchronous Non-Echo` mode provided a reduction of the traffic size by 27-48% against BFD in `Asynchronous Echo` mode. This is due to the difference between the packet size of the two schemes. The overload signalling of the Keep-Alive scheme is noticeably fewer than `Asynchronous Echo`.

The traffic of the earlier is from the data plane of the node to the control plane of the SDN controller. While in the later the BFD traffic is between the two data planes of the participating SDWSN node. The speed of which signal is faster depends on the placement of the SDN controller. However, as the numerical experiments showed earlier, BFD has faster failure detection time.

Chapter 6 | Conclusion

The IoT is a revolutionary technology that can take humans to the next chapter of technological evolution. It has applications in almost all walks of life, and its implementation not only improves quality of life but also increases the stability and security of humans and their surroundings. Research is focused on improving IoT systems and eliminating the challenges and issues currently hindering its wide-scale adaptation. Some of these challenges include data security, privacy, energy consumption, lack of standardisation, mobility, and interoperability. However, many new technologies are being developed to overcome these challenges and IoT is expected to progress rapidly in the near future.

6.1 Addressed Problems

This thesis addressed some aspects of the survivability problem in WSNs. With the increased interest in the IoT, with WSNs as its major building blocks, the need to increase IoT survivability and dependability is impartial. Our approach proposed the utilisation of network programmability as a mean to solve these challenges. With SDN gaining much popularity and sense, its application in WSNs raises the question of added benefit to such an environment. To achieve that, we first conducted a comprehensive literature review with a focus on SDN applications in wireless networks. The advantages of SDN technology and its potential are enormous. However, the challenge is the limited resources/research tools for applying this new concept to the existing complex environment of WSNs.

With the goal of taking advantage of the unlimited possibilities of SDN, the second step was adapting this new networking paradigm to WSNs. We addressed the following survivability challenges in existing WSN environments.

6.1.1 Lifetime Maximisation for SDWSN

We investigated practical solutions that extend WSN lifetime and address the resource-constrained nature of wireless nodes when migrated into SDN (i.e. SDWSN). The problem was mathematically formulated as a linear program conforming with the centralised nature of an SDN environment, and residual energy level constraints were imposed on WSN nodes to determine the route that maximises the network lifetime in an SDN environment.

We proposed also an A-star-based routing algorithm to maximise WSNs in an SDN environment. The algorithm, in addition to finding the shortest path for WSN nodes, benefits from the SDN controller to obtain fair distribution of traffic to maximise resource utilisation among the nodes. We built a simulation model of the algorithm, in which it was deployed within the SDN controller. The algorithm was then evaluated against existing WSN energy-saving algorithms.

Our simulation results indicate that SDN improved network lifetime by 16% compared with the traditional LEACH. This result is due to the fact that SDN technology requires a smaller number of control messages and that less processing power is consumed because the route calculations are performed in the SDN controller rather than in the sensor nodes in traditional routing. However, considering network lifetime definition as being 50% of the depleted network nodes offers a brighter perspective; the lifetime has gained an impressive 22.6% increase. These results confirm the feasibility of our proposed approach compared with classical routing algorithms.

In particular, this centralised routing scheme provided a graceful network out-

age and energy consumption drop. Due to the balancing of power use in the network, wireless sensor nodes depleted their energy sources at approximately the same time. This finding is highly beneficial since all the nodes can be recharged or replaced simultaneously, instead of constantly monitoring and servicing individual devices.

6.1.2 Failure Recovery of SDWSN

To address the failure detection problem, we presented a comparative evaluation of existing SDN failure detection techniques and evaluated their applicability in IoT simulation environment. The control traffic overhead and failure detection time were the main criteria in this evaluation. The selection of any particular technique is an application based decision, which our work contributing to clarify. For failure recovery, we measured the performance various combinations of existing failure recovery techniques that adhere to SDN specifications constrains and suit IoT applications. At the centre of this contribution lie the identification of architectural aspects and considerations of `OpenFlow` SDN protocol when applied to a resource-constrained environment like IoT.

Our extensive experimentation of the feasibility of SDN as a viable environment for solving failure recovery in WSN and future IoT. Three different failure detection mechanisms, namely Keep-Alive, BFD `Asynchronous Echo` and BFD `Asynchronous Non-Echo` mode were examined.

The protection mechanism provided faster recovery time compared to restoration. However, the choice of `Asynchronous Echo` or `Asynchronous Non-Echo` mode had no significant effect on the recovery time. The only only drawback of the protection scheme is the required extra memory space to store flow entries of both the `FlowTable` and `GroupTable`. The second experiment was carried out through monitoring the SDN controller traffic. The results showed the traffic behaviour of BFD protocol has less overhead on the controller than Keep-Alive messaging. The final exper-

iment categorised the control traffic and deeply examined the size of each type of control messages and its traffic volume. This experiment examined not only the SDN controller messages but also the traffic between the two entities participating in an BFD session. The drawback of the BFD in `Asynchronous Echo` mode was noticeably larger than Keep-Alive (ECHO_REQUEST and ECHO_REPLAY messages). However, the custom BFD in `Asynchronous Echo` mode provided a reduction of the traffic size of no less than half the BFD in `Asynchronous Non-Echo` mode.

6.2 Discussion and Future Work

The failure detection techniques investigated here provide an insight into the viability of the BFD `Asynchronous Echo` mode. However, not all IoT applications require continuous sensing activity. Therefore, the periodic echo process might not suit these applications. One possible solution to this problem is the use of the BFD demand mode. This mode can be implemented, for example, based on the number of transmitted or received packets. In this context, the BFD protocol depends on the interface statistics to trigger the error detection functionality. Currently, however, Open vSwitch does not support this functionality; the implementation of such a functionality requires modifications of OvS kernel.

In failure recovery, the protection mechanism that utilises GroupTables had a quicker recovery time than the restoration scheme. However, the larger the size of the network, the more memory storage for both FlowTables and GroupTables. This factor raises the question of the trade-off between memory cost and performance, which could be a future optimisation research problem.

Our proposed lifetime maximisation algorithm provided nodes with remaining power after the end network coverage. This finding could lead to future improvement by investigating network lifetime regarding coverage. In addition, the feasibility of using

multiple SDN controllers in the future could result in greater network lifetime.

Although software defined networking shows good improvements in terms of lifetime compared to some well-known protocols such as LEACH, there is an area of improvement in terms of optimising the number of SDN controllers. For example, the study of energy hole problem could be solved by this optimisation. In the energy hole problem, nodes near the sink or controller exhaust their energy much sooner than other nodes.

Bibliography

[1] J. Rak, *Resilience of Wireless Mesh Networks*, pp. 85–120. Cham: Springer International Publishing, 2015.

[2] G. Bell, "Bell's law for the birth and death of computer classes: A theory of the computer's evolution," *IEEE Solid-State Circuits Society Newsletter*, vol. 13, no. 4, pp. 8–19, 2008.

[3] T. Nakagawa, G. Ono, R. Fujiwara, T. Norimatsu, T. Terada, M. Miyazaki, K. Suzuki, K. Yano, Y. Ogata, A. Maeki, *et al.*, "1-cc computer: Cross-layer integration with uwb-ir communication and locationing," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 4, pp. 964–973, 2008.

[4] J. Postel, "Transmission Control Protocol." RFC 793 (Internet Standard), Sept. 1981. Updated by RFCs 1122, 3168, 6093, 6528.

[5] K.-K. Yap, R. Sherwood, M. Kobayashi, T.-Y. Huang, M. Chan, N. Handigol, N. McKeown, and G. Parulkar, "Blueprint for introducing innovation into wireless mobile networks," in *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, pp. 25–32, ACM, 2010.

[6] M. P. Gallaher and B. R. Rowe, "The costs and benefits of transferring technology infrastructures underlying complex standards: The case of ipv6," *The Journal of Technology Transfer*, vol. 31, pp. 519–544, Sep 2006.

[7] S. Ruthfield, "The internet's history and development: from wartime tool to fish-cam," *Crossroads*, vol. 2, no. 1, pp. 2–4, 1995.

[8] "Ietf document statistics." `https://www.arkko.com/tools/rfcstats/`.

[9] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for sdn? implementation challenges for software-defined networks," *IEEE Communications Magazine*, vol. 51, pp. 36–43, July 2013.

[10] N. Operators, "Network functions virtualization, an introduction, benefits, enablers, challenges and call for action," in *SDN and OpenFlow SDN and OpenFlow World Congress*, 2012.

[11] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69–74, Mar. 2008.

[12] ONF, "OpenFlow Switch Specification 1.3." `https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf`.

[13] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using openflow: A survey," *IEEE Communications Surveys Tutorials*, vol. 16, pp. 493–512, First 2014.

[14] N. Feamster, J. Rexford, and E. Zegura, "The road to sdn: An intellectual history of programmable networks," *SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 87–98, Apr. 2014.

[15] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, *et al.*, "Onix: A distributed control platform for large-scale production networks.," in *OSDI*, vol. 10, pp. 1–6, 2010.

[16] ONF, "OpenFlow Switch Specification Version 1.0.0." `https://www.opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.0.0.pdf`, December 2009.

[17] ONF, "OpenFlow Switch Specification Version 1.4.0." `https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.4.0.pdf`, October 2013.

[18] ONF, "OpenFlow Switch Specification Version 1.5.0." `https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.0.pdf`, December 2014.

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

[20] B. Bhatta, *Research Methods In Remote Sensing*. Springer, 2013.

[21] R. R. Brooks, *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks*. SAGE Publications Sage UK: London, England, 2008.

[22] J. Gao, "Analysis of energy consumption for ad hoc wireless sensor networks using a bit-meter-per-joule metric," *IPN Progress Report*, vol. 42, no. 150, 2002.

[23] A. M. H. G. P. Kobo, Hlabishi I.; Abu-Mahfouz, "A survey on software-defined wireless sensor networks: Challenges and design requirements," *IEEE access*, vol. 5, pp. 1872–1899, 2017.

[24] "Open Networking Foundation Members." `https://www.opennetworking.org/member-listing/`.

[25] MATLAB, *Release 2014b*. Natick, Massachusetts: The MathWorks Inc., 2014.

[26] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *System sciences, 2000. Proceedings of the 33rd annual Hawaii international conference on*, pp. 10–pp, IEEE, 2000.

[27] Y. Perwej, K. Haq, F. Parwej, M. Mumdouh, and M. Hassan, "The internet of things (iot) and its application domains," *International Journal of Computer Applications*, vol. 975, p. 8887.

[28] BBC, "Bcc research report on internet of things (iot) networks: Technologies and global markets to 2022.," report, 2017.

[29] J. Ramson, Jino; Moni, "Applications of wireless sensor networks - a survey," in *2017 International Conference on Innovations in Electrical, Electronics, Instrumentation and Media Technology (ICEEIMT)*, 2017.

[30] M. G.-M. A. A. M.-M. M. O.-L. M. A. Q.-L. F. J. Moreno-Moreno, Carlos D.; Brox-Jiménez, "Wireless sensor network for sustainable agriculture," *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 2, no. 20, p. 1304, 2018.

[31] Y. C.-S. I. S. Ali, Ahmad; Ming, "A comprehensive survey on real-time applications of wsn," *Future Internet*, vol. 9, no. 4, p. 77, 2017.

[32] M. Mittal and S. C. Pandey, *The Rudiments of Energy Conservation and IoT*, pp. 1–17. Springer, 2019.

[33] K. Chopra, K. Gupta, and A. Lambora, "Future internet: The internet of things-a literature review," in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, pp. 135–139.

[34] S. Madakam, R. Ramaswamy, and S. Tripathi, "Internet of things (iot): A literature review," *Journal of Computer and Communications*, vol. 3, no. 05, p. 164, 2015.

[35] 2016.

[36] E. C.-F. M. F. Mekki, Kais; Bajic, "A comparative study of lpwan technologies for large-scale iot deployment," *ICT express*, vol. 5, no. 1, pp. 1–7, 2019.

[37] T. N. Z. Z. T. H. W. T. Queralta, J. Penã; Gia, "Comparative study of lpwan technologies on unlicensed bands for m2m communication in the iot: beyond lora and lorawan," *Procedia Computer Science*, vol. 155, pp. 343–350, 2019.

[38] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox: Towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[39] "POX Controller." `https://github.com/noxrepo/`.

[40] "Ryu Component-Based Software Defined Networking Framework." `https://github.com/osrg/ryu`.

[41] "OpenDayLight Project." `http://www.opendaylight.org/`.

[42] "Floodlight Project." `http://www.projectfloodlight.org/floodlight/`.

[43] D. Erickson, "The beacon openflow controller," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pp. 13–18, ACM, 2013.

[44] "OpenMUL Controller." `http://www.openmul.org/`.

[45] "Trema : Full-Stack OpenFlow Framework in Ruby and C." `https://trema.github.io/trema/`.

[46] "OpenStack Foundation." `https://www.openstack.org/`.

[47] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, "Sdn controllers: A comparative study," in *2016 18th Mediterranean Electrotechnical Conference (MELECON)*, pp. 1–6, April 2016.

[48] "OF–CONFIG 1.2." `https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow-config/of-config-1.2.pdf`.

[49] R. Enns (Ed.), M. Bjorklund (Ed.), J. Schoenwaelder (Ed.), and A. Bierman (Ed.), "Network Configuration Protocol (NETCONF)." RFC 6241 (Proposed Standard), June 2011. Updated by RFC 7803.

[50] A. L. Stancu, S. Halunga, A. Vulpe, G. Suciu, O. Fratu, and E. C. Popovici, "A comparison between several software defined networking controllers," in *2015 12th International Conference on Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS)*, pp. 223–226, Oct 2015.

[51] J. Naous, D. Erickson, G. A. Covington, G. Appenzeller, and N. McKeown, "Implementing an openflow switch on the netfpga platform," in *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ANCS '08, (New York, NY, USA), pp. 1–9, ACM, 2008.

[52] "The Openflow Switch." `http://www.openvswitch.org/`.

[53] ONF, "OpenFlow Switch Specification Version 1.1.0." `https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.1.0.pdf`, October 2014.

[54] V. K. S. J. V. Bala, Tarun; Bhatia, "A survey: issues and challenges in wireless sensor network," *International Journal of Engineering & Technology*, vol. 7, no. 2, p. 77, 2018.

[55] A. Kharb, Seema; Singhrova, *Next-Generation Networks*, ch. Review of Industrial Standards for Wireless Sensor Networks. Singapore, 2018.

[56] C. Lim, "A survey on congestion control for rpl-based wireless sensor networks," *Sensors*, vol. 19, no. 11, p. 2527, 2019.

[57] J. J. R. R. A. A.-M. J. K. V. Sobral, José VV; Rodrigues, "Routing protocols for low power and lossy networks in internet of things applications," *Sensors*, vol. 19, no. 9, p. 2144, 2019.

[58] M. R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. A. Grieco, G. Boggia, and M. Dohler, "Standardized protocol stack for the internet of (important) things," *IEEE communications surveys & tutorials*, vol. 15, no. 3, pp. 1389–1406, 2012.

[59] T. Winter, "Rpl: Ipv6 routing protocol for low power and lossy networks," *RFC6550*, 2011.

[60] O. Iova, F. Theoleyre, and T. Noel, "Using multiparent routing in rpl to increase the stability and the lifetime of the network," *Ad Hoc Networks*, vol. 29, pp. 45–62, 2015.

[61] D. Culler and U. Berkeley, "Hydro: A hybrid routing protocol for lossy and low power networks draft-tavakoli-hydro-01," *Netw. Work. Group IETF, Fremont, CA, USA, Tech. Rep*, 2009.

[62] K. Kim, G. Montenegro, S. Park, I. Chakeres, and C. Perkins, "Dynamic manet on-demand for 6lowpan (dymo-low) routing," *Internet Engineering Task Force, Internet-Draft*, 2007.

[63] T. Clausen, J. Yi, and U. Herberg, "Lightweight on-demand ad hoc distance-vector routing-next generation (loadng): Protocol, extension, and applicability," *Computer Networks*, vol. 126, pp. 125–140, 2017.

[64] G. Wu, C. Yang, S. Li, and G. Y. Li, "Recent advances in energy-efficient networks and their application in 5g systems," *IEEE Wireless Communications*, vol. 22, no. 2, pp. 145–151, 2015.

[65] M. S. Al-Kahtani, "Efficient cluster-based sleep scheduling for m2m communication network," *Arabian Journal for Science and Engineering*, vol. 40, no. 8, pp. 2361–2373, 2015.

[66] R. Ahmad, M. A. Asim, S. Z. Khan, and B. Singh, "Green iot—issues and challenges," *Available at SSRN 3350317*, 2019.

[67] I. Al Ridhawi, M. Aloqaily, Y. Kotb, Y. Jararweh, and T. Baker, "A profitable and energy-efficient cooperative fog solution for iot services," *IEEE Transactions on Industrial Informatics*, 2019.

[68] G. Yang, X.-W. Wu, Y. Li, and Q. Ye, "Energy efficient protocol for routing and scheduling in wireless body area networks," *Wireless Networks*, pp. 1–9, 2019.

[69] J. Pullmann and D. Macko, "A new planning-based collision-prevention mechanism in long-range iot networks," *IEEE Internet of Things Journal*, 2019.

[70] D. Ghose, "Protocol design and performance evaluation of wake-up radio enabled iot networks," *Doctoral dissertations at University of Agder*, 2019.

[71] Y. Sasaki, T. Yokotani, and H. Mukai, "Mqtt over vlan for reduction of overhead on information discovery," in *2019 International Conference on Information Networking (ICOIN)*, pp. 354–356, IEEE.

[72] R. Piyare, O. Berder, and R. L. Cigno, *Wake-up Radio based Approach to Low-Power and Low-Latency Communication in the Internet of Things.* Thesis, 2019.

[73] S. Y. Shahdad, M. Khan, H. Sultana, M. A. Hussain, and S. M. Bilfaqih, "Routing protocols for constraint devices internet of things network," in *2019 International Conference on Communication and Signal Processing (ICCSP)*, pp. 0114–0117, IEEE.

[74] M. B. Yassein, A. Flefil, D. Krstic, Y. Khamayseh, W. Mardini, and M. Shatnawi, "Performance evaluation of rpl in high density networks for internet of things (iot)," in *Proceedings of the 2019 8th International Conference on Software and Information Engineering*, pp. 183–187, ACM.

[75] M. D. Alshehri, F. Hussain, M. Elkhodr, and B. S. Alsinglawi, *A Distributed Trust Management Model for the Internet of Things (DTM-IoT)*, pp. 1–9. Springer, 2019.

[76] J. Bauwens, B. Jooris, S. Giannoulis, I. Jabandžić, I. Moerman, and E. De Poorter, "Portability, compatibility and reuse of mac protocols across different iot radio platforms," *Ad Hoc Networks*, vol. 86, pp. 144–153, 2019.

[77] 2019.

[78] M. A. L. Peña and I. M. Fernández, "Sat-iot: An architectural model for a high-performance fog/edge/cloud iot platform," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pp. 633–638, IEEE.

[79] G. Hosangadi, D. Wang, and A. Rao, "System design considerations for internet of things (iot) with category-m devices in lte networks," *arXiv preprint arXiv:1902.00408*, 2019.

[80] A. Capone, C. Cascone, A. Q. T. Nguyen, and B. Sansò, "Detour planning for fast and reliable failure recovery in SDN with OpenState," in *2015 11th International*

Conference on the Design of Reliable Communication Networks (DRCN), pp. 25–32, mar 2015.

[81] M. Johnston, H. W. Lee, and E. Modiano, "A Robust Optimization Approach To Backup Network Design With Random Failures," *Proceedings - IEEE INFOCOM*, vol. 23, no. 4, pp. 1512–1520, 2011.

[82] S. S. W. Lee, K. Y. Li, K. Y. Chan, G. H. Lai, and Y. C. Chung, "Path layout planning and software based fast failure detection in survivable OpenFlow networks," *DRCN 2014 - Proceedings, 10th International Conference on Design of Reliable Communication Networks*, 2014.

[83] J. Feigenbaum, B. Godfrey, A. Panda, M. Schapira, S. Shenker, and A. Singla, "Brief announcement," *Proceedings of the 2012 ACM symposium on Principles of distributed computing - PODC '12*, 2012.

[84] M. Borokhovich and S. Schmid, "How (not) to shoot in your foot with sdn local fast failover," in *Principles of Distributed Systems* (R. Baldoni, N. Nisse, and M. van Steen, eds.), (Cham), pp. 68–82, Springer International Publishing, 2013.

[85] J. Liu, A. Panda, A. Singla, B. Godfrey, M. Schapira, and S. Shenker, "Ensuring connectivity via data plane mechanisms.," in *NSDI*, pp. 113–126, 2013.

[86] S. Yamaguchi, A. Nakao, M. Oguchi, A. Goto, and S. Yamamoto, "Monitoring dynamic modification of routing information in openflow networks," in *Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication*, IMCOM '16, (New York, NY, USA), pp. 27:1–27:7, ACM, 2016.

[87] N. A. Jagadeesan and B. Krishnamachari, "Software-defined networking paradigms in wireless networks: a survey," *ACM Computing Surveys (CSUR)*, 2015.

[88] D. B. Rawat and S. Reddy, "Recent advances on Software Defined Wireless Networking," *SoutheastCon*, pp. 1–8, 2016.

[89] *OpenFlow for Wireless Mesh Networks*, IEEE, 2011.

[90] J. Chung, G. González, I. Armuelles, T. Robles, R. Alcarria, and A. Morales, "Experiences and Challenges in Deploying OpenFlow over Real Wireless Mesh Networks," *IEEE Latin America Transactions*, vol. 11, no. 3, pp. 955–961, 2013.

[91] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," RFC 3561, Internet Engineering Task Force, July 2003.

[92] A. Neumann, C. Aichele, M. Lindner, and S. Wunderlich, "Better approach to mobile ad-hoc networking (batman)," tech. rep., Internet Engineering Task Force, 2008.

[93] P. Dely, A. Kassler, and N. Bayer, "Openflow for wireless mesh networks," in *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pp. 1–6, IEEE, 2011.

[94] G. Araniti, J. Cosmas, A. Iera, A. Molinaro, R. Morabito, and A. Orsino, "Openflow over wireless networks: Performance analysis," in *Broadband Multimedia Systems and Broadcasting (BMSB), 2014 IEEE International Symposium on*, pp. 1–5, IEEE, 2014.

[95] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, p. 60, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[96] W.-S. Kim, S.-H. Chung, C.-W. Ahn, and M.-R. Do, "Seamless handoff and performance anomaly reduction schemes based on openflow access points," in *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on*, pp. 316–321, IEEE, 2014.

[97] A. S. Yuan, H.-T. Fang, and Q. Wu, "Openflow based hybrid routing in wireless sensor networks," in *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference on*, pp. 1–5, IEEE, 2014.

[98] A. Detti, C. Pisa, S. Salsano, and N. Blefari-Melazzi, "Wireless mesh software defined networks (wmsdn)," in *2013 IEEE 9th international conference on wireless and mobile computing, networking and communications (WiMob)*, pp. 89–95, IEEE, 2013.

[99] T. Clausen and P. Jacquet, "Optimized link state routing protocol (olsr)," tech. rep., ietf.org, https://tools.ietf.org/html/rfc3626, October 2003.

[100] F. Yang, V. Gondi, J. O. Hallstrom, K.-C. Wang, and G. Eidson, "Openflow-based load balancing for wireless mesh infrastructure," in *Consumer Communications and Networking Conference (CCNC), 2014 IEEE 11th*, pp. 444–449, IEEE, 2014.

[101] S. Hasan, Y. B. David, R. C. Scott, E. Brewer, and S. Shenker, "Enabling rural connectivity with sdn," *Technical Report No. UCB/EECS-2012–201*, 2012.

[102] S. Hasan, Y. Ben-David, C. Scott, E. Brewer, and S. Shenker, "Enhancing rural connectivity with software defined networks," in *Proceedings of the 3rd ACM Symposium on Computing for Development*, p. 49, ACM, 2013.

[103] S. Ruponen, "On software-defined networking for rural areas: controlling wireless networks with openflow," in *International Conference on e-Infrastructure and e-Services for Developing Countries*, pp. 39–48, Springer, 2013.

[104] M. Mendonca, K. Obraczka, and T. Turletti, "The case for software-defined networking in heterogeneous networked environments," in *Proceedings of the 2012 ACM conference on CoNEXT student workshop*, pp. 59–60, ACM, 2012.

[105] G. Sato, N. Uchida, and Y. Shibata, "Performance evaluation of pc router based cognitive wireless network for disaster-resilient wans," in *Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on*, pp. 611–616, IEEE, 2014.

[106] H. Ali-Ahmad, C. Cicconetti, A. de la Oliva, M. Dräxler, R. Gupta, V. Mancuso, L. Roullet, and V. Sciancalepore, "Crowd: an sdn approach for densenets," 2013.

[107] B. O. Kahjogh and G. Bernstein, "Energy and latency optimization in software defined wireless networks," *International Conference on Ubiquitous and Future Networks, ICUFN*, pp. 714–719, 2017.

[108] A. Alfieri, A. Bianco, P. Brandimarte, and C. F. Chiasserini, "Maximizing system lifetime in wireless sensor networks," vol. 181, pp. 390–402, 2007.

[109] B. Behdani, Y. S. Yun, J. Cole Smith, and Y. Xia, "Decomposition algorithms for maximizing the lifetime of wireless sensor networks with mobile sinks," *Computers and Operations Research*, vol. 39, no. 5, pp. 1054–1061, 2012.

[110] 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.

[111] L. Peizhe, W. Muqing, L. Wenxing, and Z. Min, "A game-theoretic and energy-efficient algorithm in an improved software-defined wireless sensor network," *IEEE Access*, vol. 5, pp. 13430–13445, 2017.

[112] 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, p. 360428, 2015.

[113] M. F. Tuysuz, Z. K. Ankarali, and D. Gözüpek, "A survey on energy efficiency in software defined networks," *Computer Networks*, vol. 113, pp. 188–204, 2017.

[114] R. R. Fontes, S. Afzal, S. H. Brito, M. A. Santos, and C. E. Rothenberg, "Mininet-wifi: Emulating software-defined wireless networks," in *Network and Service Management (CNSM), 2015 11th International Conference on*, pp. 384–389, IEEE, 2015.

[115] M. H. R. Jany, N. Islam, R. Khondoker, and M. A. Habib, "Performance analysis of openflow based software defined wired and wireless network," in *Computer and Information Technology (ICCIT), 2017 20th International Conference of*, pp. 1–6, IEEE, 2017.

[116] J. H. Cox, R. Clark, and H. Owen, "Leveraging sdn and webrtc for rogue access point security," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 756–770, 2017.

[117] N. Apolónia, F. Freitag, L. Navarro, S. Girdzijauskas, and V. Vlassov, "Gossip-based service monitoring platform for wireless edge cloud computing," in *Networking, Sensing and Control (ICNSC), 2017 IEEE 14th International Conference on*, pp. 789–794, IEEE, 2017.

[118] M. Ozcelik, N. Chalabianloo, and G. Gur, "Software-defined edge defense against iot-based ddos," in *2017 IEEE International Conference on Computer and Information Technology (CIT)*, pp. 308–313, IEEE, 2017.

[119] H. T. Larasati, F. H. Ilma, B. Nuhamara, A. Mustafa, R. Hakimi, and E. Mulyana, "Performance evaluation of handover association mechanisms in sdn-based wireless network," in *Wireless and Telematics (ICWT), 2017 3rd International Conference on*, pp. 103–108, IEEE, 2017.

[120] I. D. F. Tarigan and D.-S. Kim, "Wireless station mobility and load balancing for wireless lan," 2016.

[121] H. T. Larasati, R. Hakimi, and T. Juhana, "Extended-llf: A least loaded first (llf)-based handover association control for software-defined wireless network," *International Journal of Computer Engineering and Information Technology*, vol. 9, no. 9, p. 203, 2017.

[122] S. Indriyanto, M. N. D. Satria, A. R. Sulaeman, R. Hakimi, and E. Mulyana, "Performance analysis of vanet simulation on software defined network," in *Wireless and Telematics (ICWT), 2017 3rd International Conference on*, pp. 81–85, IEEE, 2017.

[123] B. Van Leeuwen, J. Eldridge, and V. Urias, "Cyber analysis emulation platform for wireless communication network protocols," in *Security Technology (ICCST), 2017 International Carnahan Conference on*, pp. 1–6, IEEE, 2017.

[124] M. Chan, C. Chen, J. Huang, T. Kuo, L. Yen, and C. Tseng, "Opennet: A simulator for software-defined wireless local area network," in *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 3332–3336, April 2014.

[125] T. Hu, K. Xue, W. Wei, and W. Jiang, "Lenv: A new light-weighted edge network virtualization framework in software-defined wireless networks," in *2015 International Conference on Wireless Communications Signal Processing (WCSP)*, pp. 1–6, Oct 2015.

[126] Y. Liu, C. Chen, and S. Chakraborty, "A software defined network architecture for geobroadcast in vanets," in *2015 IEEE International Conference on Communications (ICC)*, pp. 6559–6564, June 2015.

[127] M. Karimi, M. S. Najafi, R. Akbari, and M. Keshtgari, "Presenting a new method, using topology virtualization for stabilizing flow tables in sdwn," in *2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC)*, vol. 00, pp. 219–226, Oct 2018.

[128] F. Pakzad, S. Layeghy, and M. Portmann, "Evaluation of mininet-wifi integration via ns-3," in *2016 26th International Telecommunication Networks and Applications Conference (ITNAC)*, vol. 00, pp. 243–248, Dec. 2016.

[129] N. Cardona, J. F. Botero, and D. Ospina, "Handoff management for smart access points in ieee 802.11 networks," in *2017 IEEE Colombian Conference on Communications and Computing (COLCOM)*, pp. 1–6, Aug 2017.

[130] H. H. R. Sherazi, L. A. Grieco, and G. Boggia, "A comprehensive review on energy harvesting mac protocols in wsns: Challenges and tradeoffs," *Ad Hoc Networks*, vol. 71, pp. 117 – 134, 2018.

[131] Z. Hu and B. Li, "On the fundamental capacity and lifetime limits of energy-constrained wireless sensor networks," in *Real-Time and Embedded Technology and Applications Symposium, 2004. Proceedings. RTAS 2004. 10th IEEE*, pp. 2–9, IEEE, 2004.

[132] E. J. Duarte-Melo, M. Liu, and A. Misra, "A modeling framework for computing lifetime and information capacity in wireless sensor networks," in *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, Cambridge, UK, March, 2004.

[133] R. Fourer, D. M. Gay, and B. Kernighan, *Ampl*, vol. 117. Boyd & Fraser Danvers, MA, 1993.

[134] D. Katz and D. Ward, "Bidirectional Forwarding Detection (BFD) for Multihop Paths," RFC 5883, Internet Engineering Task Force, June 2010.

[135] A. Yaqini, "Managing wireless mesh networks–a survey of recent fault recovery approaches," in *International Conference on Mobile Computing, Applications, and Services*, pp. 317–324, Springer, 2015.

[136] L. Paradis and Q. Han, "A survey of fault management in wireless sensor networks," *Journal of Network and systems management*, vol. 15, no. 2, pp. 171–190, 2007.

[137] M. Pióro and D. Medhi, *Routing, Flow, and Capacity Design in Communication and Computer Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

[138] M. Pióro and D. Medhi, "CHAPTER 10 - Application of Optimization Techniques for Protection and Restoration Design," in *Routing, Flow, and Capacity Design in Communication and Computer Networks* (M. Pióro and D. Medhi, eds.), The Morgan Kaufmann Series in Networking, pp. 403 – 454, San Francisco: Morgan Kaufmann, 2004.

[139] M. Pióro and D. Medhi, "CHAPTER 9 - Restoration and Protection Design of Resilient Networks," in *Routing, Flow, and Capacity Design in Communication and Computer Networks* (M. Pióro and D. Medhi, eds.), The Morgan Kaufmann Series in Networking, pp. 353 – 401, San Francisco: Morgan Kaufmann, 2004.

[140] "IEEE Std 802.1AB-2009 : IEEE Standard For Local And Metropolitan Area Networks– Station And Media Access Control Connectivity Discovery," IEEE Std 802.1AB-2009 (Revision of IEEE Std 802.1AB-2005) 3561, IEEE, `http://ieeexplore.ieee.org/servlet/opac?punumber=5251688`, Sept. 2009.

[141] ONF, "Mininet: Rapid Prototyping for Software Defined Networks." `https://github.com/mininet/mininet`.

[142] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, p. 19, ACM, 2010.

[143] "Wireshark." `https://www.wireshark.org/`.

[144] E. Mannie (Ed.) and D. Papadimitriou (Ed.), "Recovery (Protection and Restoration) Terminology for Generalized Multi-Protocol Label Switching (GMPLS)." RFC 4427 (Informational), Mar. 2006.

[145] J.-P. Vasseur, M. Pickavet, and P. Demeester, *Network recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. Elsevier, 1st edition ed., July 2004.

[146] C. Pham, "Communication performances of ieee 802.15.4 wireless sensor motes for data-intensive applications: A comparison of waspmote, arduino mega, telosb, micaz and imote2 for image surveillance," *Journal of Network and Computer Applications*, vol. 46, pp. 48 – 59, 2014.