CAPITAL UNIVERSITY OF SCIENCE AND TECHNOLOGY, ISLAMABAD

A Robust Routing Architecture for Hybrid Wireless Mesh and Software Defined Networks

by

Mukhtiar Bano

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Faculty of Engineering
Department of Electrical Engineering

2023

A Robust Routing Architecture for Hybrid Wireless Mesh and Software Defined Networks

By

Mukhtiar Bano

(PCE141001)

Dr. Thierry Turletti, Professor

INRIA Sophia Antipolis, Cedex, France

(Foreign Evaluator 1)

Dr. Isabelle Guérin Lassous, Professor

Université Lyon I UFR d'Informatique, Bâtiment Nautibus, France

(Foreign Evaluator 2)

Dr. Amir Qayyum

(Thesis Supervisor)

Dr. Noor Muhammad Khan

(Head, Department of Electrical Engineering)

Dr. Imtiaz Ahmad Taj

(Dean, Faculty of Engineering)

DEPARTMENT OF ELECTRICAL ENGINEERING

CAPITAL UNIVERSITY OF SCIENCE AND TECHNOLOGY

ISLAMABAD

2023

To my parents

Thanks for teaching me the ethics and discipline that today guide my life.

To my family

Thanks for your unconditional love and support.

CAPITAL UNIVERSITY OF SCIENCE & TECHNOLOGY
ISLAMABAD

Expressway, Kahuta Road, Zone-V, Islamabad
Phone:+92-51-111-555-666 Fax: +92-51-4486705
Email: info@cust.edu.pk Website: https://www.cust.edu.pk

CERTIFICATE OF APPROVAL

This is to certify that the research work presented in the dissertation, entitled "**A Robust Routing Architecture for Hybrid Software Defined and Wireless Mesh Networks**" was conducted under the supervision of **Dr. Amir Qayyum**. No part of this dissertation has been submitted anywhere else for any other degree. This dissertation is submitted to the **Department of Electrical Engineering, Capital University of Science and Technology** in partial fulfillment of the requirements for the degree of Doctor in Philosophy in the field of **Electrical Engineering**. The open defence of the dissertation was conducted on **July 24, 2023**.

Student Name : Mukhtiar Bano (PCE141001) [illegible]

The Examining Committee unanimously agrees to award PhD degree in the mentioned field.

Examination Committee :

(a) External Examiner 1:	Dr. Majid Iqbal Khan, Professor COMSATS University, Islamabad	[illegible]
(b) External Examiner 2:	Dr. Muhammad Zeeshan, Associate Professor SEECS NUST, Islamabad	[illegible]
(c) Internal Examiner :	Dr. Muhammad Siraj Rathore Assistant Professor CUST, Islamabad	[illegible]
Supervisor Name :	Dr. Amir Qayyum Professor CUST, Islamabad	[illegible]
Name of HoD :	Dr. Noor Muhammad Khan Professor CUST, Islamabad	[illegible]
Name of Dean :	Dr. Imtiaz Ahmad Taj Professor CUST, Islamabad	[illegible]

AUTHOR'S DECLARATION

I, **Mukhtiar Bano (Registration No. PCE141001)**, hereby state that my dissertation entitled, '**A Robust Routing Architecture for Hybrid Software Defined and Wireless Mesh Networks**' is my own work and has not been submitted previously by me for taking any degree from Capital University of Science and Technology, Islamabad or anywhere else in the country/ world.

At any time, if my statement is found to be incorrect even after my graduation, the University has the right to withdraw my PhD Degree.

[illegible]

(Mukhtiar Bano)

Dated: 24th July, 2023

Registration No : PCE141001

<u>**PLAGIARISM UNDERTAKING**</u>

I solemnly declare that research work presented in the dissertation titled "**A Robust Routing Architecture for Hybrid Software Defined and Wireless Mesh Networks**" is solely my research work with no significant contribution from any other person. Small contribution/ help wherever taken has been duly acknowledged and that complete dissertation has been written by me.

I understand the zero-tolerance policy of the HEC and Capital University of Science and Technology towards plagiarism. Therefore, I as an author of the above titled dissertation declare that no portion of my dissertation has been plagiarized and any material used as reference is properly referred/ cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled dissertation even after award of PhD Degree, the University reserves the right to withdraw/ revoke my PhD degree and that HEC and the University have the right to publish my name on the HEC/ University Website on which names of students are placed who submitted plagiarized dissertation.

[illegible]

(Mukhtiar Bano)

Dated: 24th July, 2023

Registration No : PCE141001

List of Publications

It is certified that following publication(s) has been made out of the research work that has been carried out for this thesis:-

1. Bano, Mukhtiar, Amir Qayyum, Rao Naveed Bin Rais, and Syed Sherjeel A. Gilani. "Soft-Mesh: A Robust Routing Architecture for Hybrid SDN and Wireless Mesh Networks." IEEE Access 9 (2021): 87715-87730. DOI:10.1109/ACCESS.2021.3089020

2. Gilani, Syed Sherjeel A., Amir Qayyum, Rao Naveed Bin Rais, and Mukhtiar Bano. "SDNMesh: An SDN based routing architecture for wireless mesh networks." IEEE Access 8 (2020): 136769-136781. DOI: 10.1109/ACCESS.2020 .3011651

3. Bano, Mukhtiar, Rao Naveed Bin Rais, Syed Sherjeel A. Gilani, and Amir Qayyum. "SDNHybridMesh: A hybrid routing architecture for SDN based wireless mesh networks." In Workshops of the International Conference on Advanced Information Networking and Applications, pp. 366-375. Springer, Cham, 2020. DOI: 10.1007/978-3-030-44038-1_33

4. Bano, Mukhtiar, Syed Sherjeel A. Gilani, and Amir Qayyum. "A comparative analysis of hybrid routing schemes for SDN based wireless mesh networks." In 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 1189-1194. IEEE, 2018. DOI: 10.1109/HPC-C/SmartCity/DSS.2018.00200

(Mukhtiar Bano)

Registration No: PCE141001

Acknowledgement

Many people have encouraged, impacted, and assisted me during the process that led to this dissertation. First and foremost, I want to express my gratitude to my supervisor, Dr. Amir Qayyum, for allowing me to do my research. His critical but always constructive style, as well as his attention to the tiniest details, aided me in various facets of completing rigorous scientific work. I appreciate all of his invaluable counsel, constructive criticism, and moral support. Dr. Qayyum has served as a mentor in both my academic and personal lives, always guiding and assisting me when I face difficulties and disappointments. I'd also like to express my gratitude to Dr. Rao Naveed Bin Rais, equally important have been his critical reviews, suggestions, and counsel. I am grateful for his time and contribution to my dissertation.

Many thanks to all of my co-workers who assisted, encouraged, and accompanied me in overcoming all of the challenges in my research. Last but not least, I'd like to express my gratitude to my family. None of this would have been possible without their help throughout and after the journey.

(Mukhtiar Bano)

Registration No: PCE141001

Abstract

Wireless Mesh Networks (WMNs) are considered self-organizing, self-healing, and self-configuring networks. Notwithstanding these appealing characteristics, WMNs face a number of challenges including scalability, reliability, and connection failures, as well as mobility, flexibility, and other network management problems. To address these issues, WMNs must be made programmable, allowing standard approaches to be modified and applied using software programs, which can be accomplished by incorporating Software Defined Networking (SDN) architecture. SDN, as a cutting-edge technology, promises to make network management and routing challenges in wireless mesh networks easier. The existing solutions mainly propose for entire network paradigm shift from legacy WMN to SDN. However, a few existing solutions propose partial deployment resulting in interoperability issues of hybrid architectures. The evolution of the legacy network model as a whole, causes technical, operational, and economic challenges that can be avoided by a seamless interoperability between SDN and current IP devices.

This research presents a Robust Routing Architecture for Hybrid Wireless Mesh and Software Defined Networks, by systematically and gradually migrating WMNs to SDNs in an effective manner. The primary goal of this study is to improve the routing of wireless mesh networks by partial and incremental transitioning from legacy network to software defined network. Such transition makes WMNs programmable to address the issues and challenges of various network services such as routing, load balancing, network control, and traffic engineering. In addition, the architecture of SDN node is modified which facilitates in bridging the interoperability gap between SDN nodes and legacy nodes. Nonetheless, software defined networking paradigm faces the challenge of single-point-of-failure due to centralized controlling entity 'SDN Controller', the proposed hybrid routing approach also addresses routing challenge in case of controller failure or unreachable. The proposed approach makes congestion control easier by load balancing approach that makes use of adaptive network monitoring module. It allows a network administrator to keep track of network traffic as well as link information such as link utilization and bandwidth, that are required by the controller for load balancing.

For evaluation purpose Mininet-WiFi Simulator is used to run several experiments to evaluate the performance of the proposed architecture by generating a hybrid network topology with varying number of nodes that is 50, 100, 150, 200, and 250 nodes respectively, with varied proportions of SDN hybrid and legacy nodes. Furthermore, the percentage of SDN hybrid nodes and legacy nodes in the network topology is configurable to produce three alternative simulation scenarios, with 10%, 25%, and 50% of SDN hybrid nodes and 90%, 75%, and 50% of legacy nodes in the network topology.

The OLSR and BATMAN routing systems are simulated because they are regarded to be more stable than any other standard routing approaches for wireless mesh networks. Soft-Mesh architecture has been compared to wmSDN and Hakiri for hybrid approaches. In terms of different performance measures, such as average UDP throughput, end-to-end delay, packet drop ratio, and routing overhead, the suggested routing Soft-Mesh gives improved results. For the incremental fraction of SDN hybrid nodes, Soft-Mesh improves the results by 70%. As a result, our findings suggest that the SDN method will benefit the distributed routing protocol's operations. Furthermore, the proposed routing architecture also provides a routing solution in case of controller goes down or failure occurs, this particular solution makes our routing approach robust and reliable. Results show that our proposed routing architecture's average throughput behavior, packet drop ratio, end-to-end delay, and routing overhead are extremely similar to that of OLSR with a little increase due to the rules already installed by controller at the setting up of network. Such an improvement is brought about by the cohabitation of OpenFlow and OLSR in the SDN hybrid nodes architecture.

The proposed network monitoring architecture 'AdNetMon', is a flexible, adapt able, and expandable architecture for hybrid SDN network, that is introduced in this chapter. The generic RESTful API can be used to specify almost every aspect of monitoring. Furthermore, new techniques can replace the basic components in the proposed architecture without affecting the other components. We introduced an adaptive scheduling approach for flow statistics gathering to demonstrate the

effectiveness of the proposed architecture. Using this approach, we created a specific use case of monitoring connection consumption. We assessed and compared it to the performance of Flowsense and a periodic polling method. We discovered that the proposed approach can collect statistics with more accuracy than Flowsense. Despite this, the incurred communications cost is up to 50% higher than in a comparable periodic poling technique.

Contents

List of Figures

List of Tables

Abbreviations

APIs	Applications Programming Interfaces
BMCP	Budgeted Maximum Coverage Problem
CBR	Constant bitrate
CPLR	Control Packet Loss Ratio
CW	Continuous wave
EFTM	Embedded Flow Table Manager
GPIA	Generic Path Inconsistency Avoider
IP	Internet Protocol
Mbps	Megabits per Second
MPR	Multi-point relays
NFV	Network function virtualization
O2O	OLSR-to-OpenFlow
ODL	OpenDaylight
OSGi	Open Service Gateway Initiative
OVS	Open vSwitch
RTT	Round-trip time
SCTs	Solitary Confinement Trees
SDN	Software-Defined Networking
SLA	Service Level Agreements
TCAM	Ternary Content Addressable Memory
TC	Traffic Control
WMNs	Wireless Mesh Networks
WPA	Wi-Fi Protected Access

Symbols

τ	Initial Statitics collection timeout
β	Beta
ϕ	Phi
α	Alpha
∞	Infinity
$\triangle$	Threshold

Chapter 1

Introduction

Traditional network administration is getting increasingly complex and difficult to handle user demands and network consumption growth at a rapid rate. Despite the fact that various types of applications including cloud computing, mobile data, huge datasets, and multimedia applications along with huge network traffic, have been used to generate significant revenue, these applications continue to present network operators with numerous operational and performance challenges [1]. Modern networks must still be efficient and flexible while coping with these issues; nonetheless, network programming is one way for making these networks more efficient and versatile. Software-Defined Networking (SDN) paradigm, which is based on the idea of centralizing control and administration and provides a solution for network management and control concerns, can be used to accomplish network programmability. As illustrated by Figure 1.1, the decoupling of data management layer and control management layer [2, 3] by SDN reduces the complexity of network management and makes it possible for revolutionary innovation and transformation through the programmability of network interfaces [4–6]. Furthermore, the SDN control plane gives a centralized view of the distributed network, enabling for more efficient network service orchestration and automation. SDN can predict future service demands and assign resources in advance, unlike previous protocols that can only react after services have been built. Furthermore, SDN-based network applications provide highly granular user-defined per-application traffic flow limitations [7, 8].

The SDN design has several advantages over traditional network topologies, including traffic engineering policy adjustment, optimization of live or run-time traffic, and the provision of unique services such as packets being processed differently depending on the user or application [9]. SDN and NFV operate jointly to make service providers and network operators have greater control over the networks by allowing them to start and terminate network activities and services on the fly. NFV [10] is all about the software implementation of network functions and their execution on non-specialized, shared hardware while decreasing CAPEX and OPEX.

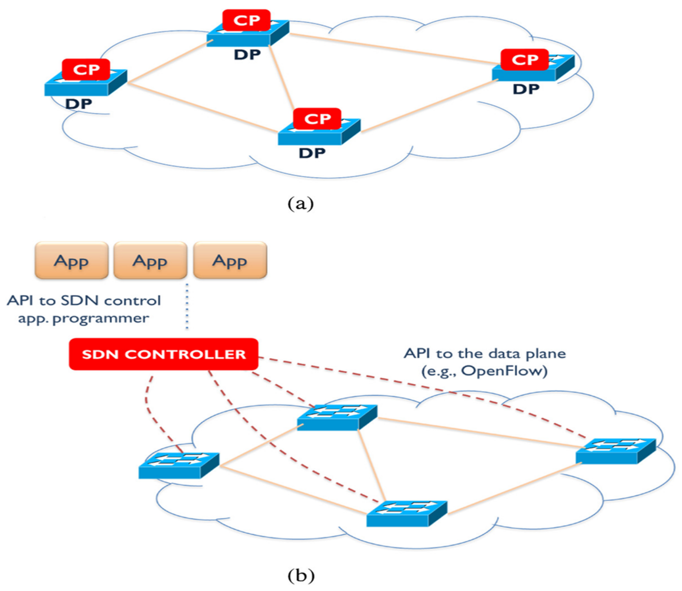

CP
DP
(a)
App
App
App
API to SDN control app. programmer
SDN CONTROLLER
API to the data plane (e.g., OpenFlow)
(b)

FIGURE 1.1: (a) Traditional networking (b) Software-defined networking

1.1 Wireless Mesh Networks

Radio nodes are grouped in a mesh topology to form a Wireless Mesh Network (WMN), these nodes include mesh clients, mesh routers, and gateways (as shown by Figure 1.2). WMN does not support frequent mobility, as it causes spending more time updating routes than providing data. The topology of a wireless mesh network is more static, allowing for route computation and data delivery to their destinations. As a result, this is a centralized low-mobility wireless ad hoc network. It is also not a totally all-wireless ad hoc network because it sometimes relies on static nodes to act as gateways. Laptops, cell phones, and other wireless devices are used as mesh clients. Mesh routers route traffic to and from gateways, which may or may not be online. A mesh cloud is the coverage area of all radio nodes acting as a single network. Access to this mesh cloud is dependent on radio nodes collaborating to form a radio network. When one node ceases to function, the other nodes can still communicate with one another, either directly or via one or more intermediary nodes that makes mesh network both dependable and redundant.

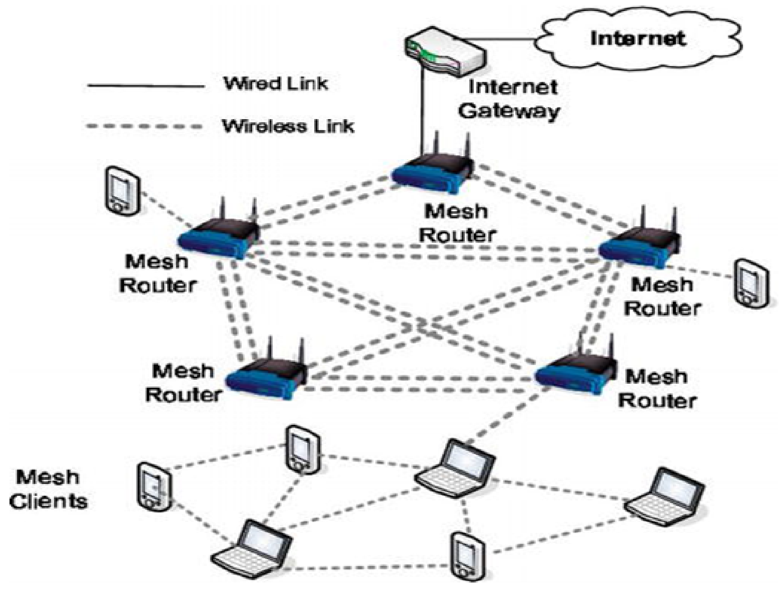


FIGURE 1.2: Wireless Mesh Network

1.2 Challenges in Wireless Mesh Networks

In today's communication systems and internet access applications due to its architecture, WMNs are progressively getting more significance. WMN features a long-standing and reliable network architecture that includes Adhoc networks, MANETs, and other wireless systems. WMN is intrinsically adaptable due to the variety of network nodes it contains, such as switches, cell- phones, and routers. WMNs, on the other hand, are difficult to manage because the addition and removal of nodes from the network can cause dynamic topology changes and a wide range of communication requirements [11]. Congestion has worsened as a result of the WMN's insufficient number of gates. WMNs must use appropriate resource allocation to overcome these issues along with other main challenges like traffic engineering and load balancing [12].

In addition, a multi-hop environment also affects network performance and causes packet loss [13]. Other considerations include variable connection quality, network asymmetry, and traffic load [14, 15]. Furthermore, any changes to WMN network nodes after they have been created and deployed may necessitate the replacement of all or part of the hardware equipment, resulting in a large rise in operational costs. WMNs confront a range of routing challenges, including fluctuating connection quality, network heterogeneity, and traffic load [16], in addition to the aforementioned network management requirements. While multi-hop WMNs remain a popular internet access option, their effectiveness is somewhat dependent on strong routing protocols. In addition, installing a large number of different devices, especially in a WMN, results in security weaknesses in wireless channels, bandwidth limitations, link and device instability, power limitations, as well as restrictions in backhaul, all of which impair performance. Furthermore, the preceding has an effect on data availability and network dependability, both of which are important factors in remote locations. As a result, in order to maximize bandwidth usage, minimize delays in packet delivery, reduced costs to access data, and enabling of throughput improvement under a range of applications/services, the protocols used must be adaptive to topology changes. Furthermore, routing protocols in these situations must have a high rate of convergence, low overhead, and low latency [17]. As a result of making WMN programmable, software-based adjustments may be possible.

1.3 Software Defined Network

The network administration (control) plane is split from the data plane in Software Defined Networking (SDN) [18], a new networking paradigm. A centralised controller is widely used to control routers/switches [19, 20]. In SDN paradigm, it is the responsibility of controller to make sure the network devices install forwarding rules to manage users' created traffic flows [21]. Using a common programming interface, such as an OpenFlow-based interface, the switches monitor and route network traffic flows according to the controller's policies [22–26]. Network control and management are made easier by separating the network control and data planes [27, 28].

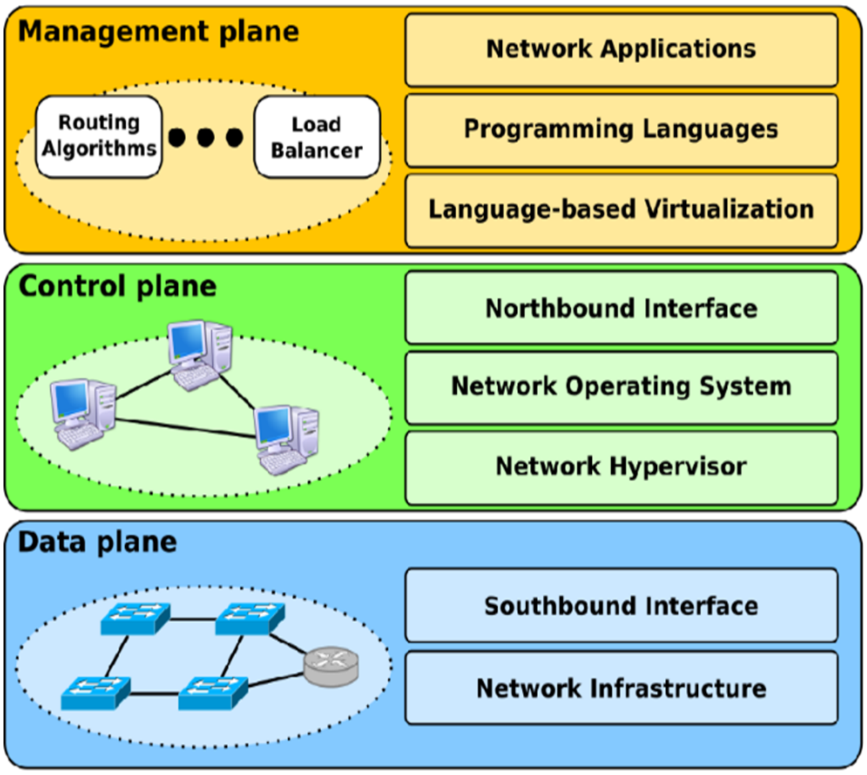


Figure 1.3: Architecture of Software-Defined Networking (SDN)

The three fundamental layers of the SDN architecture are the application layer, control plane layer, and data plane layer, as shown in Figure 1.3. Network operations and programs that are regularly used by organizations include security systems, firewalls, and load balancing. The logically centralized SDN controller is represented by the SDN brain, a control plane layer. The policies are managed by the controller, who is based on a computer. The data plane layer is made up of physical devices like routers, nodes, and other network devices. To construct a three-layer relationship, applications programming interfaces (APIs), also known as northbound and southbound APIs, are used. Northbound APIs, such as RESTful [29], link to the controller, whereas southbound APIs, such as OpenFlow, connect to the controller and data plane devices. Some of the most popular controllers include OpenDaylight, Floodlight, Ryu, and POX [30]. When an SDN node (using the OpenFlow protocol) gets the first packet of a flow, it asks the controller for the flow's forward route, and the controller installs the required rules in the node firmware. These rules, as shown in Figure 1.4, detail the behavior required by the packet [31]. The controller is in charge of developing the rules that each node in the forwarding chain need. An originating node that does not have a path associated with the accompanying packet can send a route request message [32].

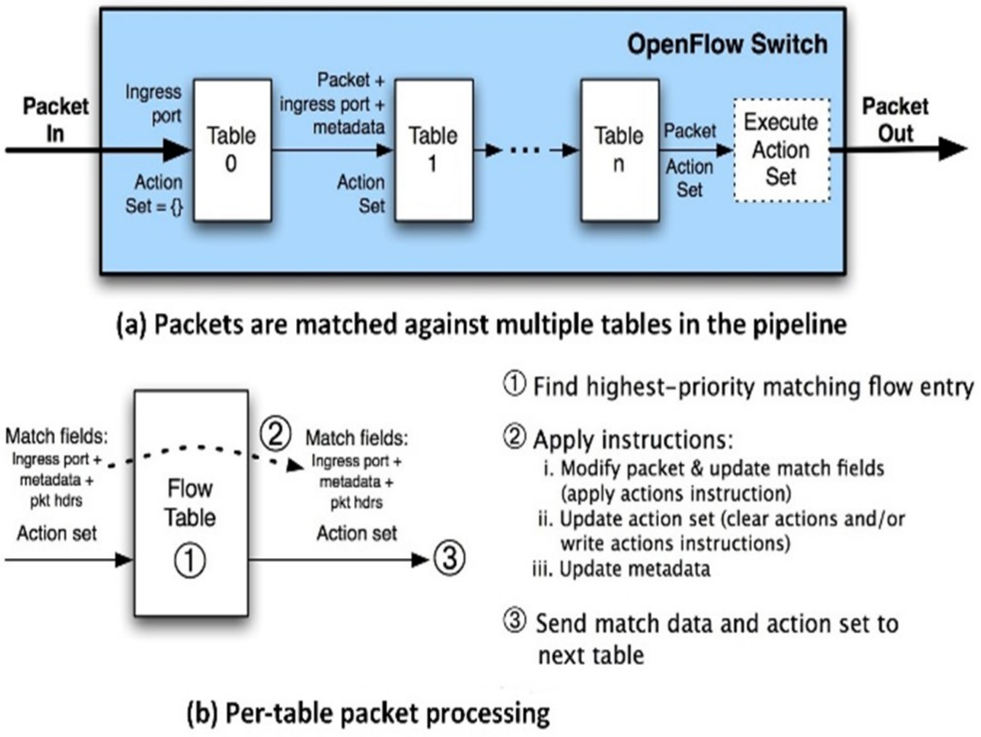

OpenFlow Switch
Packet In
Ingress port
Action Set = {}
Table 0
Packet + ingress port + metadata
Action Set
Table 1
Table n
Packet
Action Set
Execute Action Set
Packet Out
(a) Packets are matched against multiple tables in the pipeline
Match fields: Ingress port + metadata + pkt hdrs
Action set
Flow Table ①
② Match fields: Ingress port + metadata + pkt hdrs
Action set
③
① Find highest-priority matching flow entry
② Apply instructions:
i. Modify packet & update match fields (apply actions instruction)
ii. Update action set (clear actions and/or write actions instructions)
iii. Update metadata
③ Send match data and action set to next table
(b) Per-table packet processing

FIGURE 1.4: Packet Processing in SDNs

1.4 SDN in WMNs: Challenges and Solutions

Existing research focuses on making a complete paradigm change from conventional networks to software-defined networking, taking into consideration OpenFlow protocol extensions and alterations. Embedding SDN technology into the WMN networking paradigm is the main focus of present research that results in developing a hybrid network based on SDN-enabled nodes and conventional nodes in the network architecture. As a new and promising architecture, SDN adds agility and flexibility to WMNs, addressing the aforementioned routing and management concerns [33]. By remotely controlling and configuring mesh routers and converting them into simple data forwarders, SDN achieves the goal of programmable WMNs. Traffic management in WMNs can also be improved by using congestion control and load balancing algorithms. Using a single controller to implement SDN, on the other hand, could compromise network dependability. When changing the SDN paradigm, fault tolerance must be taken into account [34]. Depending on the user and system parameters, network operators must apply different QoS policies [35]. Network efficiency significantly rises as a result of integrating SDN into the WMN, and traffic engineering also gets more effective [36]. On the other hand, rip-and-replace is not a practical approach for the widespread adoption of any contemporary networking technology. Network operators are not expected to update their equipment when transitioning to a modern technological paradigm [37]. Full adoption of SDN is not feasible due to certain economic and operational constraints, such as support for all network nodes and management by a logically centralized controller. As a result, a gradual hybrid deployment of SDN and WMN is required [38].

Additionally, the management of data, processing capabilities of wireless nodes and speed of media are becoming increasingly important to networks. Furthermore, challenges arise when SDN routing is used in conjunction with existing protocols. When only a few wireless nodes are activated in a single network, multi-hop link-layer routing issues, such as MAC layer-based routing, exist. When implementing data processing capabilities like filtering and routing, which may be done using a variety of protocols, adopting OpenFlow nodes instead of legacy nodes gives you greater flexibility. This adaptability would aid in improving traffic engineering, mobility management, a more widespread efficient use of WMN communication resources as well as advanced routing [39]. WMN-integrated OpenFlow makes network administration easier by using a centralized approach to run and control the logical issues of network by processing actions according to the criteria that has matched for network nodes implemented with OpenFlow. We must overcome some WMN-related challenges to achieve these benefits, such as wireless channel unreliability that can temporarily disrupt controller communications, unavailability of auto learning mechanism, spanning tree protocol or other commonly used layer 2 protocols to support switch-controller communication for switching purpose [40].

1.5 Advantages of Hybrid WMN and SDN

SDN networks have a number of advantages over traditional networks, including easier network management and security policy implementation [41]. For a variety of reasons, SDN is rarely fully implemented in networks. One of the main problems is a lack of CAPEX and OPEX for a new network infrastructure. Businesses are sometimes hesitant to invest major resources in constructing a new network infrastructure from the ground up, and they are also concerned about downtime during the SDN transition. Installing a few SDN-capable devices [42] alongside the network devices with conventional protocols and gradually, incrementally substituting traditional devices with SDN-enabled devices, as depicted in Figure **??**, is one approach to solving these problems. The following are some of the benefits of integrating SDN with traditional networks:

1. In terms of Capital expenditures (CAPEX) and Operational Expenditures (OPEX), migrating from conventional network to SDN is significantly expensive. To replace all current traditional devices with SDN devices, a large investment in SDN device acquisitions is required. After full deployment of SDN and the building of a pure SDN network, more budgetary requirements are necessary for training the operators to design, develop, and to realize the SDN network [42]. Such restrictions can be alleviated by partial deployment.

2. With hybrid network based on SDN-enabled devices, we can make use of advantages of SDN, , a new promising paradigm, that does not commit to complete network transitioning to SDN. For instance, Thousands of hundreds forwarding entries are controlled and monitored by ISPs, , however, generally thousands of forwarding entries are managed by SDN-enabled devices [43]. As a result, the ISP may manage these forwarding entries using

hybrid network based on SDN-enabled devices in the access network while also gaining from SDN in the distribution network [44].

3. Data traffic flows can be controlled precisely with SDN. A hybrid network based on SDN-enabled devices can be created if only a small piece of the network needs fine-grained management by adopting SDN for that segment and relying on conventional networking for the rest [45].

4. For specialised services such as controlling the functionality of forwarding devices and various network domains by SDN controller, traditional routing protocols are especially helpful. In order to make SDN controller free from those responsibilities, that are successfully managed by traditional routing protocols, a hybrid SDN network based on SDN-enabled devices can be created [46].

5. A software-defined network (SDN) architecture is an idea that is still in its infancy. As a result, SDN equipment is less developed than conventional networking equipment. Network administrators could be reluctant to completely upgrade the outdated network as a result. Hybrid network based on SDN-enabled devices make it easier to transition from traditional network devices to SDN-enabled devices. A network operator would be able to add gradually more SDN hardware and evaluate SDN functionality in practical situations. For instance, Google [47] has employed SDN for the management of network and control functions for a number of years [48]. Future applications of SDN technology could include both small and medium-sized businesses and larger significant organizations.

1.6 Problem Statement

Transitioning from wireless mesh networks paradigms to software defined networking has remained a challenge for academia and the research community to date. Since the launch of SDN and OpenFlow technology, adoption has been gradual, particularly among smaller businesses with less resources and networks. Surprisingly, the biggest difficulty with SDN is that it has just a small number of applications in the networking business. SDN is approached in a variety of ways by different vendors, ranging from hardware-centric models and virtualization platforms to hyper-converged networking and controller-less solutions. Backward compatibility, interoperability, and control packet management are the primary obstacles for SDN deployment in existing wireless networks. Coexistence of legacy nodes in the hybrid architecture results in a high number of control packets being managed in the backbone network, putting an excessive strain on the controller that can be alleviated by making forwarding devices intelligent or smart. Furthermore, network management is often handled at two levels: (1) real-time network traffic, and (2) network parameter adjustment and both of these should be done via centralized control. The major goal of this research is to propose routing architecture for hybrid networks comprising of conventional and SDN-enabled hybrid nodes and to suggest changes to the SDN switch architecture that would allow it to work over wireless mesh networks.

The proposed hybrid architecture could be a better solution to the problems outlined above. The major goal of this research is to suggest techniques and changes to the SDN switch architecture that would allow it to work over wireless mesh networks. The research will focus on building mechanisms for controlling legacy nodes via controllers in a hybrid environment, which will necessitate the creation of new WMN-specific rules, actions, and commands. Examples of network functions include traffic engineering, network monitoring, load balancing, and routing.

1.7 Aim and Objectives

The goal of this study is to examine existing routing methodologies in hybrid SDN and WMNs and propose a methodology for improving routing protocol performance so that improved traffic management and monitoring can be accomplished more quickly and cost-effectively. Our study's key goals are as follows:

- To look into the existing routing methods for wireless mesh networks and elicit the best features for use in the proposed technique

- To improve the performance of the WMNs routing design as well as to improve its network monitoring framework

- To propose a reliable routing architecture for a hybrid topology that includes SDN hybrid nodes as well as legacy nodes

- In the event of a controller failure or an unreachable controller, a routing method will be proposed

- The architecture of SDN nodes must be changed in order to create a Smart SDN hybrid node

- To develop a mechanism for adaptive network monitoring and traffic measurement

1.8 Dissertation Goals and Contribution

To improve the performance of wireless mesh networks and enable smooth interoperability between legacy mesh nodes and SDN nodes, we offer a robust routing architecture for hybrid Wireless Mesh and Software Defined Networks. The suggested routing approach alters the SDN node architecture by combining IP-based forwarding with OLSR routing and SDN forwarding with the OpenFlow protocol to create a hybrid and coexisting architecture. It should be emphasized that our research does not endorse the SDN method as a replacement for existing routing approaches. The purpose of this research is to understand that an SDN-based architecture can help with WMN routing to improve its performance rather than arbitrarily equating legacy routing approaches that use an in-band technique of signaling with the centralized approach that employs out-of-band technique of signaling. It is explored the legacy protocols can be used with SDN networking architecture and to what extent the former can help the latter. In addition, as compared to previous hybrid routing architectures, the proposed routing architecture offers a cost-effective solution and seamless compatibility between conventional and SDN-enabled devices. The following are some of the research's major contributions:

1. A routing architecture for hybrid networks comprising of conventional mesh nodes and SDN-enabled hybrid nodes while providing robustness.
2. The architectural modification of SDN nodes to create an SDN hybrid node that allows the data plane to adapt to network topology changes without the controller having to query each time a flow is added to the network.
3. An adaptive network monitoring enables a network administrator to monitor network traffic as well as link data such as link utilization and bandwidth.

1.9 Methodology and Techniques

The methodology used for conducting the proposed research and its simulation is shown in Figure 1.5. The details are as given as:

1. Design of a network model while taking some assumptions such as mobility in the network, diversified number of flows etc. under consideration.
2. Compare and analyze existing routing architecture in WMN and SDN networking domain.
3. Propose the architecture that include hybrid routing, modification of SDN switch architecture, network monitoring mechanism, and routing in case of controller failure or unreachable.
4. Evaluate the proposed architecture using simulation tools and then validating the results by performing a thorough analysis.

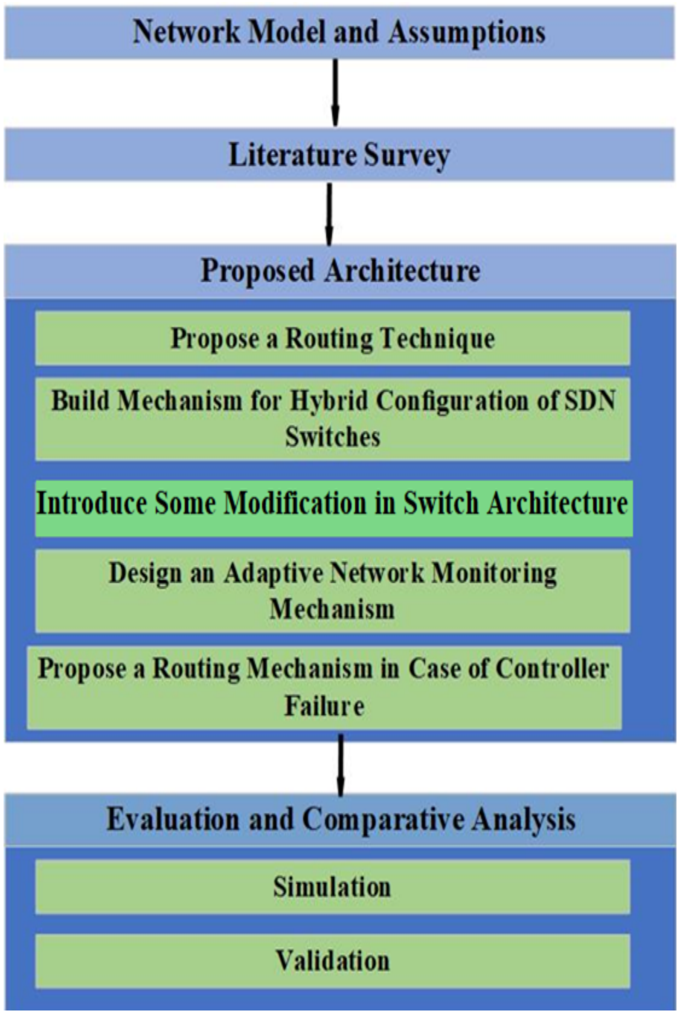

Network Model and Assumptions
Literature Survey
Proposed Architecture
Propose a Routing Technique
Build Mechanism for Hybrid Configuration of SDN Switches
Introduce Some Modification in Switch Architecture
Design an Adaptive Network Monitoring Mechanism
Propose a Routing Mechanism in Case of Controller Failure
Evaluation and Comparative Analysis
Simulation
Validation

FIGURE 1.5: Methodology of Proposed Architecture

1.10 Organization of this Dissertation

The structure of this dissertation is organized in the following way: Chapter 2 provides the background for proposed routing architecture whereas Chapter 3 investigates the related research and implementations on existing hybrid SDN/IP routing solutions. A robust routing solution for hybrid WMN and SDN is proposed in Chapter 4, while recovering from controller failure, a routing approach is also described to eliminate the possibility of a single point of failure. This chapter also explains the simulation model as well as analysis of results achieved. An adaptive network monitoring technique and analysis of its results are presented in Chapter 5. Chapter 6 concludes the dissertation while describing its future work as well.

Chapter 2

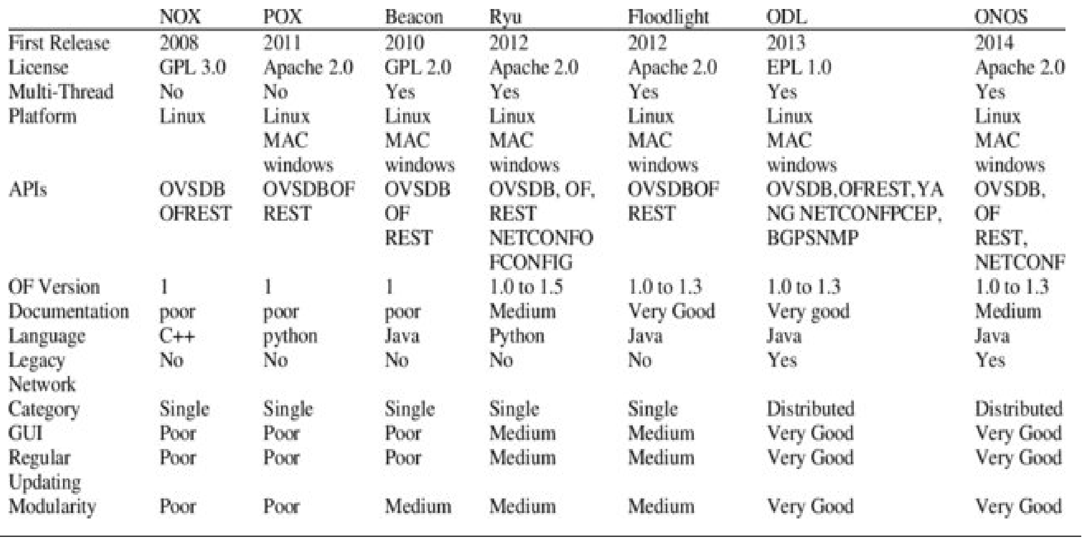
Background for Proposed Routing Architecture

The proposed routing architecture makes use of available tools and technologies for its simulation purpose, that are discussed in details in subsequent sections.

2.1 SDN Controller - OpenDaylight

The SDN controller serves as a foundation for implementing the proposed architecture "Soft-Mesh". Various SDN controller platforms are available, including NOX, POX, Beacon, Ryu, Floodlight, ONOS, and OpenDaylight (ODL) [49]. Table 2.1 summarizes the main features of above mentioned SDN controllers. The given comparison provides us with two main opensource choices, ONOS and ODL. The goal of ODL is to connect legacy (such as BGP, SNMP, etc.) and NGN (such as OpenFlow and SDN) networks. In general, Carriers are more interested in ONOS since it places a greater emphasis on performance factors and clustering to boost availability and scalability. As a result, ONOS places a greater emphasis on carrier-grade networks, and telcos participate in their projects. ODL features more vendors than ONOS, including Cisco, Juniper, and NES. Therefore, for the purpose of implementation, we choose OpenDaylight SDN Controller, and network nodes including SDN hybrid and legacy nodes. OpenFlow protocol is used as a southbound API to create communication between control plane and data plane. OpenDaylight (ODL) is a Java-based open-source project that has the backing of a number of important networking firms, including IBM, Cisco, Juniper, and VMWare. As a result, it can be used on any hardware and operating system platform that supports Java [50].

TABLE 2.1: Comparison of SDN Controllers



	NOX	POX	Beacon	Ryu	Floodlight	ODL	ONOS
First Release	2008	2011	2010	2012	2012	2013	2014
License	GPL 3.0	Apache 2.0	GPL 2.0	Apache 2.0	Apache 2.0	EPL 1.0	Apache 2.0
Multi-Thread	No	No	Yes	Yes	Yes	Yes	Yes
Platform	Linux	Linux MAC windows	Linux MAC windows	Linux MAC windows	Linux MAC windows	Linux MAC windows	Linux MAC windows
APIs	OVSDB OFREST	OVSDBOF REST	OVSDB OF REST	OVSDB, OF, REST NETCONFO FCONFIG	OVSDBOF REST	OVSDB,OFREST,YA NG NETCONFPCEP, BGPSNMP	OVSDB, OF REST, NETCONF
OF Version	1	1	1	1.0 to 1.5	1.0 to 1.3	1.0 to 1.3	1.0 to 1.3
Documentation	poor	poor	poor	Medium	Very Good	Very good	Medium
Language	C++	python	Java	Python	Java	Java	Java
Legacy Network	No	No	No	No	No	Yes	Yes
Category	Single	Single	Single	Single	Single	Distributed	Distributed
GUI	Poor	Poor	Poor	Medium	Medium	Very Good	Very Good
Regular Updating	Poor	Poor	Poor	Medium	Medium	Very Good	Very Good
Modularity	Poor	Poor	Medium	Medium	Medium	Very Good	Very Good

The design of the OpenDaylight SDN Controller is shown in Figure 2.1. The OpenDaylight SDN Controller has several layers. The higher layer is composed of apps with business and network logic. The architecture of the middle layer may contain SDN abstractions. This layer hosts both north-bound and south-bound APIs. The northbound APIs that the controller exposes are used by applications. OpenDaylight supports both bidirectional REST and the Open Service Gateway Initiative (OSGi)1 framework for the northbound API. The business logic is contained in the applications that are above the middle layer. These apps acquire network intelligence, execute analytics algorithms, and then orchestrate any new rules across the network via the controller. The bottom layer, however, is made up of both actual and virtual devices [51]. Figure 2.2 illustrates the message exchange between OpenDaylight controller and OpenFlow Switch.

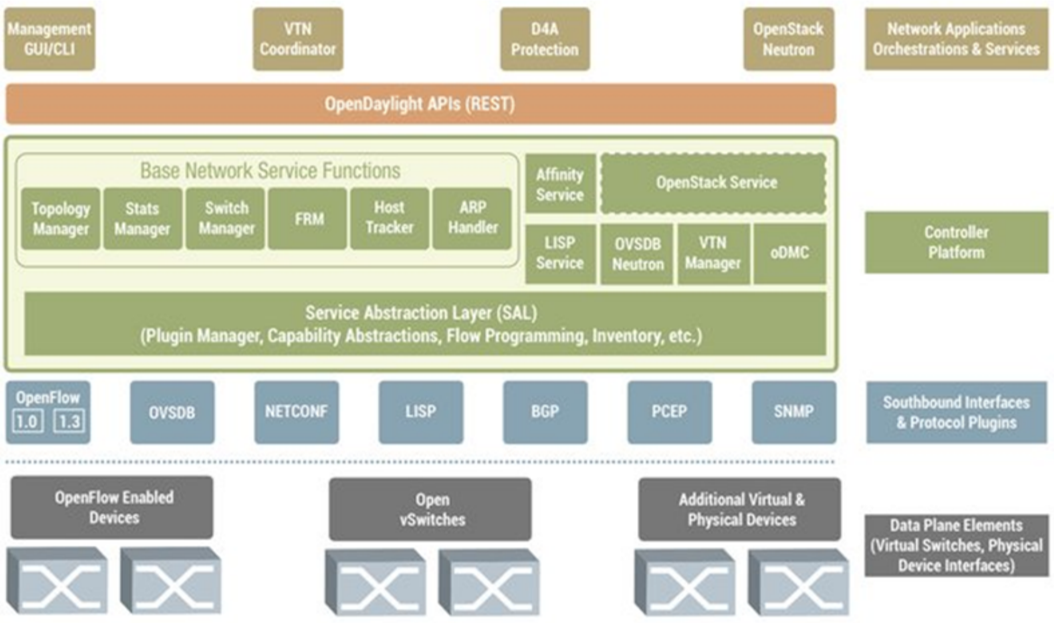


FIGURE 2.1: OpenDaylight SDN Controller Architecture [52]

2.2 OpenFlow Switch – Architecture and Operations

A communications protocol known as OpenFlow allows the control plane to decouple from the forwarding plane of many devices and communicate with it from a single location, increasing functionality and programmability [53]. Figure 2.2 illustrates the components of an OpenFlow logical switch, which include one or more flow tables, a group table, and one or more OpenFlow channels to an external controller. These tables execute packet lookups and forwarding.

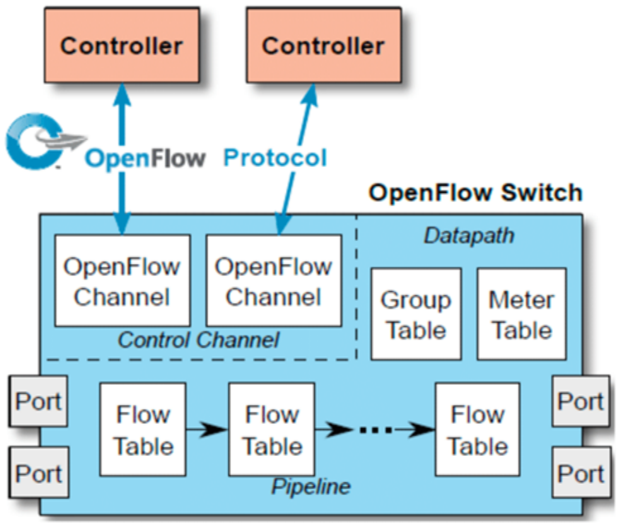


FIGURE 2.2: OpenFlow Switch Architecture [54]

Using the OpenFlow protocol, the switch and controller exchange information, and the controller controls the switch, as shown by Figure 2.3. The controller can create, update, and remove flow entries from flow tables in the both proactive and reactive manner.

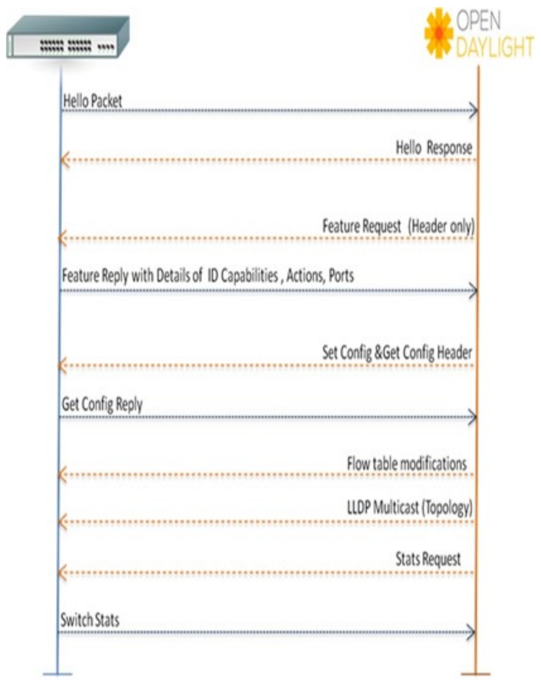

OPEN DAYLIGHT
Hello Packet
Hello Response
Feature Request (Header only)
Feature Reply with Details of ID Capabilities, Actions, Ports
Set Config &Get Config Header
Get Config Reply
Flow table modifications
LLDP Multicast (Topology)
Stats Request
Switch Stats

FIGURE 2.3: Message Exchange between OpenDaylight Controller and OpenFlow Switch

There is an OpenFlow Pipeline contains multiple flow table, and is used for processing the packets to interact with other flow tables as shown in Figure 2.4.

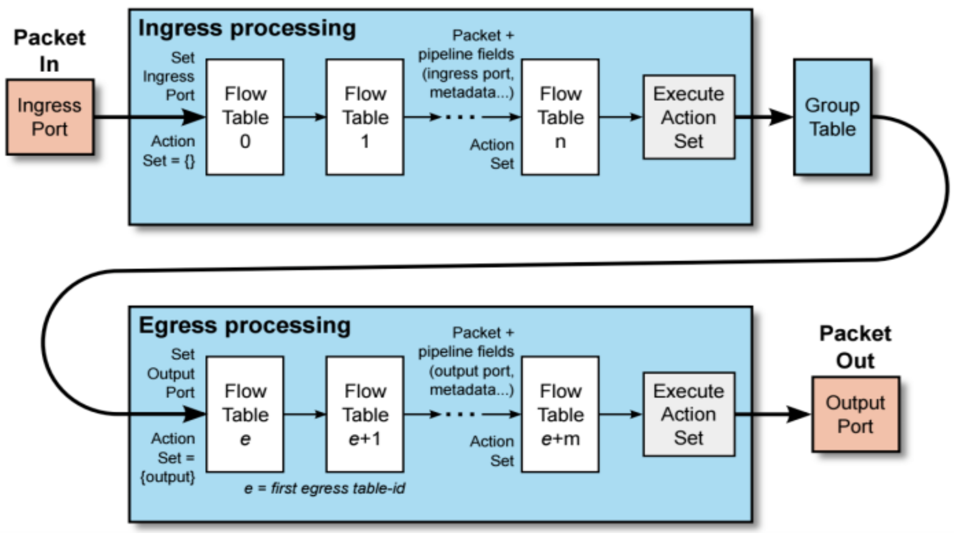


FIGURE 2.4: Packet flow in Pipeline Processing

An OpenFlow Switch executes the actions depicted in Figure 2.5 upon receiving a packet. Based on pipeline processing, the switch may do table lookups in other flow tables after beginning with the initial flow table. The OpenFlow protocol supports the message types including, 1) controller-to-switch, 2) asynchronous, and 3) symmetric. Each message type has its own subtypes. Table 2.2 presents the type of messages used by OpenFlow protocol.

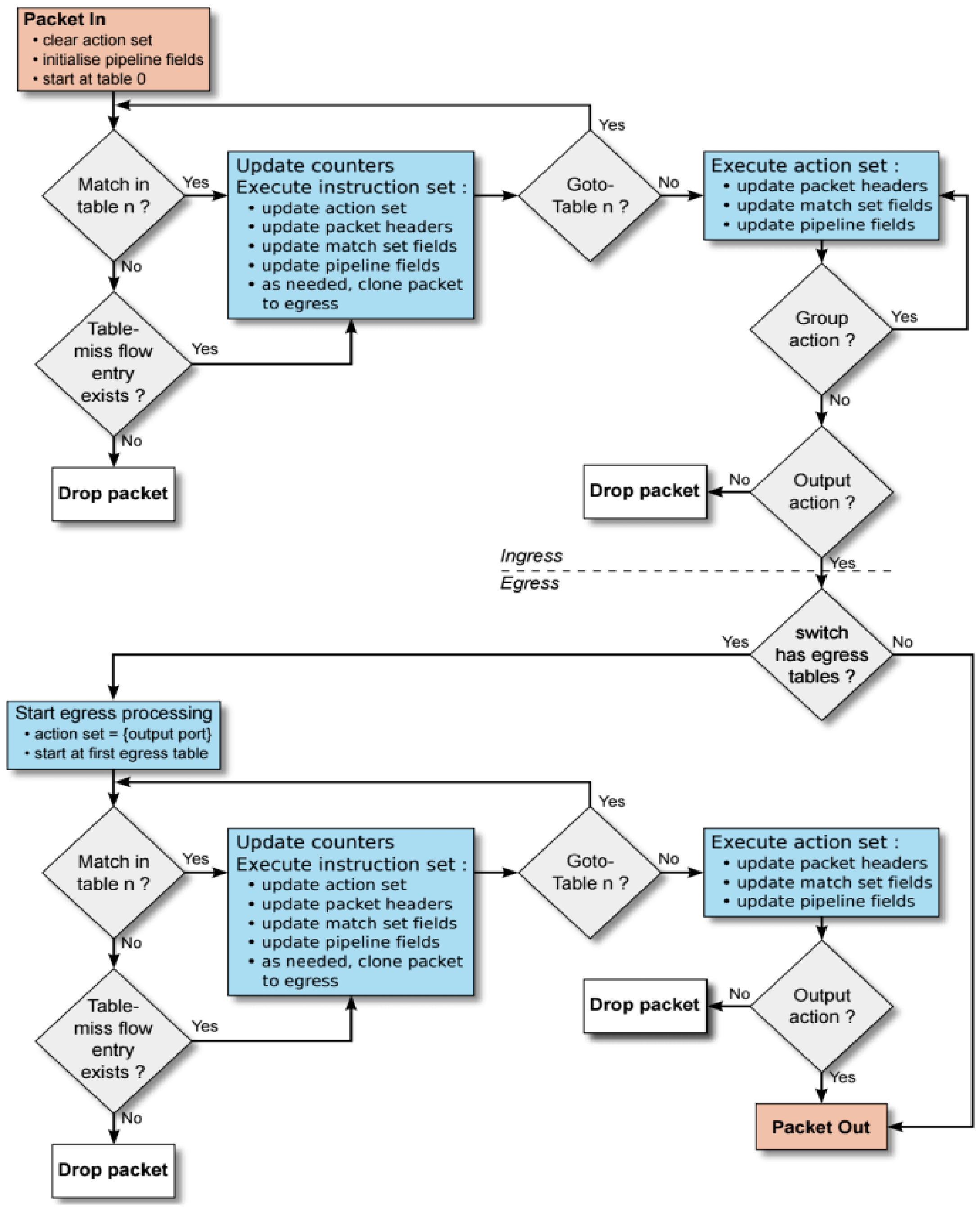


FIGURE 2.5: Flowchart of packet flow through an OpenFlow switch

TABLE 2.2: OpenFlow Protocol Messages

Message Type	**Message**	**Description**
Controller-to-Switch Messages	Features Request/Reply	Controller may request capabilities of switch by sending a feature request message to the switch. In response, the switch must reply with features reply message specifying the features and capabilities of the switch
	Configuration	The controller is able to set and query configuration parameters in the switch.
	Modify-State	These messages are sent by the controller to manage the state of the switches. They are used to add/delete or modify flow table entries or to set switch port priorities.
	Read-States	They collect statistics from the switch flow tables, ports, and the individual flow entries.
	Packet Outs	These messages send packets out of a specified port on the switch and forward the packets received via packet-in messages.
	Barrier	They ensure that message dependencies have been met. They receive notifications for completed operations.
	Role-Request	They set the role of their OpenFlow channel & also query that role
	Asynchronous-Configuration	These messages set an additional filter on asynchronous messages that it wants to receive/query. It is used when the switch connects to multiple controllers.
Asynchronous Messages	Packet-in	For all packets that do not have a matching flow entry or if a packet matches an entry with a send to controller action, a packet-in message is sent to the controller. t transfers the control of the packet to the controller.
	Flow-Removal	This message informs controller about removal of a flow entry from a flow table. The flow modify message also specifies whether the switch should send a flow removal message to the controller when the flow expires.
	Port-status	It informs the controller of a change on the port. These events include changes in port status (for example, disabled by the user) or a change in the port status as specified by 802.1D (Spanning Tree). Switch is expected to send port status to controller via the port-update messages.

Message Type	**Message**	**Description**
	Error	The switch is able to notify the controller of problems using error message.
Symmetric Messages	Hello	Hello messages are exchanged between the switch and controller upon connection setup.
	Echo	These messages can be used to indicate the latency, bandwidth, and/or liveliness of a controller-switch connection.
	Vendor	These messages offer additional functionality within the OpenFlow message type space for future revisions of OpenFlow.

When an OpenFlow switch gets a packet that cannot be matched to an existing flow rule, the packet will be buffered by switch before asking the controller for a new flow rule with an *OFPT_PACKET_IN* message also includes data packet header fields. The controller then responds with an *OFPT_FLOW_MOD* message that specifies the action to be taken on the data packet and the amount of time for which to keep the flow rule in its flow table. A timeout is used to describe this interval. Each flow rule has two associated timeout values: a soft timeout for idle periods of inactivity known as idle timeout value, which is triggered when the flow is inactive, and a hard timeout, which is triggered when the timer expires, as illustrated by Figure 2.6. When any of these durations expires, the switch deletes the associated flow entry from its flow table and notifies the controller with an *OFPT_FLOW_REMOVED* message.

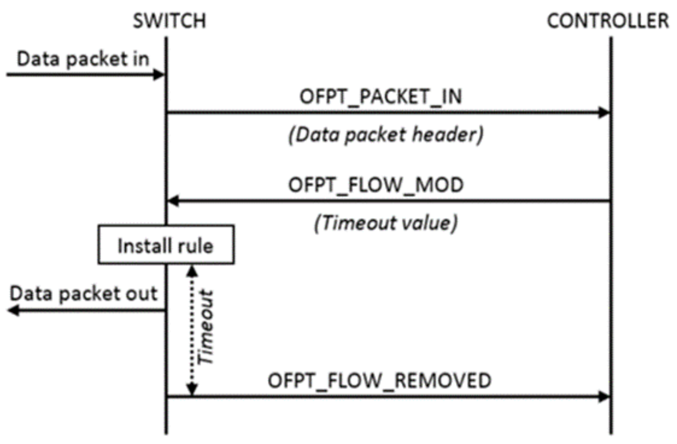


Figure 2.6: Rule installation in OpenFlow Switch

The OpenFlow specification requires the switch to send the whole packet to the controller wrapped within the *OFPT_PACKET_IN* message, except in cases where the switch lacks buffering capability or the input buffer is full. A message called *OFPT_PACKET_OUT*, which also comprises the whole data packet, is the controller's response in this situation. The switch merely executes the related action without installing any flow rules, as shown in Figure 2.7.

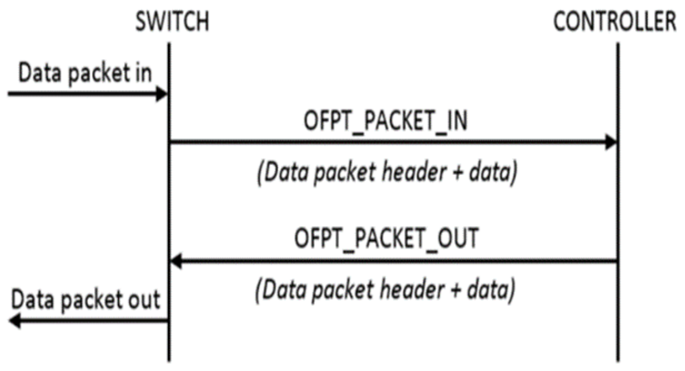


FIGURE 2.7: OpenFlow Switch cannot buffer packets

If the flow table of switch is full, then upon receiving an instruction to install a flow rule, it sends an *OFPT_ERROR* message to the controller with error code *OFPFMFC_TABLE_FULL*, the packet is dropped then, as shown by the Figure 2.8.

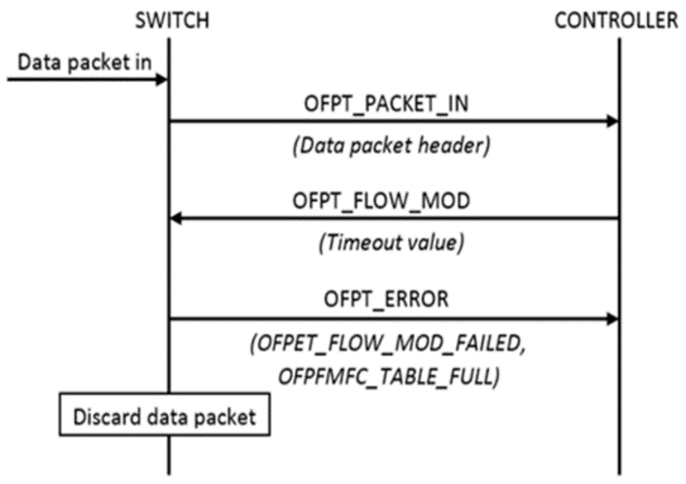


FIGURE 2.8: Flow table is full in OpenFlow Switch

2.3 Simulation Environment

The simulation environment of our proposed architecture uses the following components:

2.3.1 Mininet-Wifi

Based on the 802.11 hwsim wireless simulation driver and the common Linux wireless drivers, the network emulator Mininet-WiFi [55] uses virtualized WiFi Stations and Access Points. It also imitates the features of mobile stations, such as movement and location in relation to access points. Mininet-WiFi gives all the common SDN emulation skills by introducing extra features. In addition, it utilizes a few programs, including iw, iwconfig, and wpa supplicant. The first two are used for wireless interface configuration and data collection, while the third is utilized with Hostapd, among other things, implement WPA (Wi-Fi Protected Access). They are both necessary tools, and TC (Traffic Control) is an additional one. The rate, delay, latency, and loss parameters of the Linux kernel packet scheduler are set using this. It could imitate real-world behavior more faithfully by applying these settings to the virtual interfaces of stations and APs. The architecture of Mininet- WiFi is depicted in Figure 2.9.

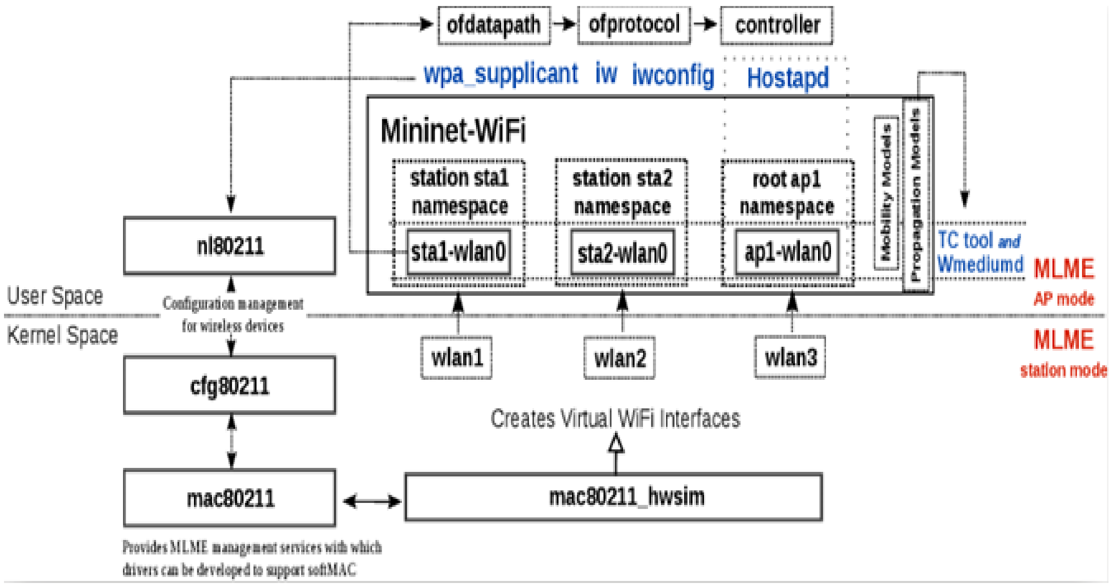


FIGURE 2.9: Mininet-Wifi Architecture [56]

2.3.2 Networks Namespace (NNS)

A network namespace is a separate replica of the network stack with its own network devices, routes, and firewall rules. In other words, each NNS creates an abstraction layer that creates segregated network resources for processes within a certain user area [57].

2.3.3 Open vSwitch (OVS)

A virtual switch that supports OpenFlow is called Open vSwitch (OVS). A physical switch's operation is comparable to this one. Physical network adapters and several virtual network adapters are connected to a virtual switch's two ends. The virtual switch searches for MAC addresses and keeps a mapping database. To finish data forwarding, locate the necessary virtual machine link [58]. Figure 2.10 demonstrates the architecture of Open vSwitch (OVS).

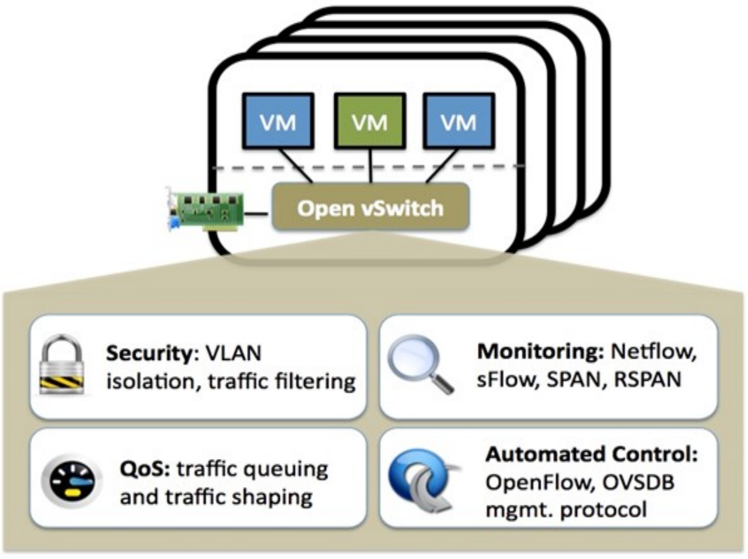

VM
VM
VM
Open vSwitch
Security: VLAN isolation, traffic filtering
Monitoring: Netflow, sFlow, SPAN, RSPAN
QoS: traffic queuing and traffic shaping
Automated Control: OpenFlow, OVSDB mgmt. protocol

FIGURE 2.10: Architecture of Open vSwitch (OVS) [59]

Chapter 3

Hybrid Wireless Mesh and Software Defined Networks: Review of Existing Approaches

3.1 Introduction

WMN features a long-standing and reliable network architecture and is intrinsically adaptable due to the variety of network nodes it contains, as mentioned in section 1.1. On the other hand, WMNs are difficult to manage because the addition and removal of nodes from the network can cause dynamic topology changes and a wide range of communication and routing requirements. Table 3.1 summarizes the existing routing strategies used in WMNs.

Existing SDN research focuses on a complete paradigm change from traditional networks to software-defined networking, taking into account OpenFlow protocol extensions and alterations. Rip-and-replace, on the other hand, is not a viable strategy for mass adoption of any modern networking technology. Network operators are not expected to update their equipment when transitioning to a new technical paradigm.

Table 3.1: Existing Routing Protocols of Wireless Mesh Networks

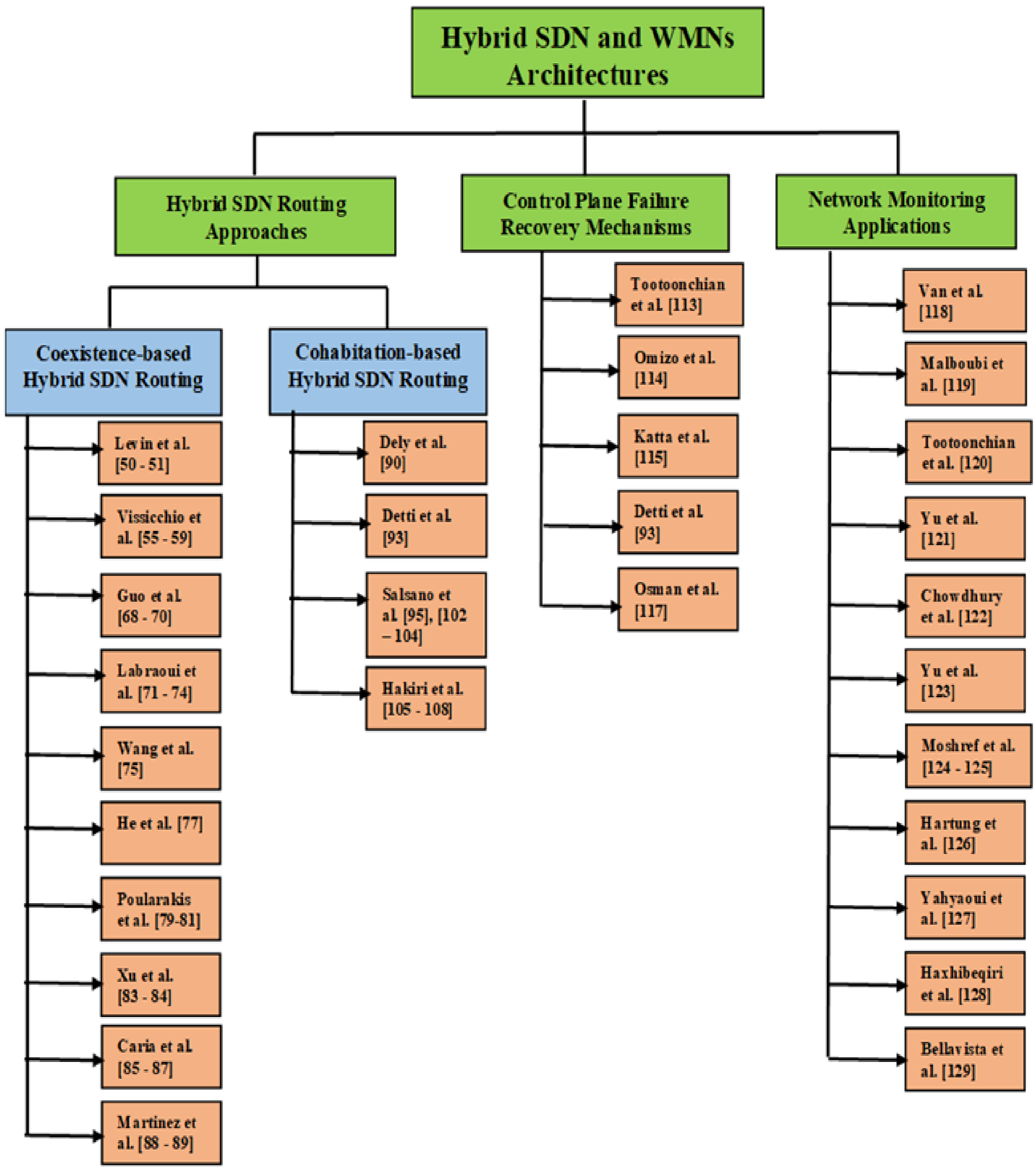
Protocols	**Advantages**	**Limitations**
AODV	Require less amount of storage space due to expiry of unused routing information, Reduces energy consumption, Support multicasting, Small routing table, Quick recovery	Lacks an efficient route maintenance technique. High route discovery latency. Increase in control overheads due to multiple route reply messages. No aggregate routing
DSDV	Sequence number ensures the freshness of routing information. Incremental updates to avoid extra traffic. Maintains only the best or shortest path. Routing table is reduced	Large amount of overhead. It doesn't support multipath routing. Unnecessary advertising of routing information
DSR	Use of route cache reduces route discovery overheads. Supports multipath routing. No periodic beaconing or hello message exchanges. Quick path recovery	High route discovery latency. Overhead carried in the packet will continue to increase as the network diameter increases. Long response time and large packet header
OLSR	No central administration to handle routing process. Reliable for the control messages exchange Suitable for high density network. No delays in the transmission of the packets	Require greater storage capacity. High control overhead. Increasing battery depletion. Maintain information about unused routes
BATMAN	No maintenance of topology changes information. Maintains best next hop towards all other nodes Deals with unreliable links. Originator message (OGM), flooded selectively	High latency, low throughput and moderate packet loss. Not suitable for real world problems. Higher overhead due to OGM flooding.
TORA	Failure of nodes is quickly resolved. Not require a periodic update. Communication overhead minimized Bandwidth utilization minimized. Reliable in-order control packet delivery	Depends on synchronized clocks. Intermediate lower layers dependency

Full adoption of SDN is not possible due to certain economic and operational constraints, such as support for all network nodes and management by a logically centralized controller. As a result, a hybrid solution consisting of gradual and incremental transition of legacy networking to software defined networking is required. Table 3.2 compares the key features of hybrid networks and traditional networks.

Table 3.2: Comparison of Hybrid SDN Network with Traditional Networks

Features	**Hybrid SDN Network**	**Traditional Networks**
Programmability	Partial	Not at all
Protocols	OpenFlow + Traditional	Traditional
Deployment	More than Pure SDN	Common
Budget	Moderate	Very low for old deployments
Fear of Down Time	Low	Very low
Old Traditional Devices	Used with SDN Devices	Used exclusively
Network Management	SDN-like Control	Difficult
Deployment Model	Specific according to the requirements	Specific
Training Cost	Low	Low
Network Controller Overhead	Moderate as local events can be handled at network devices	No Controller
Reliability	High	High
Robustness	High	High
Scalability	Easily Scalable	Easily Scalable

The term hybrid refers to two or more layers. A hybrid SDN is a networking strategy that uses both traditional networking and SDN protocols in the same environment. Instead of passing control to the hardware, SDN sends it to a controller software program. Network engineers were able to create new SDN technologies and support switched fabrics across several vendors' hardware and application-specific integrated circuits using a hybrid SDN. This effectively allowed legacy networking devices or environments to run SDN technology, such as OpenFlow, without requiring a total infrastructure rebuild. This chapter provides a comprehensive review of existing hybrid approaches with respect to three domains that are hybrid routing approaches, control plane failure mechanisms and network monitoring applications used by existing hybrid SDN approaches as illustrated by Figure 3.1.



Hybrid SDN and WMNs Architectures
Hybrid SDN Routing Approaches
Control Plane Failure Recovery Mechanisms
Network Monitoring Applications
Coexistence-based Hybrid SDN Routing
Cohabitation-based Hybrid SDN Routing
Levin et al. [50 - 51]
Vissicchio et al. [55 - 59]
Guo et al. [68 - 70]
Labraoui et al. [71 - 74]
Wang et al. [75]
He et al. [77]
Poularakis et al. [79-81]
Xu et al. [83 - 84]
Caria et al. [85 - 87]
Martinez et al. [88 - 89]
Dely et al. [90]
Detti et al. [93]
Salsano et al. [95], [102 – 104]
Hakiri et al. [105 - 108]
Tootoonchian et al. [113]
Omizo et al. [114]
Katta et al. [115]
Detti et al. [93]
Osman et al. [117]
Van et al. [118]
Malboubi et al. [119]
Tootoonchian et al. [120]
Yu et al. [121]
Chowdhury et al. [122]
Yu et al. [123]
Moshref et al. [124 - 125]
Hartung et al. [126]
Yahyaoui et al. [127]
Haxhibeqiri et al. [128]
Bellavista et al. [129]

FIGURE 3.1: Existing Hybrid WMN and SDN Approaches

3.2 Existing Hybrid SDN Routing Approaches

This section examines numerous hybrid routing systems that integrate SDN architecture with WMNs to address the load balancing, traffic engineering, and mobility management challenges that mesh networks encounter as a result of topological changes. However, depending on the implementation frameworks employed, we divide hybrid routing strategies into two categories:

1. Coexistence-based hybrid routing
2. Cohabitation-based hybrid routing

3.2.1 Coexistence-Based Hybrid Routing

As shown in Figure 3.2, in WMN and SDN hybrid routing solutions based on coexistence of SDN-enabled devices and conventional devices, that are being deployed in a traditional topology and a hybrid network is created. By constructing such a hybrid network, there is the possibility to use existing conventional nodes, to give conventional networks a management control that exists in SDN networking technology. Such networks therefore have a variety of appealing characteristics. The proposed architecture analyses the usage of traditional routing protocols as AODV, OSPF, OLSR, and other routing protocols in legacy nodes and usage of OpenFlow in SDN nodes.

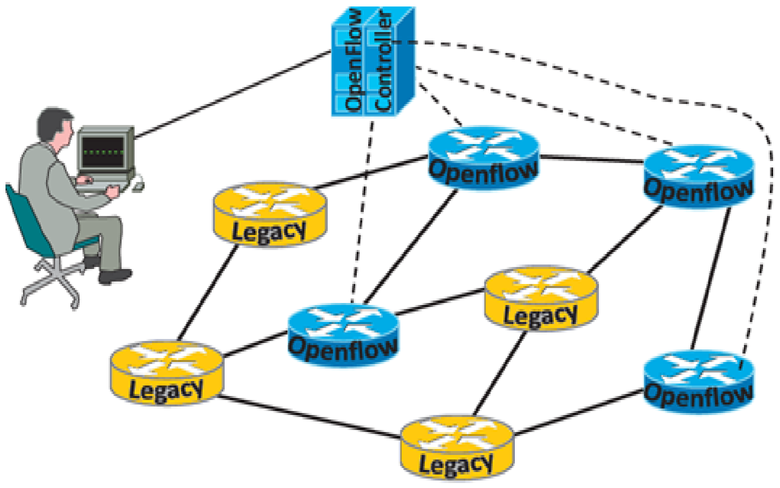


FIGURE 3.2: Coexistence-based Hybrid (SDN/IP) Routing

Levin et al. [60] proposed ”Panopticon,” a significant contribution for building hybrid network based on SDN-enabled devices. By combining traditional and SDN-based network, Panopticon provides a logical (virtual) SDN network that has the same benefits as a complete SDN network. ”Panopticon”, a terminology known for prison design with each cell arranged in a circle and is reachable from a central watchtower for management and inspection [61]. Panopticon enables the installation of all SDN network-specific applications on a physical SDN network. The waypoint enforcement technique [62],is known to be the core concept of Panopticon. Every packet crossing over the source-destination pair passes through an SDN-enabled device is required by the waypoint technique. Once the packet reaches the SDN-enabled device, it is processed by the controller. As a result, every packet is handled precisely as it would be in a network that only used SDN. Figure 3.3 (a) demonstrated an example of Panopticon architecture with logical and physical network topologies. SCTs, are overlaid to develop connectivity between the ports that are SDN-controlled SDN-enabled devices using physical architecture, the logical perspective is shown in Figure 3.3 (b). Moreover, a spanning tree is identified by using VLAN ID known as virtual local area network identification [62, 63]. Levin et al. [64, 65] studied a few usage situations and gathered experimental data to show logical SDN functions. The results of the study demonstrate that when 10% of conventional switches are replaced with SDN switches, SDN nodes manage around 25% of the traffic. The results demonstrate a 40% performance gain over the initial network.

Vissicchio et al. [65] developed an architecture to provide an alternate solution for resilience in SDN-based networks. The author proposed Interior Gateway Protocol (IGP) [66] to address the issues of recovery from network failure and robustness. Conventional protocols with a high degree of distribution can swiftly fix transient network problems. The controller configures routing patterns tin order to avoid the restrictions inherent to the scattered conventional protocols and achieves optimization of the network in the long-term perspective. Local agents are put up on each hybrid SDN switch to exchange routing information, and distributed classical IGPs are used to construct backup routes. In case of

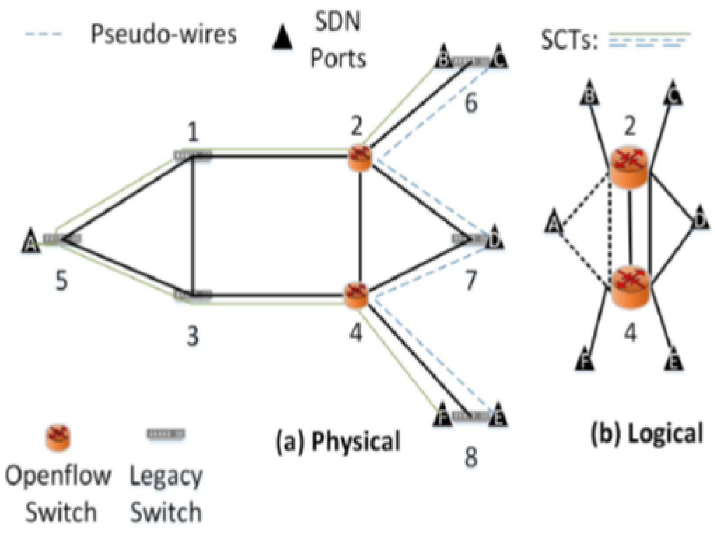


FIGURE 3.3: Waypoint enforcement in Panopticon

link failure, IGP routing information is being employed for the reconnection and re-establishment of a connection.

In hybrid SDN switches, Vissicchio et al. [67] investigated the interaction of distributed such as OSPF and centralized such as SDN routing control planes. Traffic engineering and improved robustness are two important advantages of this collaboration. In addition, the flexibility of routing can be increased by using hybrid nodes based on SDN and conventional architecture, and the network function virtualization can also be provided by using such hybrid nodes. However, the interaction between centralized and distributed control management layer causes the Fresh forwarding abnormalities. It presented the prerequisites for attaining collaboration as well as a theoretical framework of control plane that is based on problem free routing for resolving these forwarding abnormalities.

The operation of conventional and SDN-enabled networks and potential business prospects and research challenges in the hybrid SDN network environment, are all examined by Vissicchio et al. [68]. Based on the diverse functions of conventional and SDN networks, the author presented a number of potential SDN models based on hybrid networks, including the topology-based model, service-based model, traffic class-based model, and integrated model. The network is divided into zones by the network architecture based on topology where each zone is either SDN-based or conventional network-based. Several functions are delivered by SDN, in addition to those provided by conventional networking architectures, that includes forwarding based on the type of network. For some traffic classes, like TCP traffic, the class-based paradigm uses SDN, whereas for others it uses traditional networking. Inconsistent forwarding can result from updating a hybrid SDN network because multiple control planes operate concurrently [69].

Legacy devices provide forwarding tables at the control plane using RIP [70] or distance vector routing protocols like OSPF [71], which use link state routing. The SDN controller issues commands to SDN devices, which the SDN devices then use to fill their forwarding tables. To stop inconsistencies from spreading through network updates, Vissicchio et al. [69] introduce the Generic Path Inconsistency Avoider (GPIA) approach. In order to create the forwarding tables, nodes in a hybrid network using SDN routing strategy as well as conventional protocols for routing purpose. A network update is a modification made between the network's initial configuration and its final configuration. Several operations are performed on a node during a network upgrade. The calculation of forwarding tables may change as a result of an operation.

The forwarding tables at legacy or SDN devices can also have entries added or removed. According to the proposed GPIA method, (i) consistent forwarding is provided by the conventional routing protocol in case of SDN routing is not functional [72]; (ii) existence of consistent forwarding in the initial and final configurations of networks. By modelling the modification, that is, computing the resulting intermediate forwarding paths and contrasting the intermediate configuration with the initial and final network configurations, the recommended method determines whether this update results in forwarding inconsistencies [73, 74]. The GPIA technique will need to be expanded in the future to include more than one node. In addition, GPIA's scalability and adaptability must be thoroughly investigated. This technique is the more recent and efficient of the two since it employs the Generic Path Inconsistency Avoider (GPIA) algorithm with higher performance. As a result, both traditional ISP networks and data center networks can benefit from this strategy [75]. Peng et al. [76] advocates introducing SDN nodes gradually by gradually increasing the number of nodes, resulting in improved traffic forwarding. The suggested architecture is based on the OSPF [71] and OpenFlow [77] protocols, with the number of simulation efficiency metrics for hops and the use of links taken into account. It achieves a significant reduction in network congestion and load balancing.

The OSPF/SDN traffic engineering problem was handled by Guo et al. [78], who created an incremental migration deployment method based on a heuristics approach. In order to improve traffic engineering, Guo et al. looked into and proposed that in ISP networks we can place SDN-enabled devices along with traditional devices. The main objective of such mechanism for migration is to establish the best router placement within an ISP network in order to reduce the maximum connection use. A genetic algorithm selects the optimal migration sequence from a variety of alternate ones. The greedy approach and the static migration algorithm are two further heuristic methods under development.

A traffic engineering architecture based on hybrid networks of SDN, which is able to manage a number of traffic metrics, was developed by Guo et al. [79]. Effective solutions are required for the NP-hard multiple TM traffic engineering problem. For the optimization of routing, Guo et al. implemented a weight change mechanism based on offline mode for traditional devices with a number of TMs based on ratio optimization splitting in online mode. A data mining strategy is used to extract information from previous TMs by keeping past data for all previous TMs and developing a coefficient based on weight for those TMs used as representative. According to simulation results, the suggested method provides effective traffic engineering. The SDN/OSPF Traffic Engineering (SOTE) method for hybrid network based on SDN-enabled devices was introduced by Guo et al. [80]. In traditional networks, routing based on OSPF [71] frequently results in network congestion on the shortest pathways. According to Guo et al., the solution to the OSPF congestion issue is centralized SDN control. The suggested method reduces the maximum link utilization by enhancing the OSPF weight setting. The flows that arrive at the SDN switches are divided by the SDN controller. The conventional nodes employ OSPF on a regular basis. The SOTE Algorithm gives satisfactory results even with SDN-enabled devices are used only 30% in the network. The SOTE analysis in [80] has the drawback of only considering the congestion factor.

Labraoui et al. [81] looked at how an SDN-assisted solution could help reduce routing overhead, accelerate convergence, and reduce packet losses. The SDN-only architecture can boost data throughput by up to 22% while reducing packet loss by up to 25%, making this approach as efficient as a reference system. In this approach, authors looked into a scenario in which the SDN solution was combined with a traditional distributed protocol. This OLSR-based solution consists of two parts: (1) limiting the MPR broadcast scope to a small number of hops, and (2) assisting the SDN controller with routing when hosts are detected outside of the MPR broadcast scope. It improved the packet delivery ratio by 20% and reduced the number of lost packets by 20%. As a result, it shows that the SDN strategy can help with distributed routing protocol operations. Indeed, the proposed solution's centralized structure introduces additional types of issues, such as the development of a Single Point of Failure [82]. For WMNs, distributed routing methods have been in use for a long time and are an important aspect of mesh networks. However, each one is designed to address a distinct WMN issue, and as a result, they have been known to fail under certain circumstances. If link load is not taken into consideration, a protocol's computed shortest path may quickly become overwhelmed under heavy traffic conditions. By providing load balancing services and dynamically provisioning links, SDN could help the routing service.

Labraoui et al. [83] presented a method that would allow the RAN to be offloaded via WiFi access networks using Wireless Mesh Networks (WMNs) generated by the smartphones themselves, with SDN control of IP communications in the edge networks. The tool utilized is NEON, a lightweight SDN utility for dynamic networks [84]. Wang et al. [85] proposed an architecture for QoS Routing employing residual energy and hop count as efficiency indicators for route selection in a multi-hop wireless network linked with SDN. The concept of Multipoint Relay (MPR) is used to optimize transmissions. The controller determines the shortest path, and the Dijkstra Algorithm [86] is used to generate various paths.

He et al. [87] created an aerial satellite overlay hybrid network to build a new SDN/IP network architecture. By cooperatively managing both networks, the design makes it easier to overcome the difficulties and inconsistencies of hybrid SDN. While a centralized network based on SDN is established using the network of satellite, then the ground is used to establish the conventional network. A number of networks are developed on the ground, and the SDN control is used to manage at the overlay level with satellite. Sockets and openflow are used to construct an unique routing model. The packets are subsequently forwarded in accordance with a routing protocol maintained by the routing engine server once the SDN controller has installed the appropriate forwarding entries at the switches. Quagga [88] decodes routing protocol packets and transmits the routing information to the SDN controller, which has a comprehensive understanding of the entire network. Gradual deployment strategies are advised for hybrid networks based on SDN and conventional networks. He et al. [87] focused in particular on a satellite overlay networking scenario. With middle box deployment, this approach handles reliably the path failure cases with reduced overhead while providing good performance. The ISP networks' evolution has been investigated by Poularakis et al. [89] with respect to SDN networks. The ISP networks can be progressively converted to the SDN-enabled networks over an extended period, that may cover many years. Moreover, Poularakis et al. advise against completing the transition in a single stage based on variation in network traffic as well as topologies. Programmable traffic and non-programmable traffic are the two types of network traffic used in the hybrid SDN network created by the migration scenarios [90]. The two primary objectives that are taken into consideration are the maximisation of programmable traffic and the optimization of traffic engineering (TE) flexibility [91]. The formulation of the programmable traffic problem uses a variation of the Budgeted Maximum Coverage Problem (BMCP) [92]. The research of this technology reveals a 54% increase in programmable traffic, allowing for more effective traffic engineering.

To maximize the amount of SDN benefits, Xu et al. [93] looked into hybrid photonic SDN deployment alongside traditional hardware. In the majority of hybrid SDN installations, traditional devices are swapped out for SDN devices so that the SDN controller can manage the majority of the traffic. It suggested a duplicate deployment strategy in which the deployment of SDN-enabled devices is "in addition to" existing devices as opposed to "in substitute of" them. All traffic flows in this hybrid network are collaboratively negotiated by traditional routing protocols along with the SDN controller. The authors used approximation scheme based on a randomized rounding approach along with traffic mapping that solves the problem of routing optimization and joint nature of duplicate deployment for throughput maximization with a constrained budget. The suggested solution increases throughput by 26% in comparison to standard routing. The key benefit of the duplicate deployment is that it increases the utilization of resources allocated to a network and its throughput without requiring any modifications to the existing network architecture [94].

Caria et al. [95] offer the partitioning of a hybrid network based on SDN and conventional networks method for optical circuits of varying nature. The partitioning model uses a method developed by Caria et al [96] earlier, that involved breaking an OSPF network into multiple segments. SDN controller handles SDN-enabled devices directly and connects the OSPF segments. This method integrates SDN based partitioning with those optical circuits that used as bypasses. To put it another way, optical bypasses improve the initial OSPF segmentation, allowing network operators to better manage their networks. The partitioned segments are used to handle huge traffic along the network. In addition, it achieves optimization of utilization of resource by using optical bypasses [97]. Network operators do not have to completely migrate to optical transport or SDN-only transport when using the Caria et al. [95] method. In a hybrid network based on SDN and conventional nodes, the recommended mechanism achieves effective link use and enhances network management and control, according to simulation results.

Martinez et al. [98] proposed a hybrid network based on service type and wireless mesh network of backhaul. The proposed architecture blends the advantages of centralization concept of SDN and conventional network layout in diverse nature. Three levels make up the wireless mesh backhaul concept. In order to produce low-cost capacity at a lower layer, millimeter wave and sub-6 GHz technology are coupled. The control layer of this network is composed of distributed and centralized networks with time sensitive services and data sensitive services. Network services based on energy-efficiency and routing are placed at the top (application) layer. According to the simulation results, the service-based hybrid SDN strategy beats a centralized SDN model by a factor of six by lowering latency. In order to prevent improper communication between the SDN controller and the forwarding nodes when the Control Packet Loss Ratio (CPLR) of the control channel reaches an unacceptable level, Osman et al. [99] introduced a hybrid-SDN strategy with switches network control logic. Furthermore, it allows for the acquisition of aggregated throughput and latency numbers that are comparable to those provided by canonical-SDN under low CPLR.

Table 3.3: Coexistance-Based Hybrid (SDN/IP) Routing Approaches

Approach	**Features**	**Protocol**	**Limitations**
Levin et al. and	Load balancing, fault tolerance, link recovery	OSPF, EIGRP, OpenFlow	Location and number of SDN switches, additional computation to find efficient paths
Vissicchio et al.	Flexibility, partial robustness,	OSPF, GPIA	Topological position and scalability
Peng et al.	Load balancing, congestion control, and making traffic fast-forwarding	OpenFlow, OSPF	Information exchange b/w SDN and IP nodes
Guo et al.	Traffic Engineering and link utilization	OSPF, OpenFlow, SOTE,	Migration sequence of SDN nodes
He et al.	Collaborative management,	OSPF or RIP, OpenFlow	Information exchange b/w SDN and IP nodes
Labraoui et al.	Reliability and performance improvement	OpenFlow, OLSR	One hop communication b/w Controller and nodes
Wang et al.	QoS Routing, Single/multipath routing, and disjoint multipath	OSPF, OpenFlow	Expansion in network size

Approach	**Features**	**Protocol**	**Limitations**
Xu et al.	Duplicated Deployment scheme	OSPF or BGP, ECMP, OpenFlow	Capacity of traditional legacy devices to remove from the network
Caria et al.	Network management and control, efficient link utilization	OSPF, OpenFlow	Flexibility and scalability of the hybrid SDN network
Poularakis et al.	SDN upgrade scheduling, break the routing path into segments	OSPF, OpenFlow	Highly heterogeneous network environment, Controller only maintain an abstract view of topology
Martinez et al.	Resilient to path failures, supports dynamic flow installation	BDN, GPRS, OpenFlow	Incurs high overhead, does not consider path stretch

3.2.2 Cohabitation-based Hybrid Routing

As illustrated in Figure 3.4, the goal of the cohabitation-based hybrid routing strategy is to change the logical architecture of the SDN hybrid switch to allow OpenFlow and IP forwarding to coexist with traditional routing protocols like OLSR, OSPF, AODV, etc. The former is used to communicate with SDN nodes, while the latter is used to communicate with legacy nodes. In other words, network switches that are hybrid SDN switches have both SDN and conventional routing functionalities.

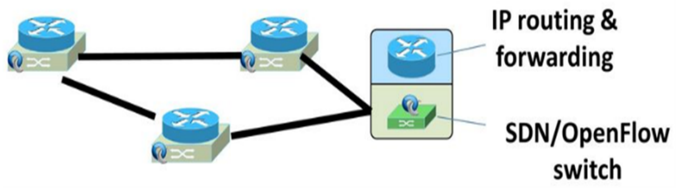


Figure 3.4: Cohabitation-based Hybrid (SDN/IP) Routing

Dely et al. [100] developed a cohabitation-based hybrid architecture to handle a variety of hybrid SDN issues, including performance optimization, efficient and scalable customer mobility, scalable routing, load balancing, and so on. The division of physical interface is done and virtual interfaces are created. Each virtual interface contains its own SSID. The data traffic is managed by these virtual interfaces. Topological alterations, on the other hand, have a significant impact on this architecture [101]. OLSR [102] is a traffic control system. The data flow is managed via OpenFlow by the centralized controller, and resource allocation is also defined. The architecture makes use of a monitoring and control manager to help with mobility management and the NOX operating system to generate flow tables. When tackling the controller's main challenge, Detti et al. [103] adopted Dely et al. [100] as a reference architecture and made some changes to eliminate a single point of failure. As a fallback, it employs a distributed control system. For traffic data and control, a single SSID is employed. Mesh access points (MAP) are used in the architecture, which are linked to a centralized controller [104]. A number of virtual interfaces that connect to other nodes are included in each MAP. To manage data and control, many subnets are employed. OLSR [102] is used in this architecture to handle routing and swap the switching table in the event of controller failure. This type of architecture allows for traffic optimization.

A hybrid architecture based on the extension of Detti et al. [105] was recommended by Salsano et al. to improve fault tolerance. Flow tables and controllers are created in line with the MAPs and controllers of the Embedded Flow Table Manager (EFTM). The synchronization of controllers in such systems is quite difficult [106]. Solutions are required for (a) scalability and fault tolerance, (b) managing the OpenFlow protocol's latency requirements for controller communication, (c) integrating WAN switches, and (d) managing the latency requirements. A realistic evaluation platform is needed, especially for networks with thousands of nodes and links. The Open Network Operating System (ONOS) has a reference node implementation that is open-source and an emulation framework that is open-source [107]. An IP forwarding engine, an IP routing daemon, and an OpenFlow capable switch are all part of the Open-Source Hybrid IP/SDN (OSHI) node (OFCS). The OFCS is depicted in Figure 3.5 as consisting of an OpenvSwitch (OVS) [108] and an IP forwarding engine created using Quagga [88] and IP networking capabilities of the Linux kernel. In contrast to the OFCS, which is connected to actual network interfaces, the IP forwarding engine is connected to a set of virtual OFCS ports. The OSHI framework may be implemented and evaluated using the Mantoo, a collection of management tools for hybrid networks based on SDN along with conventional nodes and large-scale nature. Mantoo [109–111] can be used to create a realistic network environment with large-scale. Mantoo is used for the configuration and testing of OSHI networking architecture and services on dispersed SDN testbeds and a Mininet simulation.

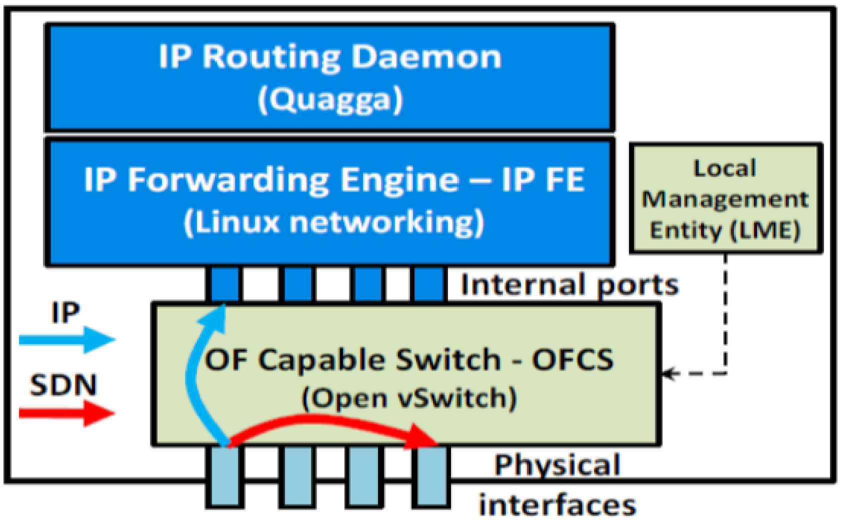

IP Routing Daemon (Quagga)
IP Forwarding Engine – IP FE (Linux networking)
Local Management Entity (LME)
Internal ports
IP
SDN
OF Capable Switch - OFCS (Open vSwitch)
Physical interfaces

FIGURE 3.5: OSHI Hybrid SDN/IP Node Architecture

Hakiri et al. [112] introduced SDNs with WMNs for network virtualization, routing, and traffic engineering in smart cities, enabling enhanced network capacity and flexibility. A wireless mesh network and software defined networking (SDN) were used in this approach to provide a strong and user-friendly system for implementing smart city services. SDN was implemented into WMNs by Hakiri et al. [112] in order to create and maintain a reliable Cyber Physical System (CPS) network that satisfies the QoS criteria of CPS applications. By changing the OpenFlow protocol, a unique technique for routing, network monitoring, and traffic engineering is being offered. The bootstrapping method, a significant advancement, enables both centralized and distributed SDN control planes and makes a dis- tinction between problems with distributed systems and fundamental controller concerns. The authors proposed employing a modified open flow protocol that supports both distributed and centralized SDN control planes to provide network monitoring and traffic engineering [113, 114]. Shastry et al. [115] suggested a software-defined wireless mesh network design to address the problems with traffic balancing brought on by node mobility. In order to reduce the SDN controller's overall response time in a complicated network topology, the proposed model assesses the chance of connection failure in the topology. A different set of paths is suggested after a connection loss is foreseen based on the network's successful traffic stability, reducing control plane overhead [116]. Software-defined networking is used by Kuznetsova et al. [117] to manage wireless mesh sensor networks. By using an SDN controller to govern the network, they are able to increase bandwidth, jitter time, and packet loss performance. A review of the characteristics, workings, protocols, goals, and constraints of the current cohabitation-based hybrid routing techniques is given in Table 3.4.

TABLE 3.4: Cohabitation-Based Hybrid (SDN/IP) Routing Approaches

Approach	**Features**	**Protocol**	**Limitations**
Dely et al.	Load balancing, mobility management, and optimization of resource allocation	OLSR, OpenFlow	Fault tolerance, MAP association and flow path optimization
Detti et al.	Reliability under controller failure and traffic optimization	OLSR, OpenFlow	Excessive packet-in traffic, dynamic topology
Salsano et al.	Multiple controllers' deployment, fault tolerance, A management tool, fast restoration, and traffic engineering	OSPF, MPLS, OpenFlow	Synchronization of multiple active controllers, frequent rule updating, dynamic topology
Hakiri et al.	Network virtualization, routing and traffic engineering, heuristic algorithm based on the submodularity concept	OLSR, OpenFlow	wireless network resources and radio resources management

3.3 Control Plane Failure Recovery Mechanisms in Hybrid SDN Approach

In SDN design, the separation of the control plane from the data plane presents a new point of failure: control plane failure. The community has put little effort into addressing the flaws of the SDN paradigm when it comes to the reliability of the control plane (including controller failures and control channel losses). Multiple SDN controllers were suggested by Lara et al. [118] as a way to avoid

SDN controller failure (controller redundancy); however, this approach may also raise issues with consistency. HyperFlow was introduced by Tootoonchian et al. [119] and is based on physically distributed controllers that are synchronized via a publish/subscribe system. For inter-controller synchronization, ONOS requires complicated procedures. Designing strategies to deal with control plane unreliability is another alternative.

In the event of channel failure, Omizo et al. [120] presented ResilientFlow to restore the control channel via alternate channels (path redundancy). On the other hand, these researches focused on a specific type of issue, like an SDN controller or control channel failure. In addition, implementing a hierarchy or cascade deployment method could be a fix for SDN controller problems. Since version 1.3 [118], OpenFlow has enabled several SDN controllers for regulating a comparable set of forwarding nodes. Event ordering, unreliable event delivery, and command repetition are three inconsistencies that continue to be of concern. During an SDN controller failure, Katta et al. [121] proposed Ravana to ensure event sequencing, accurate event processing, and command execution for exactly once. A module called OLSR-to-OpenFlow (O2O) was proposed in wmSDN by Detti et al. [122] for the forwarding of OpenFlow packets according to the set rules. In order to recover from the SDN controller failure, the O2O module is responsible for forward nodes to dump in the routing tables based on OLSR algorithm. This approach demands that routing tables be transformed into SDN rules. Kobo et al. [123] investigated that very little effort is being made that resolve the challenges of degradation of the performance of control channel, while the deployment of multi-hop wireless networks should be considered for the services of small cells, despite the fact that SDN controllers are implemented with complex functionalities that are used for the handling of management of cluster, and the master ship of device to guarantee the handling of events in a consistent manner, in case when controller fails. In hSDN, illustrated by Figure 3.6, Osman et al. [124] specifically addresses the problem of control channel failure by proposing a straightforward and a quick strategy for the recovery of SDN controller from occurring of faults, and other events like channel used for control plane communications gets failed or degraded. The authors propose variations in Control Packet Loss Ratio (CPLR) in the communication of control plane.

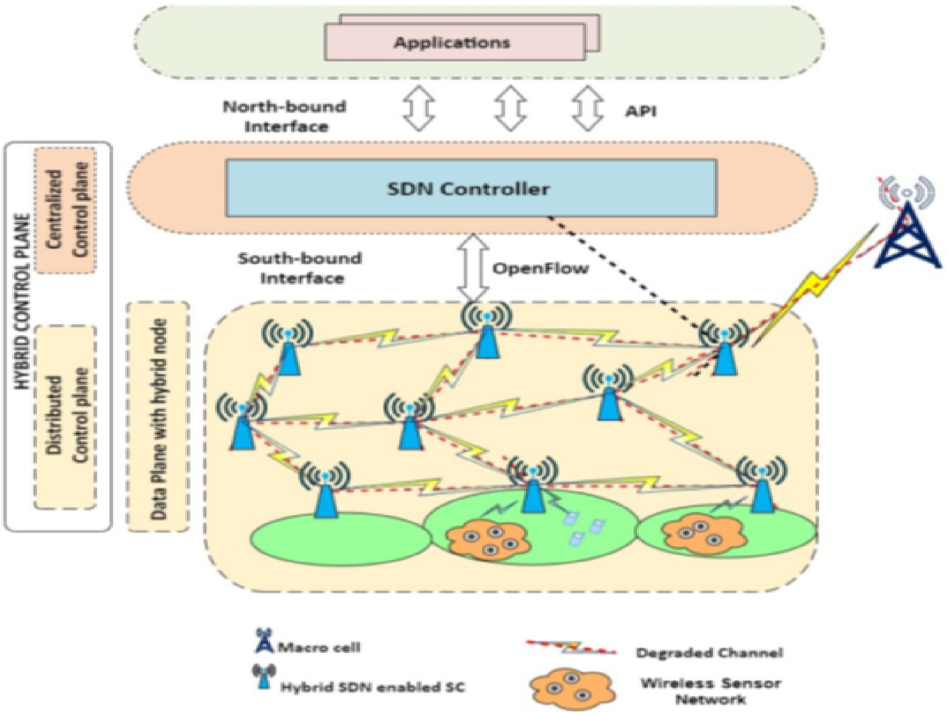


FIGURE 3.6: Architecture of hSDN

Table 3.5 provides the summary of Control Plane Failure Recovery Mechanisms in existing hybrid SDN approaches

TABLE 3.5: Control Plane Failure Recovery Mechanisms for Hybrid SDN Approach

Mechanism	**SDN Controller Failure**	**Control Communication Channel Failure**	**Degraded Control Communication Channel**
Tootoonchian et al. : HyperFlow	Yes	No	No
Omizo et al. : ResilientFlow	No	Yes	No
Katta et al. : Ravana	Yes	No	No
Detti et al. : wmSDN	Yes	No	No
Osman et al. : hSDN	Yes	Yes	Yes

3.4 Network Monitoring Applications for Hybrid SDN Approach

As hybrid network based on SDN-enabled devices allow both classical and SDN devices to coexist and work together, they are becoming increasingly popular. However, fine-grained monitoring is needed to leverage the benefits of SDN and achieve near-optimal traffic engineering performance with little SDN implementation. Monitoring related developments in hybrid network based on SDN-enabled networking are discussed in this section.

OpenNetMon is a network monitoring module that offers numerous methods for gauging various aspects of network performance, according to Van et al. [125]. OpenNetMon has the ability to continuously track network delays, packet loss rates, and speed. First, while measuring throughput rate, it employs a variable frequency to query simply the final switch on the forwarding path. S/T can be used to determine the forwarding path throughput. 'S' represents a flow's packets count that is retuned within the Time 'T' used for sampling by the counter. The forwarding path shows some packet loss that is taken by collecting counter statistics from the first and last switches on the path, dividing the result by the time, and lastly subtracting the first counter's increase from the last counter's increment in a measurement cycle. OpenNetMon sends probing messages to the forwarding path's data layers in order to ascertain the delay. These messages travel through each node before returning to the controller along the forwarding path. Calculating the time differences will give us the delays regarding communication.

Despite the fact that there are a lot of network flows overall, it is impractical to measure the size of each flow separately due to the Ternary Content Addressable Memory (TCAM) table and the CPU's restricted measurement capabilities. The iSTAMP approach created by Malboubi et al. [126] tackles this problem. It assesses the convergence flow or the magnitude of the k flows of biggest nature, then the integrated traffic matrix is estimated using the technologies that are most appropriate. Despite appearing the striking of stability among the resource constraints of a particular network and accuracy in measurement. The iSTAMP has the below mentioned drawbacks. 1) given the limitations on flow aggregation practicality, the flows are combined into a single one having same destination node, when SDN switches forward packets, and the priority and wildcard matching principles are used to select a forwarding item. When aggregating flows, however, the iSTAMP approach stays clear of these limitations. The iSTAMP measures single flows over a number of varying periods that determines the highest number of flows, which generates measurement costs that cannot be disregarded.

Tootoonchian et al. [127] presented 'OpenTM' that is an estimate approach for the matrix of SDN traffic. Based on the routing information and flow forward- ing path details supplied by the controller, all of the active flows can be detected by proposed approach. The OpenTM offers a variety of selective querying mechanisms for receiving precise information on packet number and flow evaluation for routing nodes. To effectively manage a range of measurement tasks and applications, a comprehensive framework for SDN statistics measurement must be developed. The PayLess is a polling-based flow measuring platform with low costs, according to Chowdhury et al. [128]. The PayLess framework, which provides programs with a RESTful interface, is a component of the OpenFlow controller. Interfaces of standardized programming for the application used in networks while maintaining the application's data that is used for networks is the main features of Payless architecture. The framework for traffic measurement may be expanded to include the unique components of users, tasks are defined using JSON format.

According to Yu et al. [129], the OpenSketch, a framework to measure the network statistics is an abstract framework in nature that broadly uses SDN technology. According to proposed solution, flow-based measurement is inadequate for the needs of various users due to flaws including counter overhead and measurement inaccuracy. Contrarily, OpenSketch enables user customization, and various tasks have distinct hardware needs, making it challenging to perform many tasks on the same system, in order to properly address adaptability and effectiveness. When measuring control and data layers, OpenSketch advises keeping them separate since data layers are created as three-phase processes that can be constantly changed. A variety of hash methods are available in the OpenSketch first stage to map packets into a tiny quantity of measurement data. Furthermore, network applications require multiple concurrent flow monitoring jobs rather than providing a single measurement activity at a time.

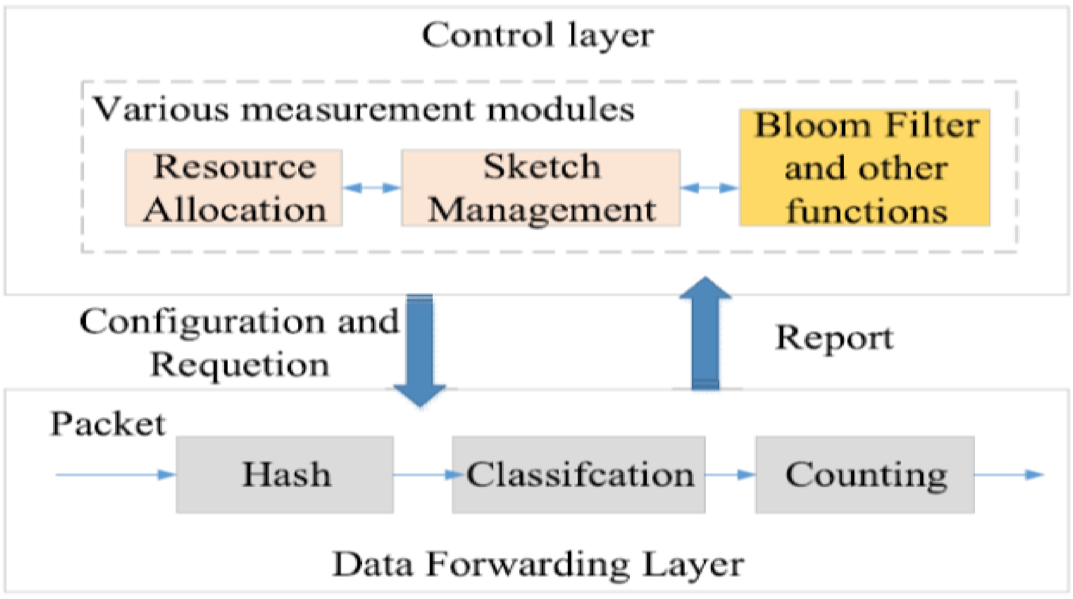


FIGURE 3.7: Architecture of OpenSketch

The DREAM is an adaptive hardware resource management architecture that successfully balances measurement precision and resource costs, according to Moshref et al. [130]. The significant features of the DREAM architecture are as follows.

1. The accuracy of measurement of a flow is determined by the hardware resources allotted to it. There will come a moment where the ability to promote task accuracy decreases as the quantity of resources allotted to the work increases

2. Because the resources required by measurement activities vary based on the time and magnitude of the flow, these resources can be adjusted to improve their utilization efficiency.

The user layer, which makes up the top level of the DREAM, is in charge of creating measurement tasks, which include, among other things, activity kinds, specific flow thresholds, and accuracy standards. The SDN controller's DREAM algorithms are at an intermediate level. They take user tasks, create task objects that are related to them, and then divide these abstract responsibilities over several switches. In practice, the deployment process entails both storing measurement logic in the TCAM and requesting TCAM resources from the switch [131]. Based on input from real-time traffic monitoring, the DREAM algorithms may adjust the resources allotted to each task or combine resources from other activities to determine the accuracy of measurements. On the other hand, depending on the situation at hand, the algorithms can choose whether or not to accept a new assignment. Basically, SDN-enabled devices are used to evaluate resources to store and to report flow statistics in real-time. Hartung et al. [132] presented SOFTmon, a NOS-independent monitoring system based on switch, port, and flow-level information, although accuracy is limited to certain tasks.

In order to reduce the number of monitoring messages, the amount of bandwidth used, and the reporting latency at the control plane, Yahyaoui et al. [133] created a heuristic approach based on a flow monitoring method. Determining which switch would report statistics on which flow was the challenge. A low-overhead in-band network telemetry and monitoring solution for wireless mesh networks was presented by Haxhibeqiri et al. [134]. The majority of industrial applications demand determinism in terms of latency, reliability, and throughput; nevertheless, enabling end-to-end network monitoring, which takes into account end devices as well, is the first step in network verification. This paper presents the design of a proof-of-concept (PoC) implementation for an in-band network telemetry enabled node architecture. A real-world SDN-based wireless network is monitored using the proof-of-concept implementation, enabling on-the-fly (re)configuration based on data monitoring. The suggested in-band monitoring technology offers a six-fold lower overhead than existing active monitoring methods on a one-hop link. Further evidence of the suitability of the recommended monitoring technique is provided by the fact that (re)configuration choices based on observed data satisfy the specified application criteria.

The hybrid federated control architecture proposed by Bellavista et al. [135] is based on a middleware that controls the quality of service (QoS) in spontaneous networks, i.e., WMNs, where mesh clients not only create their own multi-hop networks but also integrate into the network infrastructure. In conventional WMNs, infrastructure network administrators deploy mesh routers, but mesh clients often operate on a best-effort basis. Because of this, network-wide collaboration is difficult. However, [135] envisions a networking environment in which mesh routers and clients work together to share information about the resources made available, hence maintaining the necessary QoS. In this scenario, a WMN is composed of numerous closely spaced subsets of nodes that are each under the supervision of a single controller. Network traffic is managed autonomously by controllers on each island, while they collaborate with one another to decide how to prioritize traffic flows between the islands. WMN nodes regularly check the availability of their controller and, if necessary, can switch to a better one. Every WMN node that utilizes the recommended middleware has the capability of turning into a controller. A node assumes the controller position inside an island based on a node election system that evaluates various crucial factors such as computing capability, energy, and incentive mechanisms. For instance, WMN nodes have the ability to dynamically collaborate by selecting a new controller when necessary, such as when the current controller runs out of power.

The FlowSense [135] monitoring module Yu et al. designed for the SDN controller may assess dynamic changes in network flows in accordance with the signals the SDN controller receives. After receiving a message stating that a particular flow has been removed, the controller, calculates the flow's throughput rate, it divides the flows values by its size. The reactive OpenFlow installations caused the creation of FlowSense, switches send control signals whenever a new flow enters or a flow entry expires. The presence of a large number of flows generates a significant number of control packets, which can overwhelm both the controller, that cannot handle all control traffic in a timely manner, and the switches, that cannot run at line speed and quickly deplete their flow tables. Controllers in networks of 100 switches, for example, may have to process up to 10 million PacketIn messages per second with fresh flows entering every 10s. In practice, the need for scalability drives operators to increasingly adopt alternative OpenFlow deployments, such as distributing controller functionality across multiple machines, proactively setting up rules to never expire to avoid triggering control traffic, and using wildcard rules to reduce control traffic.

Table 3.6 summarizes the available network monitoring apps utilized by hybrid SDN and WMNs.

TABLE 3.6: Network Monitoring Applications for Hybrid SDN Approach

Method	**Features**	**Analysis**
Van et al. : OpenNetMon	Data fetching is adaptive	Increase in accuracy with increasing overhead
Malboubi et al. : iSTAMP	Partitioning of TCAM for traffic aggregation and deaggregation	Increase in accuracy with additional mechanism prioritize flows
Tootoonchian et al. : OpenTM	Continuous polling to collect flow statistics	Increase in accuracy with increasing overhead
Yu et al. : FlowSense	PacketIn and FlowRemoved OpenFlow messages used	Increase in accuracy with decreasing overhead
Chowdhury et al. : PayLess	The polling algorithm is adaptive with flow frequency	Variation in accuracy and overhead with polling interval length
Yu et al. : OpenSketch	Query-based monitoring, Wildcard rules at switches for only monitoring a large aggregate of flows instead of all flows for reducing monitoring overhead.	Low memory consumption with high accuracy
Moshref et al. : DREAM	Dynamic deployment of resources with the required level accuracy level, Hash-based switches for collection of traffic information	Concurrent tasks give more accuracy, Increase in accuracy with the careful delegation of monitoring rules

Method	**Features**	**Analysis**
Hartung et al. : SOFTmon	NOS-independent monitoring with utilization information on switch, port, and on a flow level	Accurate for specific tasks only
Yahyaoui et al.	A heuristic solution, decreases monitoring messages, bandwidth consumption and reporting latency	Identifies the switch to report statistics for a specific flow
Haxhibeqiri et al.	low-overhead in-band network telemetry, proof-of-concept implementation monitors a real-world SDN-based wireless network, allows for on-the-fly (re)configuration	six-fold reduced overhead
Bellavista et al.	Hybrid federated control architecture, A middleware to manage QoS	Controller election is dynamic whereas controller selection is opportunistic

3.5 Conclusion

In coexistence-based hybrid routing schemes Levin et al. [47], Vissicchio et al. [136], and Guo et al. [137] consider the physical position of the controller, the number of SDN nodes and the scalability of the controller as the most important control management challenges. In wide or highly complex networks where controllers have to make fast decisions on a high frequency of events such as connection failures, dynamic traffic demands, regular arrival of new flows, etc., these problems may be very critical. However, He et al.[138], Labraoui et al. [85], Wang et al. [100], and Xu et al. [139] consider the sharing of topological information between traditional routers and SDN nodes as the most critical issues, and SDN nodes need to be intelligent enough to exchange link-state messages to get their neighbors information. The analysis of cohabitation-based hybrid routing schemes shows that Dely et al. [140], wmSDN by Detti et al. [141], Salsano et al. [132] and Hakiri et al. [142] consider fault tolerance, excessive control traffic, and dynamic topology as the most significant challenges for the blending of SDN and IP in one node. These challenges may be mitigated by designing SDN nodes in such a way as to have a local management entity that may be used to create a logical interface between the two different paradigms of centralized and distributed network solutions.

The analysis of control plane failure mechanisms illustrates that hSDN approach by Usman et. al [124] addresses all the three aspects of control plane failure that include controller failure, control channel Failure, and degraded control channel. However, HyperFlow by Tootoonchian et al. [119], Ravana by Katta et al. [121], and wmSDN by Detti et al. [138] proposed the solution for controller failure issue whereas ResilientFlow by Omizo et al. [120] proposed a solution for control channel Failure issue.

Furthermore, the analysis for network monitoring approaches show that mostly the existing solutions such as OpenNetMon by Van et al. [143], iSTAMP by Malboubi et al. [144], OpenTM by Tootoonchian et al. [145], and DREAM by Moshref et al. [131, 146], result in increase in accuracy on the cost of increasing overhead. However, OpenSketch by Yu et al. [129], and FlowSense by Yu et al. [141] result in high accuracy with low memory consumption and low overhead respectively.

Chapter 4

Soft-Mesh: A Robust Routing Architecture for Hybrid Wireless Mesh and Software Defined Networks

4.1 Introduction

The choice of the best path and the transmission of data along it are both crucially dependent on routing protocols. Any routing protocol is built on the cost metric. The output of a routing protocol improves with greater cost metrics. The cost measure of a network link is the cost of sending packets across that channel. Due to the uneven distribution of link characteristics among nodes, defining a cost indicator for wireless networks is considerably more difficult than for conventional wired networks. The quality of wireless network transmission is impacted by a number of issues, including interference, channel fading, obstructions, and background noise [138]. However, WMNs, in particular, are thought to be self-organizing and dynamically configured, with nodes that form and maintain mesh connections on their own. This feature gives WMNs many advantages such as easy and cheaper installation, reliable connectivity, flexible interoperability with other existing wireless networks [139]. Despite these numerous benefits, the number of users wishing to exchange information is increasing, resulting in issues such as packet loss, increased transmission delay, and overcrowding of gateways, which serve as network exit points. It is feasible to implement routing and traffic load-balancing algorithms at the gateways using the Software Defined Network (SDN) architecture and the Openflow protocol, which may be deployed at the level of the SDN Controller [140]. As WMNs can operate on a multi-protocol basis down to the transport layer headers, the introduction of SDN ideas into these networks gives these networks more flexibility in the implementation of packet processing activities like routing and filtering. This versatility can promote creativity in the control of equipment mobility, intricate traffic engineering, and the most effective use of the constrained communication capabilities of WMNs. Furthermore, by re-designing network control logic, the network management in WMNs is simplified by incorporating SDN technology.

4.2 Architecture of the Proposed Routing Approach

The main platforms, technologies, and techniques employed for the evaluation of our suggested hybrid method are described in this section. The SDN controller platform serves as a foundation for implementing the proposed architecture ‘Soft-Mesh’. Numerous SDN controller platforms are available, including NOX, POX, Beacon, Ryu, Floodlight, ONOS, and OpenDaylight (ODL) [141]. In general, Carriers are more interested in ONOS since it places a greater emphasis on performance factors and clustering to boost availability and scalability. As a result, ONOS places a greater emphasis on carrier-grade networks, and telcos participate in their projects. ODL features more vendors than ONOS, including Cisco, Juniper, and NES. Therefore, for the purpose of implementation, we choose OpenDaylight SDN Controller, and network nodes including SDN hybrid and legacy nodes. The detailed description of ODL SDN controller is provided in Section 2.1. OpenFlow protocol is used as a southbound API to create communication between control plane and data plane, detailed description is provided in Section 2.2. However, the subsequent sections provide the description of SDN hybrid and legacy mesh nodes.

A node in the SDN networking architecture queries the controller for path information when it wants to connect to another node, which results in the application of the required routing rules. In this instance, creating connections from one node to another depends on the connection to the controller node. Due to the static nature of network nodes in wired networks, routing rules do not frequently change or update, having little impact on the controller's node connectivity. The constant mobility of nodes in wireless networks, particularly wireless mesh networks, has a substantial impact on node-to-node communication. The issues of WMN backbone routing and network monitoring are the foundation of the suggested routing design, called "Soft-Mesh". The proposed routing considers a hybrid topology under discussion, as shown in Figure 4.1, consists of an SDN Controller connected to SDN hybrid nodes and legacy nodes. This method uses IP-based forwarding for data packet delivery and OpenFlow for control traffic. Four modules make up the controller: load balancing, network monitoring, traffic measuring, and routing.

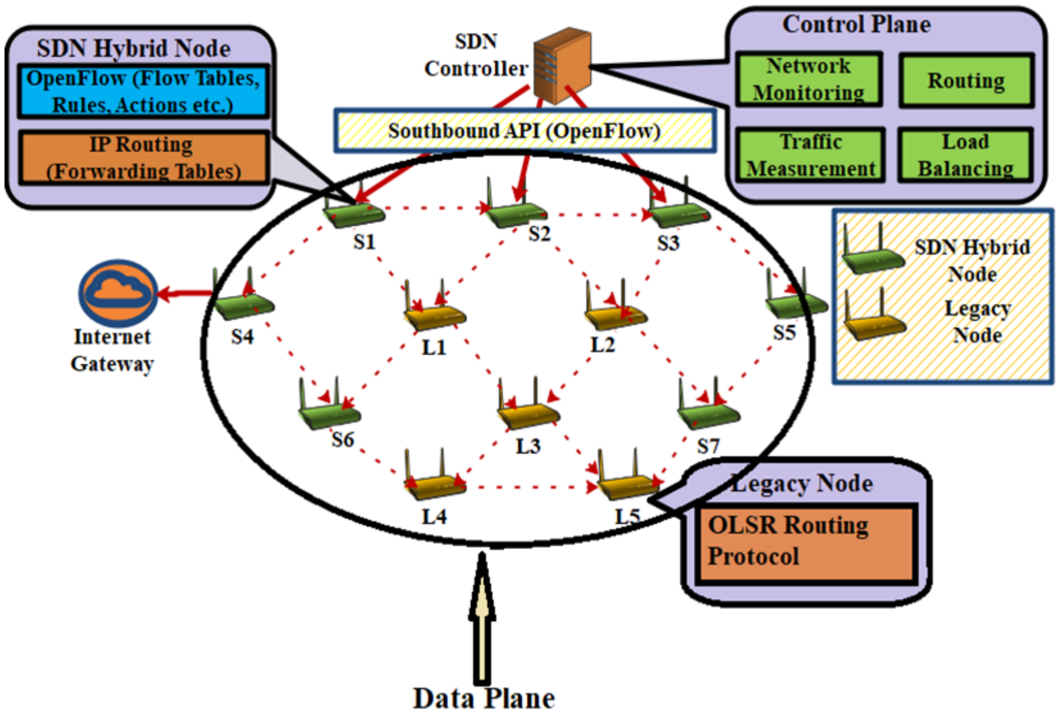


FIGURE 4.1: Architecture of Soft-Mesh

4.2.1 Routing Module of Proposed Architecture

In order to manage the network and provide services like routing, an SDN controller needs up-to-date knowledge of the network state, in particular the network topology. Topology discovery is not supported by any specific feature in OpenFlow switches; it is the controller's sole obligation to provide this service. Furthermore, topology discovery in OpenFlow-based SDNs is not defined by any formal standards. However, the majority of current controller platforms use the OpenFlow Discovery Protocol to implement topology discovery (OFDP). The IEEE 802 Local Area Network's Link Layer Discovery Protocol (LLDP), which enables nodes to broadcast their capabilities and neighbors to other nodes, is used by OFDP. Switches do not forward LLDP packets; instead, each port of the switch is used to send each packet across a single hop to a multicast address that has been bridge-filtered. Every switch keeps the information learned from LLDP packets it has received in a local Management Information Base (MIB). By crawling each node in the network and obtaining the pertinent information from the MIB, such as via SNMP, a network management system can determine the network topology. The protocol is demonstrated in Figure 4.2.

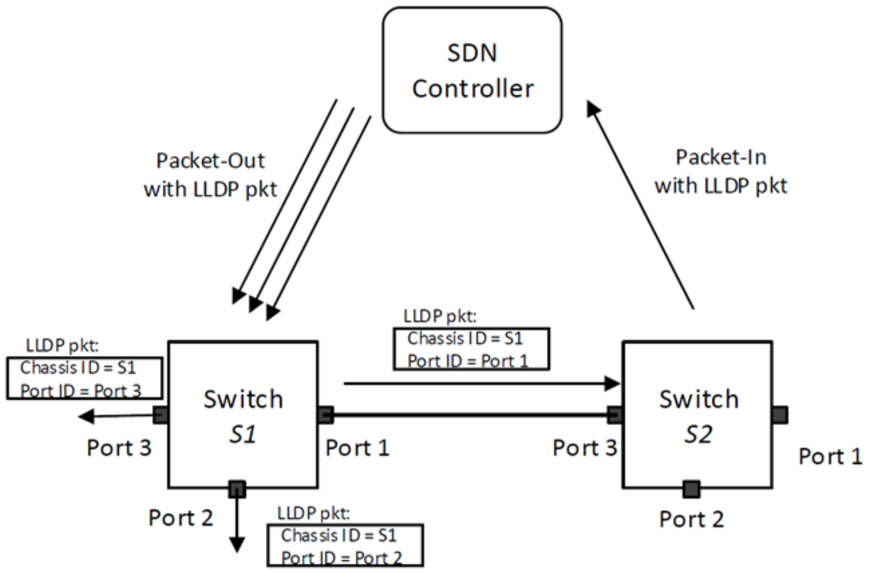

SDN Controller
Packet-Out with LLDP pkt
Packet-In with LLDP pkt
LLDP pkt: Chassis ID = S1 Port ID = Port 1
LLDP pkt: Chassis ID = S1 Port ID = Port 3
Switch S1
Port 3
Port 1
Port 2
LLDP pkt: Chassis ID = S1 Port ID = Port 2
Switch S2
Port 3
Port 1
Port 2

FIGURE 4.2: Topology Discovery Protocol

The proposed routing module provides the shortest path approach to construct an effective routing strategy to route packets across the hybrid nodes and legacy nodes

[147, 148]. As illustrated in Figure 4.3, there are two submodules that make up the hybrid node routing function. One supports SDN routing by implementing the OpenFlow protocol to transmit routing rules and policies, while the other supports IP routing by using the traditional OLSR routing protocol. Cohabitation of SDN and IP is preferred in order for the controller to, where possible, send the best path options to the nodes. If the controller is down or unavailable, packets are sent via the conventional routing protocol. Every SDN node keeps track of its neighbors, often updates the routing table, and selects the shortest new route to every destination. The controller accomplishes this by obtaining topology data from nearby SDN nodes.

4.2.1.1 SDN Hybrid Nodes – Cohabitation of OpenFlow & OLSR

We separated the routing functionality into two layers, in order to enable effective and scalable routing in wireless mesh nodes in order to support SDN integration. The OpenFlow protocol is supported by the top layer of SDN routing for data transmission. At the lowest layer, both the OLSR routing protocol and IP-based forwarding are used. It is the former's duty to inform the SDN controller of OpenFlow policies. This latter component is in charge of directing IP routing within the OLSR interfaces of the mesh nodes. We employ an in-band control mechanism to establish long-distance wireless communication among the wireless mesh backhaul in order to allow the controllers to speak with all of the geographically dispersed nodes. IP-based and SDN routing have two benefits. Before setting the mesh nodes, the controller can use its own routing algorithms to find the optimum route and add, remove, or update OpenFlow rules. The most recent network conditions can also be given by the nearest mesh node. To route packets in accordance with OLSR routing tables at the controller's request, the second technique makes use of OpenFlow protocol. The addition or deletion of a new mesh node or wireless link is just one example of the changes to the topology graph that OLSR provides. The routing database is frequently updated, the newest shortest route is chosen for each destination, and each wireless mesh node keeps track of its neighbors in a list known as the multipoint relays (MPR) selection list. The Controller then gets the topological information from its mesh nodes.

As shown by the Figure 4.3, by dividing each physical node into two full-duplex virtual nodes, each with its own virtual hardware resources and virtual wireless interface, such as PHY1 and PHY2, the coexistence of OpenFlow and OLSR serves to address the issue of network virtualization as well as the challenge of effective and scalable routing. OLSR and OpenFlow are bridged by using virtual interfaces *br0*, *br1*, *br3*. It also contains two wireless interfaces wlan0 and wlan1 for wireless mesh network whereas another interface tap is used to integrate a real mesh node into SDN environment. In order to support four virtual APs on a node, ESSID 1 and ESSID 2 are two non-overlapping virtual APs that can be created from each physical access point (AP). Since each virtual ESSID has its own virtual wireless channel, mobile clients can switch between them and communicate across the virtualized channels. Each SSID passes communications independently of the others in order to differentiate control traffic, or signaling, from other communications, i.e., data. A physical AP can be split into two virtual ones for two reasons. The first benefit is the optimal scheduling of airtime fairness through channel sharing, which enables efficient downlink bandwidth sharing between several smart clients. Second, it resolves the difficulties with uplink channel access when several clients send at once, allowing those smart clients to transmit quickly and at high data rates. It is also possible for clients to be synchronized with the wireless network and for each virtual AP to emit Beacon messages in order to give its own traffic indicator map thanks to the ability of two virtualized access points to coexist inside of one wireless node. These beacon frames, also known as management frames, are used by mesh nodes to verify that every client connected to a wireless node is still active. This method allows for the simultaneous use of current link layer protocols and MAC setting changes.

4.2.1.2 Legacy Nodes with OLSR Protocol

Along with SDN hybrid nodes, the topology under consideration also includes legacy nodes. Each legacy node uses the OLSR routing protocol, which maintains

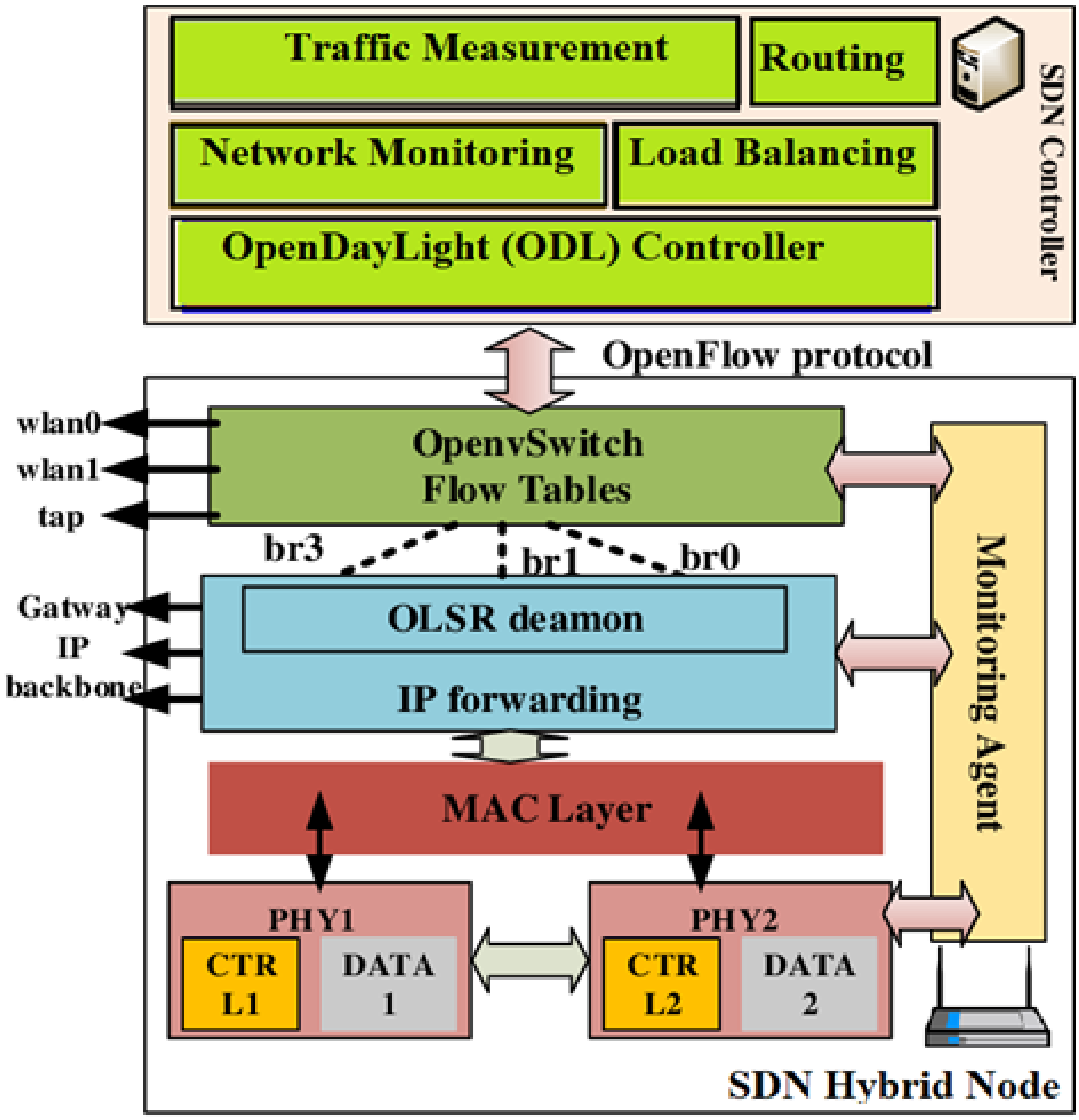
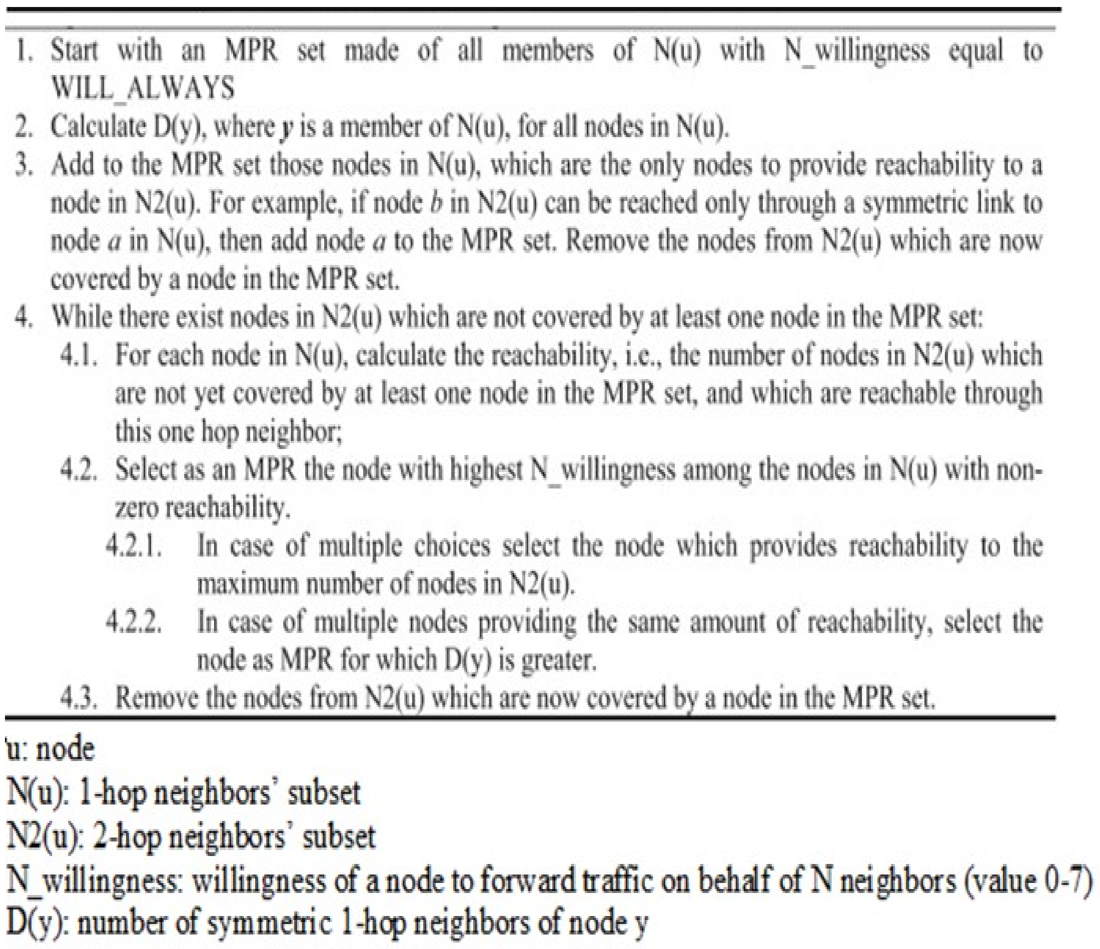


FIGURE 4.3: The SDN Hybrid Node Architecture

an orderly database outlining the topology of an autonomous system. The local state information of each router, the network state information related to the router (the routers connected to this network), the external state information (the external routing information of this autonomous system), and other components are all included in this database. In contrast to distance vector routing technologies, each OLSR-running router disseminates the corresponding status data throughout an independent system. Each node regularly floods the status of its links in the OLSR routing protocol and rebroadcasts the link state information it has received from its neighbors. Nodes keep note of the link state data they receive from other nodes, which is then utilized to calculate the next hop to each destination. A spanning tree is expanded by the nodes, that leads to acquiring complete network topology.

The OLSR protocol employs the MultiPoint Relaying (MPR) approach to minimize message flooding. This is accomplished by having each network node, N, select a collection of close-by nodes to serve as multipoint relays, or MPR(N), and retransmit control packets from N-Non-MPR(N) neighbors process the control packets sent to them by N, but they do not forward the messages. When MPR(N) is selected, all of N's two-hop neighbors are covered by MPR's (one-hop neighbors). The diffusion mechanism of TC messages in OLSR routing protocol is an essential component of its optimization that is based on the process of MPR selection. The MPR selection process of OLSR routing protocol is demonstrated by Figure 4.4 whereas Figure 4.5 illustrates flow chart of topology control messages in OLSR.



```
1. Start with an MPR set made of all members of N(u) with N_willingness equal to
   WILL_ALWAYS
2. Calculate D(y), where y is a member of N(u), for all nodes in N(u).
3. Add to the MPR set those nodes in N(u), which are the only nodes to provide reachability to a
   node in N2(u). For example, if node b in N2(u) can be reached only through a symmetric link to
   node a in N(u), then add node a to the MPR set. Remove the nodes from N2(u) which are now
   covered by a node in the MPR set.
4. While there exist nodes in N2(u) which are not covered by at least one node in the MPR set:
   4.1. For each node in N(u), calculate the reachability, i.e., the number of nodes in N2(u) which
        are not yet covered by at least one node in the MPR set, and which are reachable through
        this one hop neighbor;
   4.2. Select as an MPR the node with highest N_willingness among the nodes in N(u) with non-
        zero reachability.
        4.2.1. In case of multiple choices select the node which provides reachability to the
               maximum number of nodes in N2(u).
        4.2.2. In case of multiple nodes providing the same amount of reachability, select the
               node as MPR for which D(y) is greater.
   4.3. Remove the nodes from N2(u) which are now covered by a node in the MPR set.
```

u: node
N(u): 1-hop neighbors' subset
N2(u): 2-hop neighbors' subset
N_willingness: willingness of a node to forward traffic on behalf of N neighbors (value 0-7)
D(y): number of symmetric 1-hop neighbors of node y

FIGURE 4.4: MPR Selection in OLSR

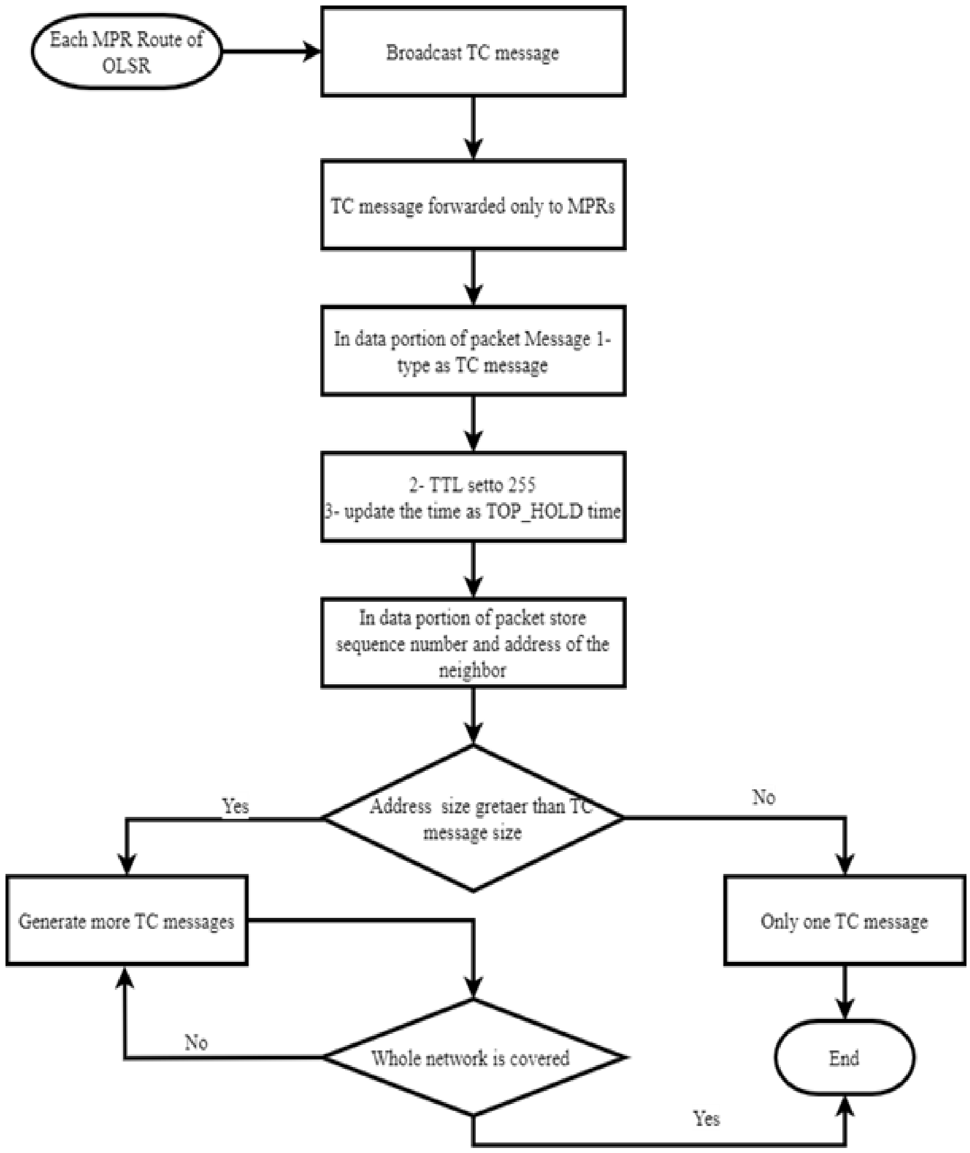

Each MPR Route of OLSR
Broadcast TC message
TC message forwarded only to MPRs
In data portion of packet Message 1- type as TC message
2- TTL setto 255
3- update the time as TOP_HOLD time
In data portion of packet store sequence number and address of the neighbor
Address size gretaer than TC message size
Yes
No
Generate more TC messages
Only one TC message
Whole network is covered
No
Yes
End

FIGURE 4.5: Flowchart of TC Messages in OLSR Routing Protocol

4.2.1.3 Standard Operations of Proposed Routing Module

Standard operations of the proposed routing method are presented in this section. On receiving the packet of a particular flow, a hybrid node analyses the first packet of a relevant flow when it gets it to decide where it should be routed as the next hop, as illustrated by Figure 4.6. It works in one of the two ways, SDN forwarding and IP forwarding.

1. In case of SDN forwarding, OpenFlow performs following operations

(a) If the received packet belongs to an SDN Switch, extract packet header information and match with flow table entry

(b) If the match is successful, next hop is determined and

(c) If it is unsuccessful then header information is forwarded to the controller

(d) Controller updates the flow table with relevant match action

(e) This lookup process is the repeated until the destination is identified

2. In case of IP forwarding, next hop is determined by OLSR protocol that is cohabited with OpenFlow protocol

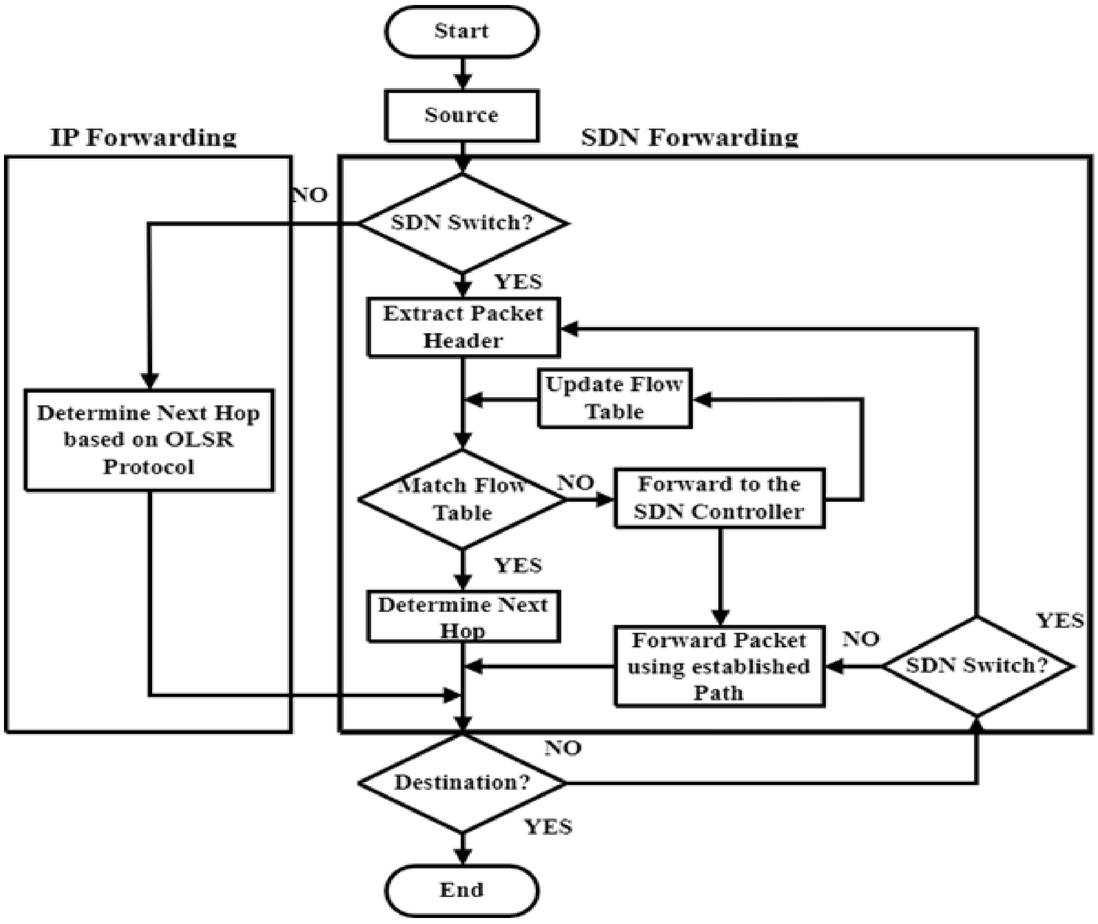


FIGURE 4.6: Flow Chart of Proposed Routing Module

As shown in Figure 4.7, two tables are used in the SDN hybrid node architecture; TCAM table and SRAM table respectively. For IP forwarding entries, the latter is used, whereas the former is used for OpenFlow forwarding entries. The software node OpenVSwitch, which implements a software pipeline based on flow tables [149], is used by each hybrid node to forward OpenFlow messages. An IP-based forwarding daemon using the common OLSR routing protocol is also included [150]. OpenVSwitch utilizes virtual network interfaces to take advantage of IP networks' capacity to route packets along the quickest path, bridging OpenFlow and conventional routing protocols. When an OpenFlow message is received, the following actions are taken:

1. Extract the header for the packet to get flow specifications
2. To balance the flow table, look at TCAM table entries
3. If the respective flow entry is not identified, then it forwards packet information to the controller.
4. The controller sends to the hybrid node the packet relevant rules
5. The SDN hybrid node flow table is updated accordingly
6. If the flow input in the TCAM table is found, the next hop is calculated, then the packet is forwarded.

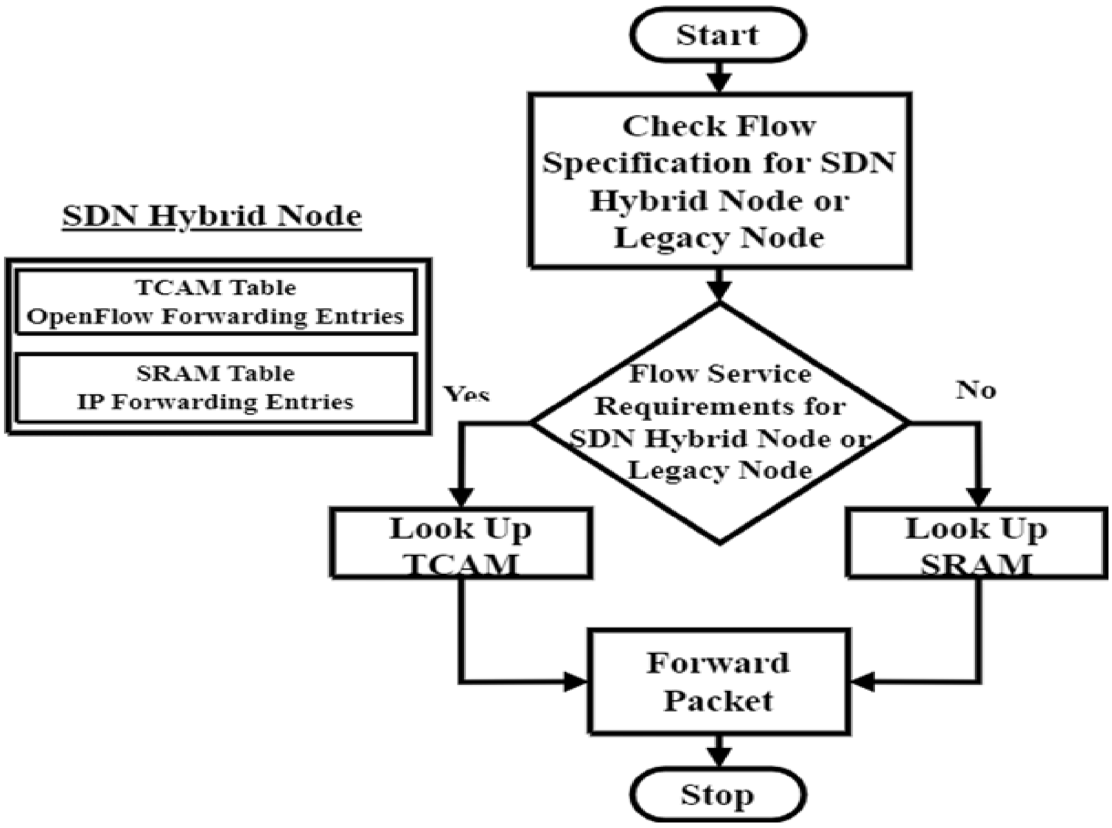

Start
Check Flow Specification for SDN Hybrid Node or Legacy Node
SDN Hybrid Node
TCAM Table
OpenFlow Forwarding Entries
SRAM Table
IP Forwarding Entries
Yes
Flow Service Requirements for SDN Hybrid Node or Legacy Node
No
Look Up TCAM
Look Up SRAM
Forward Packet
Stop

Figure 4.7: Working of SDN Hybrid Node

Using the control subnet, nodes periodically send the controller their nearby information tables so that the controller can see the entire network's topology. It periodically finds the shortest path (in terms of the number of hops) from the source to the destination using Dijkstra's Algorithm [151]. The controller notifies participating nodes (source and intermediate nodes) of the modified rules if an optimal path is found. The graph topology, which includes all reachable nodes and the links connecting them, is used by the controller to identify the new optimal path, as demonstrated by Algorithm 4.1. The process then creates updated OpenFlow rules to program flow entries at each stage of the software pipeline. The equivalent was put up by the nodes. For each iteration, the optimal path function enables the search for the optimum path. We created all of the processes and features necessary for data routing along the predetermined path on the controller side. It specifically uses the routing feature to send data to SDN routers. From the arriving packets, the data path is first collected, then data and protocols are used to initialize the SDN controller. All routers in the destination path are then introduced to OpenFlow rules.

Algorithm 4.1: Path Optimization Algorithm

```
Data: rules, PATH
Result: Function to find optimal path
optimalPath(rules);
if (∃ PATH in (rules) then
    PATH ← find(rules);
    return PATH
else
    rules ← calculateNewRules(); FlowMod_router();
    return rules
end
optimal_path(rules);
```

In contrast, if the IP packet is received, the next hop is determined using the OLSR routing protocol, and the message is according to that calculation. Figure 4.8 shows how a legacy node functions when the standard OLSR routing protocol is used. Using HELLO messages, it primarily collects information from its 1-hop and 2-hop neighbors before selecting Multipoint Relays (MPRs). These relays are utilized when forwarding a transmitted packet to reduce the number of redundant retransmissions. This strategy restricts the node collection retransmitted by a packet from all nodes to a subset of all nodes. In this manner, it computes its forwarding table.

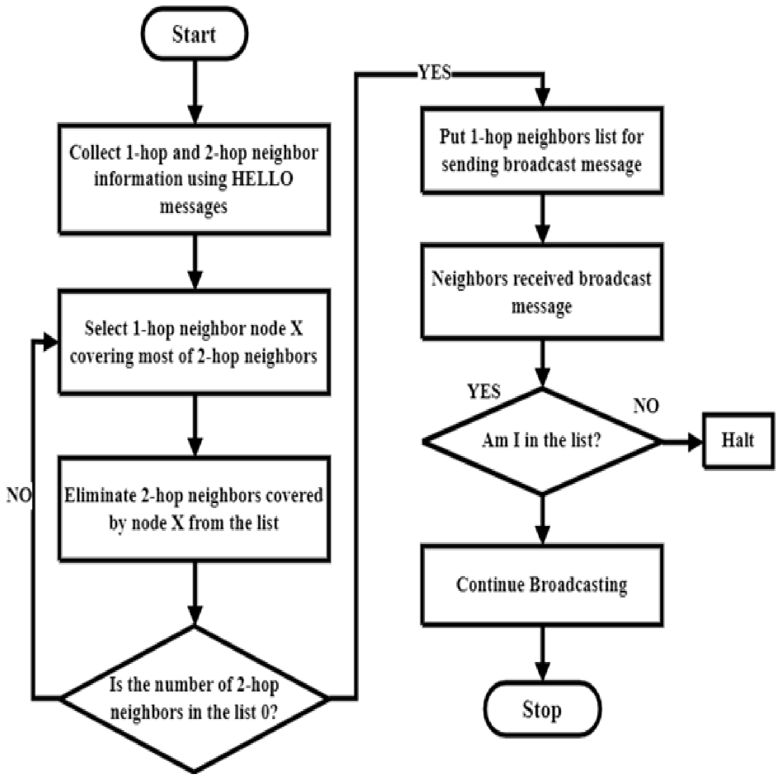

Start
Collect 1-hop and 2-hop neighbor information using HELLO messages
Select 1-hop neighbor node X covering most of 2-hop neighbors
Eliminate 2-hop neighbors covered by node X from the list
Is the number of 2-hop neighbors in the list 0?
NO
YES
Put 1-hop neighbors list for sending broadcast message
Neighbors received broadcast message
Am I in the list?
YES
NO
Halt
Continue Broadcasting
Stop

FIGURE 4.8: Working of Legacy Node

4.2.2 Load Balancing Module

A network monitoring and traffic analysis module collects node and connection statistics before turning on this module to deal with the congestion problem. To direct traffic to the optimum link, the controller considers connection parameters such as bandwidth, latency/delay, and connection utilization. The load balancing technique used to choose the optimum route is shown in Algorithm 4.2. It chooses the new rules for the new route to the new mesh nodes, which are the MAC and IP addresses. In the event that a new path is established by sending *FLOW_MOD* messages end-to-end, the controller floods all ports to the selected virtual nodes. In addition, it opens the client connection to enable packet delivery while simultaneously determining and keeping track of the network topology. The default parameters are set by the controller in order to create the path and to check the link states. The load balancing algorithm detects the network bottleneck and starts calculating new OpenFlow rules to reroute the traffic over new links when the client's bandwidth utilization reaches the "Th" level and there is traffic congestion. The controller then floods all ports (using OpenFlow FlowMod packets) in the direction of the targeted nodes along its path. The controller chooses the new optimal path, which consists of all accessible nodes N and the connections E connecting them, based on the graph topology. Then, in order to program the flow entries inside the software pipeline, it installs new OpenFlow rules in each node.

Algorithm 4.2: Load Balancing Algorithm

```
Data: Th, $\mathcal{N}$, $\mathcal{E}$
Result: Rerouting traffic to the optimal path
installDefaultFlowRules($\mathcal{N}$);
while Listening to LLDP packets do
    if TrafficCongestion(Th) then
        calculateNewOFRules($\mathcal{N}$, $\mathcal{E}$);
        FloodPackets($\mathcal{N}$);
        calculateOptimalPath(source, destination, $\mathcal{N}$, $\mathcal{E}$);
        if isBestPATH then
            InstallNewOFRules($\mathcal{N}$);
        else
            goto;
            calculateNewOFRules($\mathcal{N}$, $\mathcal{E}$);
        end
    else
        monitoring();
    end
end
```

4.2.3 Traffic Measurement Module

The controller can calculate traffic from the final SDN node on the forwarding path and ascertain the throughput of the forwarding path by using the counter that is used to provide packets' count of sample period. The probe messages are sent by the controller to the data plane layers. These signals travel via each node along the path before returning to the controller, making it possible to gauge time differences and contact delays.

4.2.4 Controller Failure Recovery Mechanism

Addressing problems with failure recovery and network robustness, distributed legacy protocols are used to swiftly fix momentary network issues. An SDN controller configures the routing paths while avoiding the shortcomings of the distributed classical protocols to guarantee long-term optimal network performance. Each hybrid SDN switch is configured with local agents to exchange routing information, and backup routes are built using a distributed classical OLSR. When a connection breaks down, these local agents use the OLSR routing information to reconnect the network and restore connectivity. Link failures in pure SDN networks can be quickly rectified in this fashion by leveraging the backup routes obtained from a traditional switch function.

The OLSR-to-OpenFlow (O_2O) module regularly regulates the controller's liveliness using standard OpenFlow methods. The O_2O enters an emergency status when the controller fails (for example, because of a hardware or communication issue), during which it deletes all of the rules the controller added to the flow table and the entire OLSR routing table, including all routes outside the control-subnet and the de- fault route advertised by the gateways. By doing this, OLSR gains a significant amount of control over mesh routing, but OpenFlow techniques are always used for forwarding. When the controller can be reached, the O2O exits emergency mode and deletes the flow table rules governing routes outside the control-subnet. This forces current data flows to deliver packet-in data units to the controller, which will then decide how to reroute them.

4.3 Simulation Model and Results Analysis

In WMNs, deploying topologies and locating nodes has become a complicated issue. The neighborhood and interference relationships can change depending on where the nodes and controller are placed [155]. The network topology based on SDN controllers, SDN hybrid nodes, and legacy nodes is taken into consideration by our proposed routing architecture 'Soft-Mesh' is illustrated by Figure 4.9. The

SDN controller can be physically located anywhere in the network, but it must be logically centralized. With SDN hybrid nodes and legacy nodes, we've assumed that the controller is one hop away. Another assumption is that SDN controller must have all or at least one SDN Hybrid node as its 1-hop neighbor, such assumption facilitates in establishing connection between controller and legacy nodes. The SDN Hybrid node, necessitates two interfaces and hence two subnets, one for control packets and the other for data packets. The controller has direct access to all nodes in the control subnet (one-hop), whereas the data network is a normal multi-hop network in which data must pass through numerous nodes to reach its destination. By employing OLSR information to reconstruct the whole topology of the network for route calculation, the controller would have knowledge about other nodes that are more than one hop distant. This implies that a strictly centralized solution is im- possible, hence SDN is combined with a distributed routing protocol to transport control messages from remote nodes to the controller.

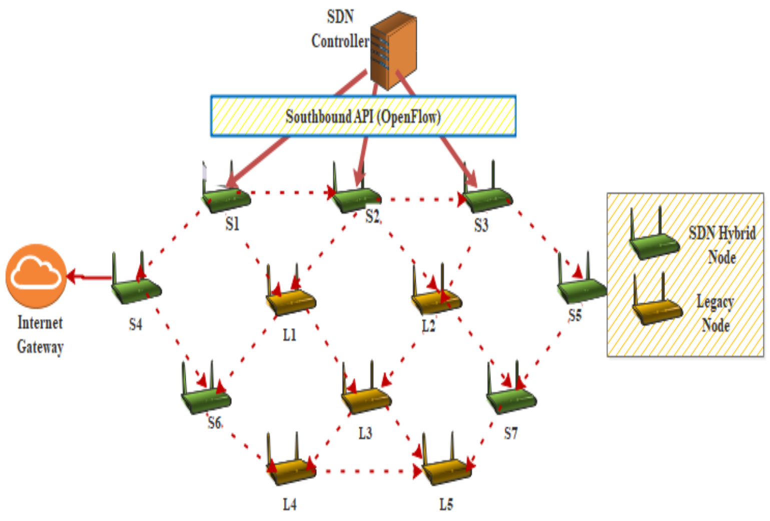


FIGURE 4.9: Topology for Proposed Routing Architecture

Nodes use the control subnet to broadcast their adjacent information tables to the controller on a regular basis, giving the controller a global view of the network's topology. Dijkstra's algorithm determines the shortest route (in terms of number of hops) from the source to the destination on a regular basis. The controller notifies participating nodes (source and intermediate nodes) of the modified rules if an optimal path is found. The graph topology, which includes all of the accessible nodes as well as the links connecting them, is used by the controller to construct the new optimum path. The software pipeline's flow entries are then programmed on each path using new OpenFlow rules that have been deployed. The nodes install the respective rules in their routing tables and the path is formed. An SDN architecture must include a controller since it has a comprehensive view of the whole network, including data plane SDN devices. It establishes a connection between these resources and management software, carrying out flow operations between the devices as specified by application rules [159]. SDN controllers are built with Java, Python, event-oriented, etc. As long as the controller uses the same Openflow version, the emulation platform theoretically may handle any kind of controller. The java based OpenFlow controller in our testbed, ODL, is used to create networking applications for topologies of 50, 100, 150, 200, and 250 nodes of SDN hybrid nodes and legacy nodes, respectively.

4.3.1 Simulation Model

The proposed routing architecture 'Soft-Mesh' uses Mininet-WiFi simulator as its simulation framework, which provides a simple network for creating OpenFlow applications. A controller, legacy nodes, and SDN hybrid nodes make up the topology under investigation. The OpenDaylight (ODL) SDN Controller [152] is utilized, and Soft-Mesh is implemented as a network application on the ODL controller, as shown by Figure 4.10 and 4.10. We compared Soft-Mesh's to that of the classic routing systems OLSR and BATMAN [153], which are considered to be the most stable of all traditional routing approaches for wireless mesh networks. Soft-Mesh design has been compared against wmSDN [103] and Hakiri [112] for hybrid systems, employing performance criteria such as Average UDP Throughput, Packet Loss Ratio, End-to-End Delay, and Routing Overhead. Simulations are carried out for topologies of 50, 100, 150, 200, and 250 nodes of SDN hybrid nodes and legacy nodes, respectively. Furthermore, the percentage of SDN hybrid nodes and legacy nodes in the network topology is configurable to produce three alternative simulation scenarios:

1. Scenario 01: This scenario consists of 50, 100, 150, 200, and 250 nodes of SDN hybrid nodes and legacy nodes, with 10% of SDN hybrid nodes and 90% of legacy nodes in the network topology, respectively. The scale of the topology has also been kept configurable, with 500m*500m topologies for 50 and 100 nodes and 1000m*1000m topologies for 150, 200, and 250 nodes

2. Scenario 02: This scenario consists of 50, 100, 150, 200, and 250 nodes of SDN hybrid nodes and legacy nodes, with 25% of SDN hybrid nodes and 75% of legacy nodes in the network topology, respectively. The scale of the topology has also been kept configurable, with 500m*500m topologies for 50 and 100 nodes and 1000m*1000m topologies for 150, 200, and 250 nodes

3. Scenario 03: This scenario consists of 50, 100, 150, 200, and 250 nodes of SDN hybrid nodes and legacy nodes, with 50% of SDN hybrid nodes and 50% of legacy nodes in the network topology, respectively. The scale of the topology has also been kept configurable, with 500m*500m topologies for 50 and 100 nodes and 1000m*1000m topologies for 150, 200, and 250 nodes.

To obtain more accurate findings, simulations are carried out numerous times, each with around 10 tests for the specified number of nodes, while using the Random walk mobility model to explore support for node mobility in the algorithm. Standard deviations for the required performance measures were also taken into account to observe how the outcomes varied. The mac80211_ hwsim is used for the MAC Layer, with an operating time of 17ms for an AP, 63ms for the station, 10ms for two nodes, and 350ms for the AP and stations. Constant bitrate (CBR) traffic flows are used. The mobility model random walk follows the mobile node's speed, which is between 10 and 50 m/s. The parameters used in our simulations are presented in table 4.1.

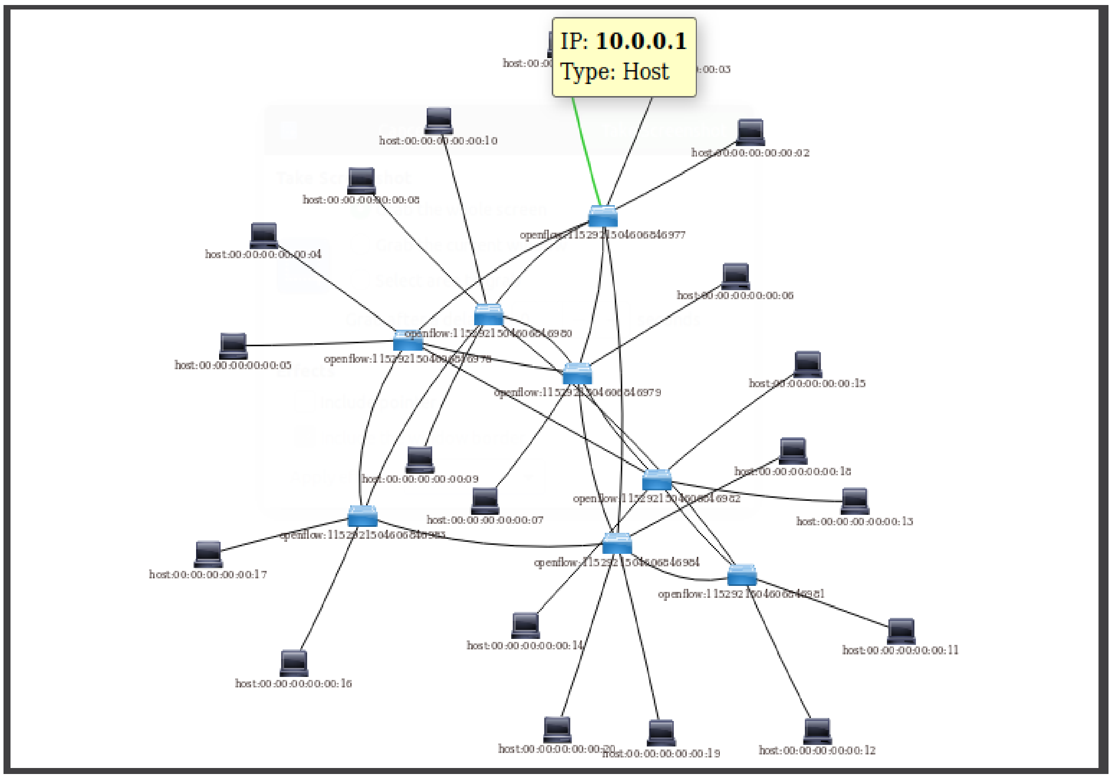


FIGURE 4.10: Mininet-WiFi Simulation of Soft-Mesh using ODL Platform

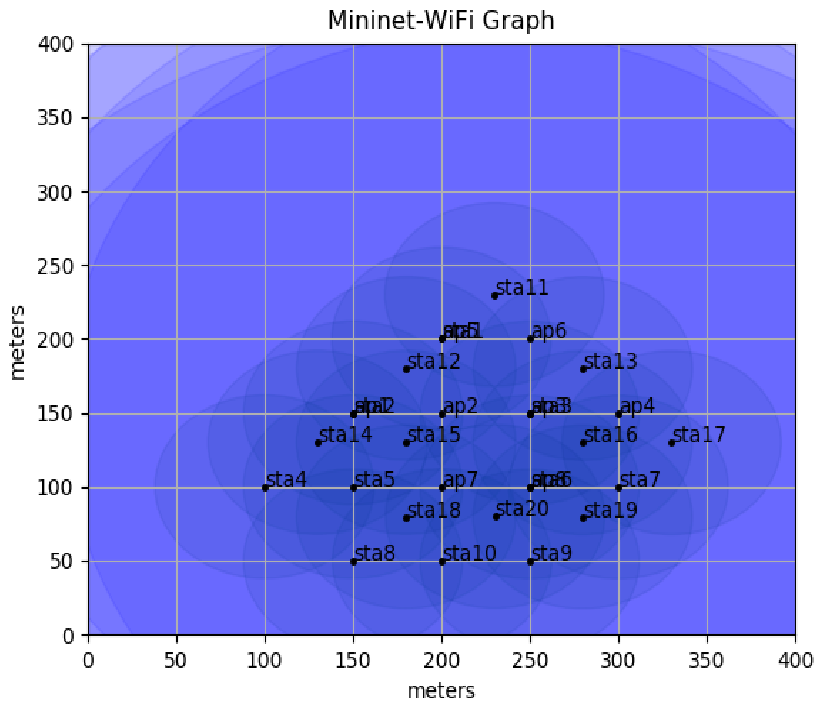


FIGURE 4.11: Mininet-WiFi Simulation of Soft-Mesh with Mobile Nodes

TABLE 4.1: Simulation Parameters of Soft-Mesh Architecture

Parameters	Value
Simulator	Mininet-WiFi
Protocols	OLSR, BATMAN, wmSDN, Hakiri, Soft-Mesh
Simulation Time	100 seconds
PHY and MAC model	802.11n
No of nodes	50, 100, 150, 200, 250
Proportion (SDN hybrid nodes% + Legacy nodes%)	10%+90%, 25%+75%, 50%+50%
Topology Size	500m * 500m (50 and 100 nodes), 1000m * 1000m (150, 200 and 250 nodes)
Mobility Model	Random Walk
Mobility Speed	Min 10 m/s, Max 50 m/s
Traffic rate (Mbps)	5 Mbps
Traffic Type	UDP/TCP
Traffic flows Characteristics	Constant Bit Rate (CBR)
Packet Size (byte)	1500 bytes
HELLO emission interval	2 seconds
Controller-Node message emission	3 seconds
Timeout for unused routes	10 seconds
Topology reconstruction interval	5 seconds

4.3.2 Simulation Results and Analysis

This section compares the proposed routing architecture Soft-Mesh to existing traditional and hybrid SDN/IP routing architectures in terms of performance parameters such as average UDP throughput, end-to-end delay, packet drop ratio, and routing overhead. The OLSR and BATMAN routing systems are simulated as they are considered relatively more stable than any other standard routing approaches for wireless mesh networks.

4.3.2.1 Throughput

The throughput of a routing design is a crucial performance indicator. We use UDP traffic exchange between end users and set the packet size to 1,500 bytes in order to evaluate the performance and robustness of our suggested architecture in terms of throughput. Each node delivers data at a rate of 5 Mbps, as assumed. All nodes are considered in a full mesh architecture, and the data traffic is determined by averaging out the different packet forwarding sections. The nodes are placed in various locations, and the controller side traffic is monitored to evaluate the effects of OLSR forwarding and OpenFlow. We assume that the controller has already pushed down and installed the flow rules in the OpenFlow tables of the underlying mesh nodes. Therefore, the packets arriving at a certain ingress port of a node are immediately forwarded to its physical output port in order to allow the packets to reach the future hop. The following is how the throughput metric is extracted:

$$Throughput = \frac{TotalRecievedPayload}{TimeLastPacketReceived - TimeFirstPacketTransmitted} \tag{4.1}$$

Figures 4.12, 4.13, and 4.14 displays the throughput as determined by the Iperf measurement tool [154] for the aforementioned three scenarios, based on varying number of network nodes in the topology, including the fraction of SDN hybrid nodes and legacy nodes. To guarantee the consistency of the findings, we conducted the trials several times. Each run contains one of the following three forms of traffic: end-user UDP/IP data traffic; OLSR forwarding traffic; and OpenFlow control traffic. The maximum estimated throughput for scenarios 01, 02, and 03, respectively, is limited to 9 Mbps, 15 Mbps, and 17.5 Mbps. However, the typical throughput is very nearly 14 Mbps. The volume of OLSR traffic, thread priorities, CPU interruptions, data plane to control plane encapsulation, and the transmission of OpenFlow control information over the network are just a few of the factors that might cause the throughput to decrease. The average throughput falls closer to 7 Mbps with such unpredictable traffic, which we consider to be a decent deal.

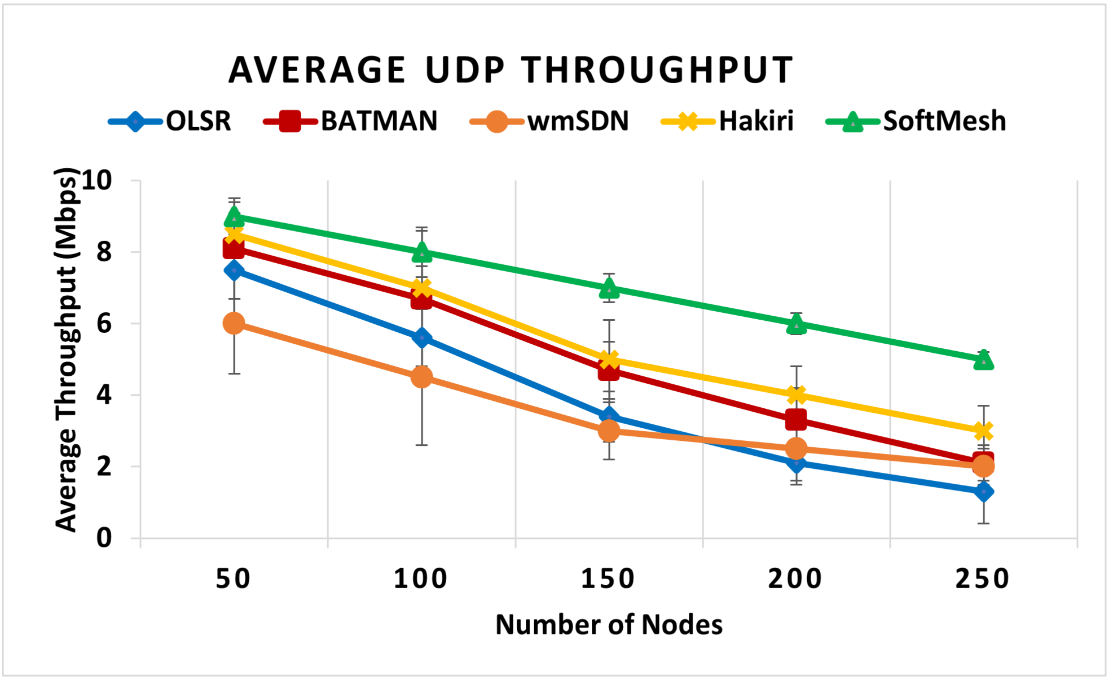


FIGURE 4.12: Average UDP Throughput - Scenario 01 (10% of SDN hybrid nodes and 90% of legacy nodes)

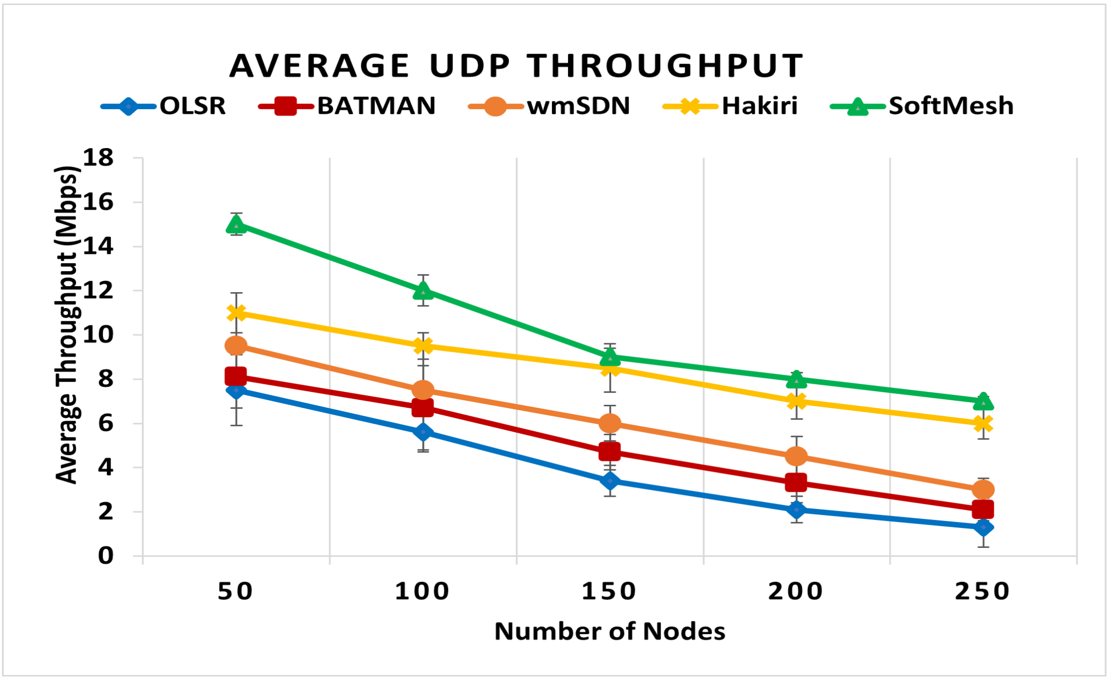


FIGURE 4.13: Average UDP Throughput - Scenario 02 (25% of SDN hybrid nodes and 75% of legacy nodes)

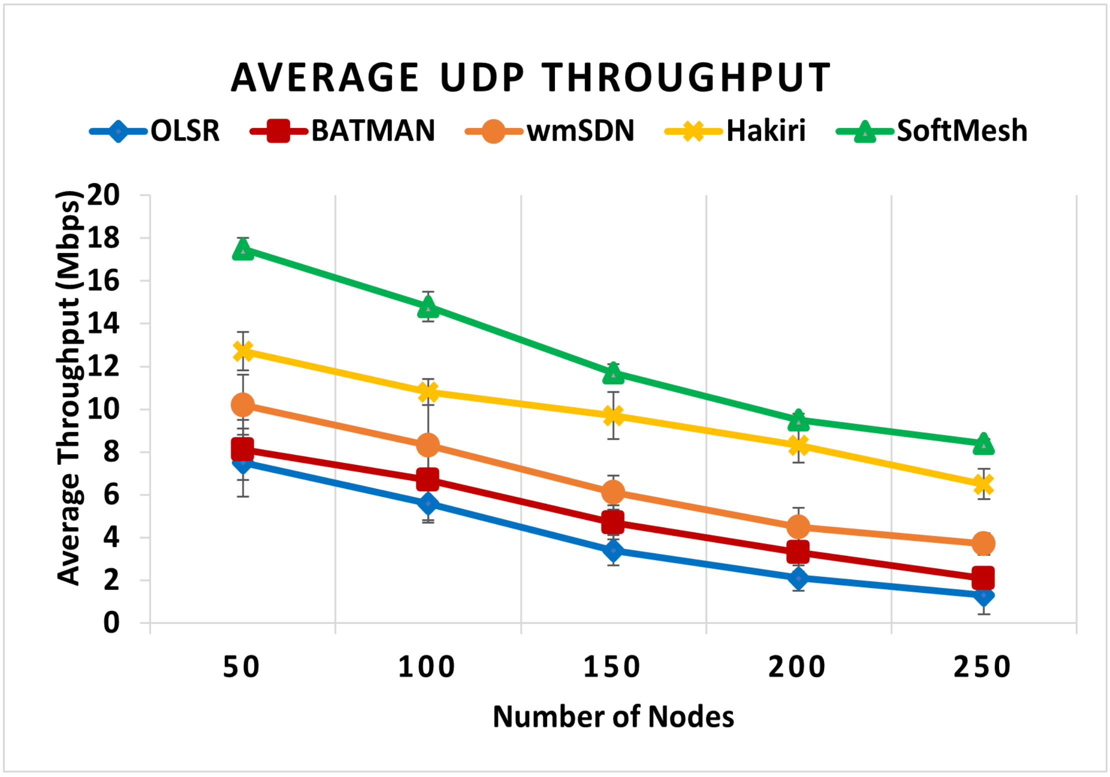


FIGURE 4.14: Average UDP Throughput - Scenario 03 (50% of SDN hybrid nodes and 50% of legacy nodes)

The simulation results clearly show that the proposed routing architecture Soft-Mesh's outperforms conventional protocols and other existing hybrid alternative solutions, this is mainly because of low latency. The fundamental properties of the SDN paradigm are directly linked to this performance improvement. The network's topology is continuously monitored, allowing for quick identification of changes and recalculation of new routes for the network's continuing traffic flows. The necessary rules are sent to the nodes and installed along the measured path without flooding the network if the recalculation improves the node's identification. In contrast, network flooding serves as the main mechanism for spreading information about topology change in distributed protocols. As the process of flooding causes significant delays and lengthens the time it takes for the routing protocols to converge. In short, Soft-Mesh routing reacts to topology changes more quickly than standard protocols and ensures the best paths (in terms of hop count) are taken at all times. This results in the best performance in terms of UDP throughput. It is possible to fine-tune the flow distribution using SDN and WMN, such as via load-balancing between alternative routes.

4.3.2.2 Packet Drop Ratio

Packet loss is a useful indicator of a routing protocol's effectiveness in maximizing internode interface exchanges and to check application's dependability. Data about the lost packets is gathered as a result of the absence of a routing rule. To provide an in-depth analysis of the average relative error in the throughput, we estimated the per-flow packet loss by polling the flow data, assuming a relationship between the link packet loss and the throughput. The packet loss can be calculated by subtracting the average throughput of the edge node on the client side from the edge node on the server side. Figures 4.15, 4.16 and 4.17 depict the packet drop ratio for the aforementioned three scenarios, with OLSR having the highest dropped packet rate and thus the slowest convergence phase. BATMAN follows OLSR due to its relatively long update intervals and lack of an overhead optimization mechanism. Because of its buffering function, BATMAN has a lower packet drop ratio than OLSR. The number of nodes increases the packet loss. The ability of route protocols to optimize routes and the overhead they incur, two crucial considerations, will therefore directly affect the rate of packet loss. However, due to collisions and erroneous packets, interference frequently has a direct effect on the rate of packet loss.

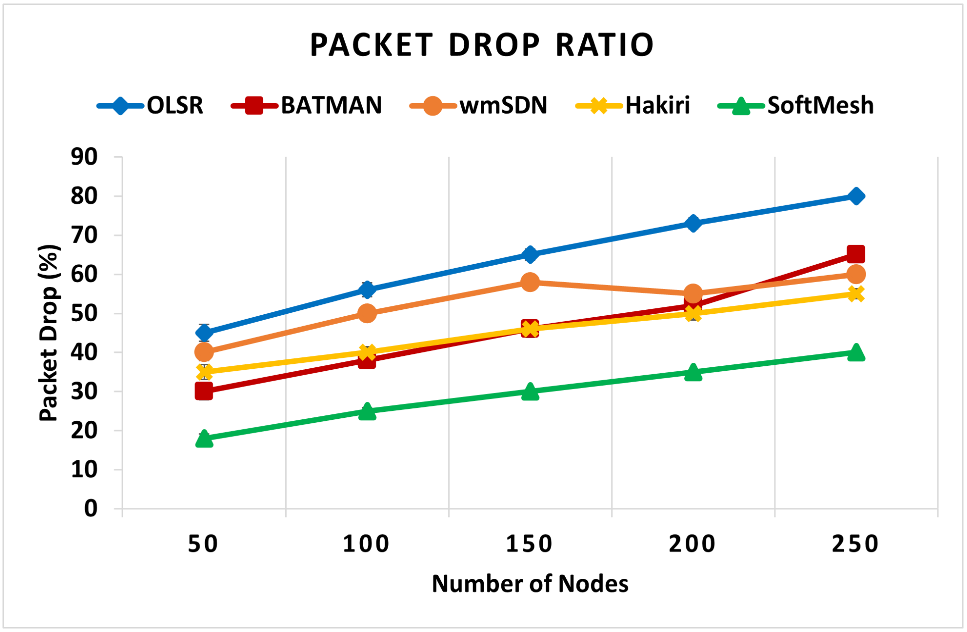


FIGURE 4.15: Packet Drop Ratio - Scenario 01 (10% of SDN hybrid nodes and 90% of legacy nodes)

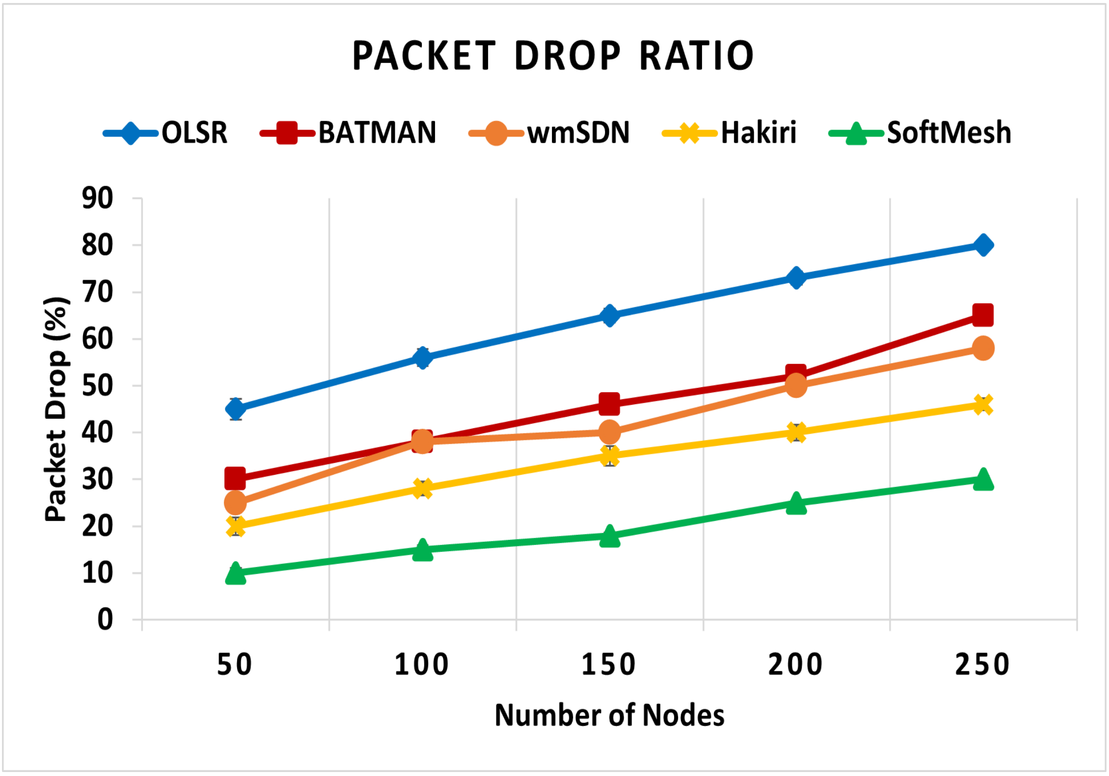


FIGURE 4.16: Packet Drop Ratio - Scenario 02 (25% of SDN hybrid nodes and 75% of legacy nodes)

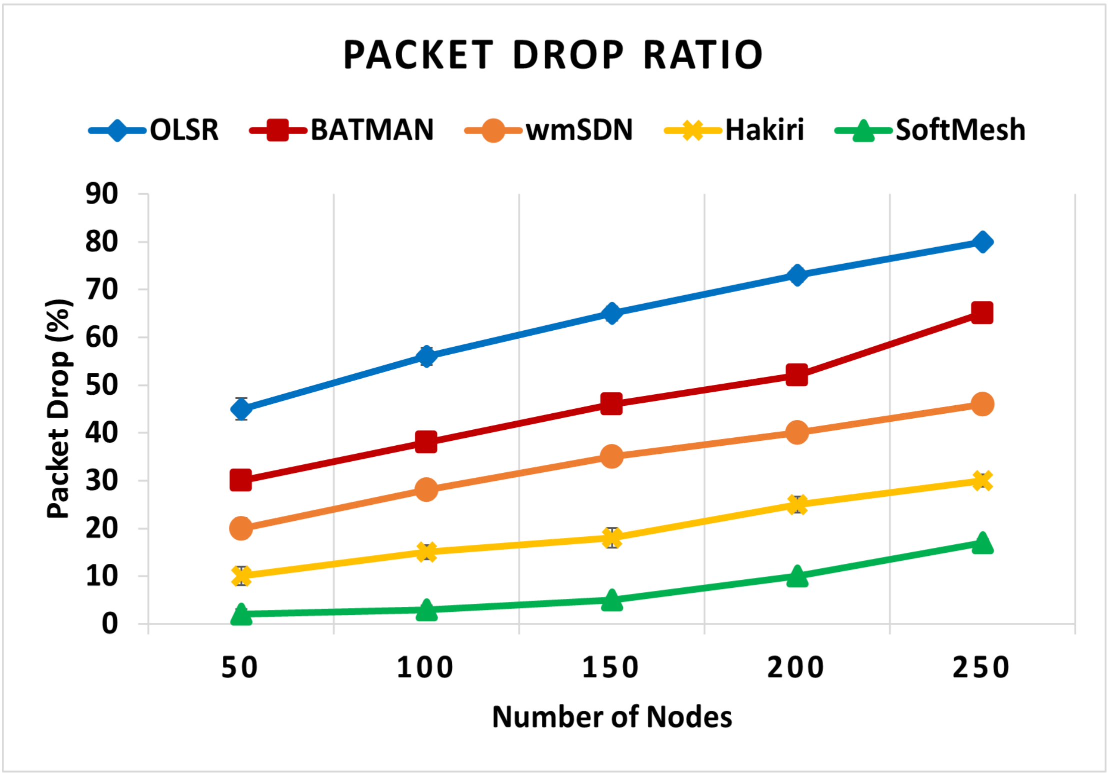


FIGURE 4.17: Packet Drop Ratio - Scenario 03 (50% of SDN hybrid nodes and 50% of legacy nodes)

The average error is extremely near to 1%, according to the information on packet loss. The average packet loss is calculated by subtracting the difference between the packet counters in the edge nodes between the client and the server. These measures offer a sufficient evaluation of service degradation. The present OpenFlow design does not support QoS service differentiation and only supports per-class packet categorization, scheduling, and forwarding. As a result, traffic priority does not provide any protection for computational flows. The outcomes of the simulation show how well our technology supports SDN-based communication in the context of enhancing WMN performance.

4.3.2.3 End-to-End Delay

The end-to-end delay gives the predictability of the response times that can be used to measure real-timeliness of an application. This delay is depicted as the amount of time it takes for a packet to go from the source mesh node to the destination node. We repeated this experiment multiple times while maintaining the standard latency. One-way delays are difficult to measure because packets experience a variety of network delays, such as collection delays, queuing delays, transmission delays, and propagation delays. As a result, we calculated the one-sided delay and estimated the round-trip time (RTT) by assuming half of the RTT. In addition, we determined how long it would take to deliver a packet to the controller before it would get its node unreachable. The controller periodically broadcasts route request messages in an effort to address the issues of mobility, node failure, and connection failure. The findings demonstrate that in 50 and 100 node network topologies, the end-to-end delay of present hybrid and Soft-Mesh architecture is higher. End-to-end delay is higher in legacy routing protocols than in SDN networks for network topologies with 150, 200, and 250 nodes, respectively. In traditional networks, after the routing table has been defined, each packet must be questioned and forwarded. The network scale grows along with the size of the routing database, which slows down querying and forwarding. In SDN, the data packet that the node receives is first transmitted to the controller, after which the controller pushes the flow entries to the node, the packets can then be forwarded using the flow table query. Since the OpenvSwitch flow table scale is smaller than the node routing table size in networks with a similar architecture, the OpenvSwitch forwarding speed will be higher when the network scale is high. After the controller updates the node's Open-Flow rules, the controller-switch delay decreases to roughly 3 ms and goes to approximately 10 ms during startup. To check whether an idle control connection might be a sign of a break in controller-switch communication, just the OpenFlow maintain alive messages are now being exchanged. The network receives a new mobile client at time 40 seconds, but neither the controller nor the switch are aware of the client's forwarding policies. In order to set new forwarding rules for packets coming from that client, they begin communicating. At 80 seconds, the same event takes place. In each of those situations, the controller-switch delay is restricted to less than 10 ms in all other situations and to 15 ms during setup. A network bottleneck is not caused by the controller-switch delay. The latency is typically near to zero when setup traffic is not injected into the network. Each time new OpenFlow rules are negotiated between the controller and the switches, the latency increases to about 30 ms. The maximum delay in any case is 20 milliseconds. In a manner similar to that, the experiment demonstrated that real-time applications require the preservation of predictable delays, or a lower jitter, of approximately 2.5ms.

$$End-to-EndDelay = \frac{TotalDelay}{TotalReceivedPackets} \tag{4.2}$$

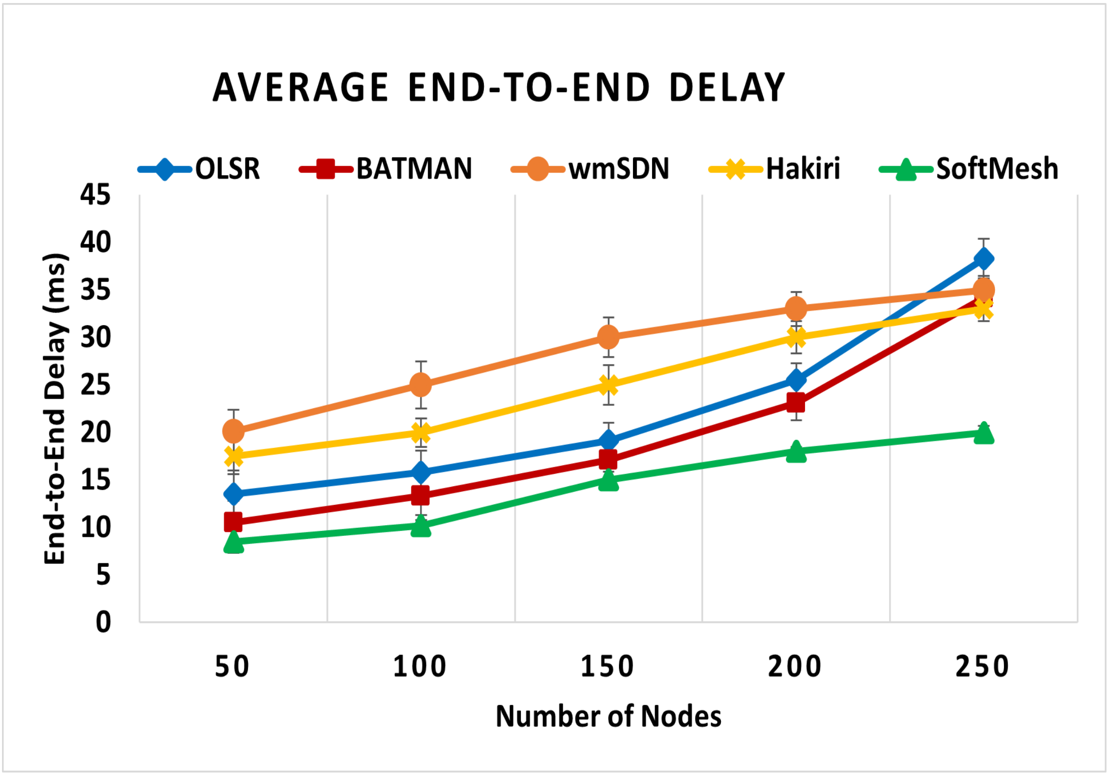


FIGURE 4.18: Average End-to-End Delay - Scenario 01 (10% of SDN hybrid nodes and 90% of legacy nodes)

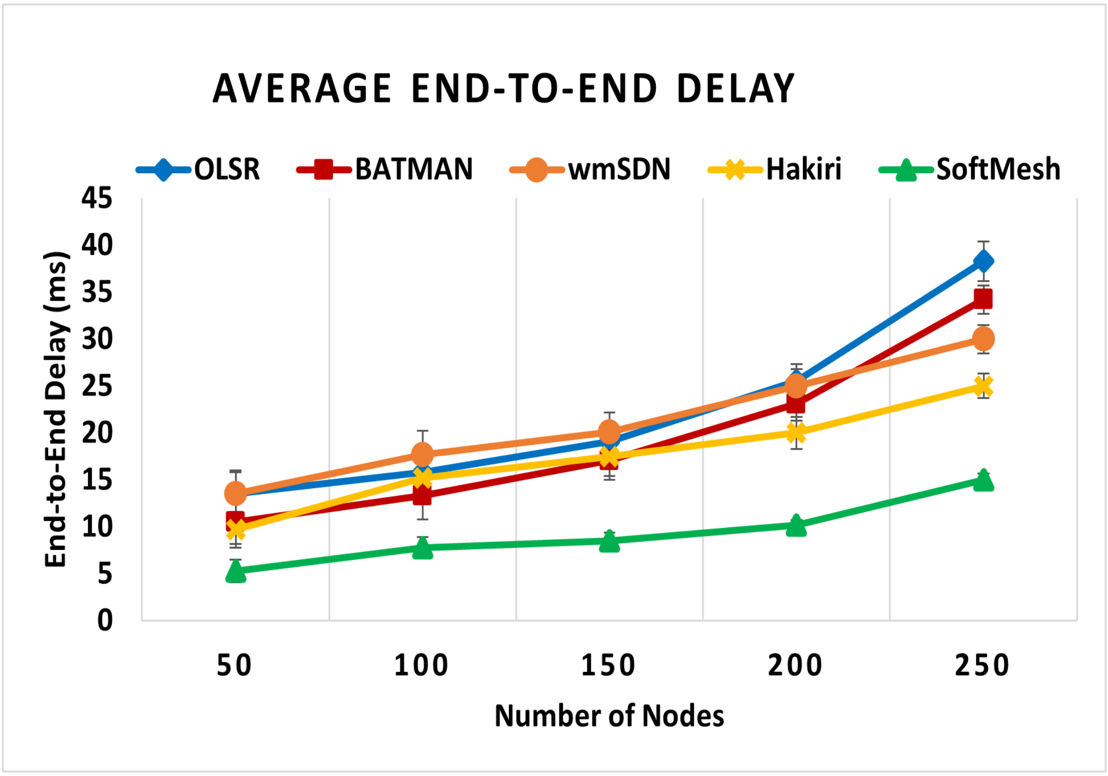


FIGURE 4.19: Average End-to-End Delay - Scenario 02 (25% of SDN hybrid nodes and 75% of legacy nodes)

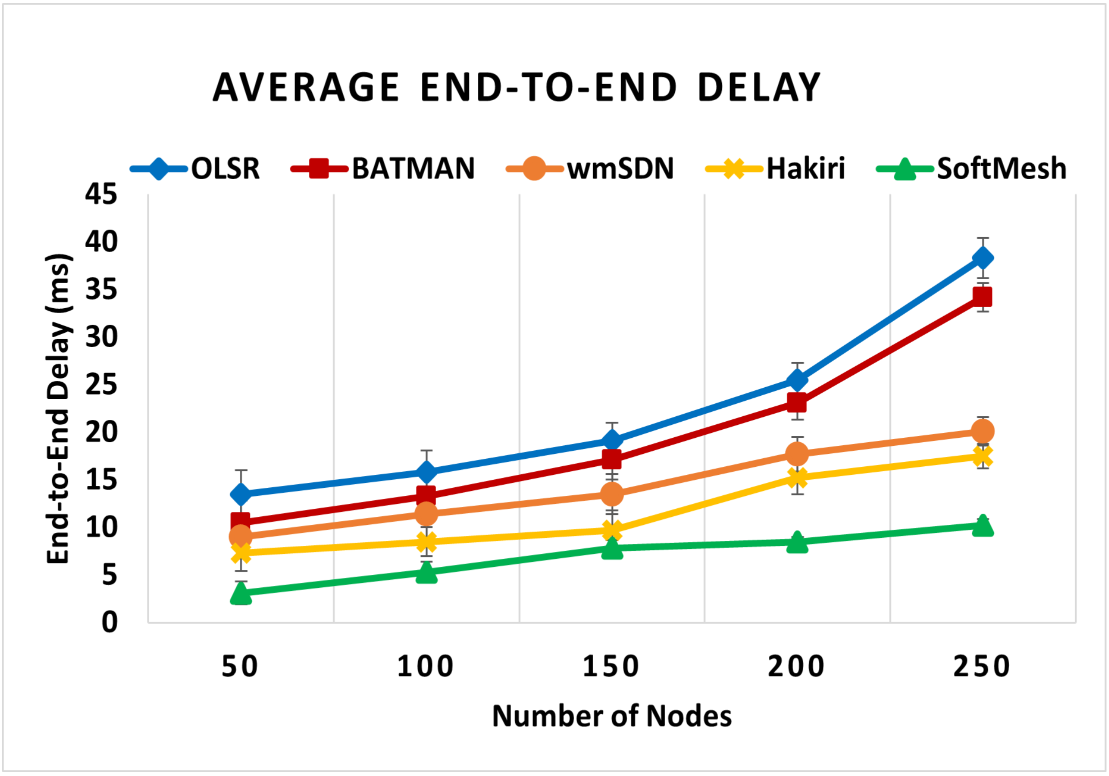


FIGURE 4.20: Average End-to-End Delay - Scenario 03 (50% of SDN hybrid nodes and 50% of legacy nodes)

Simulations are conducted with varying number of nodes for all the scenarios mentioned in Section 5.3, in order to test performance metric end-to-end delay. Additionally, we contrasted our solution's delay with classical routing protocol OLSR and BATMAN, wmSDN and Hakiri for hybrid architectures. We repeated this test numerous times, averaging the results each time. Additionally examined and explicitly compared to the number of broken links was the controller-switch latency for reconnecting them in the event of failure. The controller attempts to establish a connection with all switches using discovery messages in order to determine the fastest path to all underlying SDN nodes. Figures 4.18, 4.19 and 4.20 demonstrate the simulation results, that clearly show that as compared to other traditional and existing hybrid routing approaches, our approach exhibits lower delay. Our technique has a maximum 20ms delay, which is nearly 50% shorter than that of the other approaches, compared to other traditional approaches OLSR and BATMAN, and existing hybrid approaches wmSDN and Hakiri, respectively. The same pattern was also seen when we compared the number of broken wireless links to the delay after the controller re-connection. As the number of hops between switches increases, our solution performs better than the other approaches. The SDN nodes have a greater connection establishment to the controller than in the other existing solutions.

4.3.2.4 Routing Overhead

As demonstrated in Figures 4.21, 4.22, and 4.23, the global routing burden that routing protocols experience grows proportionally with the size of the network. In addition, OLSR, while offering the highest network performance among conventional routing algorithms, has a high overhead than BATMAN protocol. Better outcomes are possible even when the controller and hosts exchange messages at a high rate because to the centralized SDN operation, which eliminates the need for flooding required by the OLSR protocol. As a result, the impact of interference and convergence time will be reduced by employing the SDN technique and having less overhead.

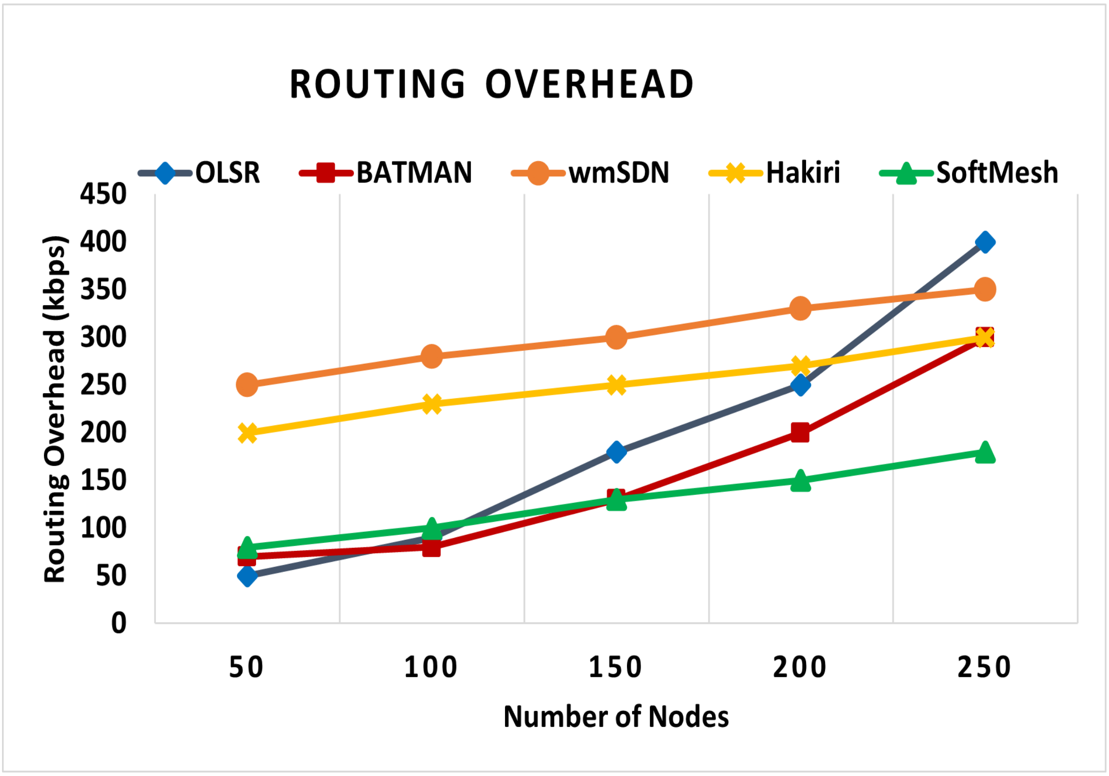


FIGURE 4.21: Routing Overhead - Scenario 01 (10% of SDN hybrid nodes and 90% of legacy nodes)

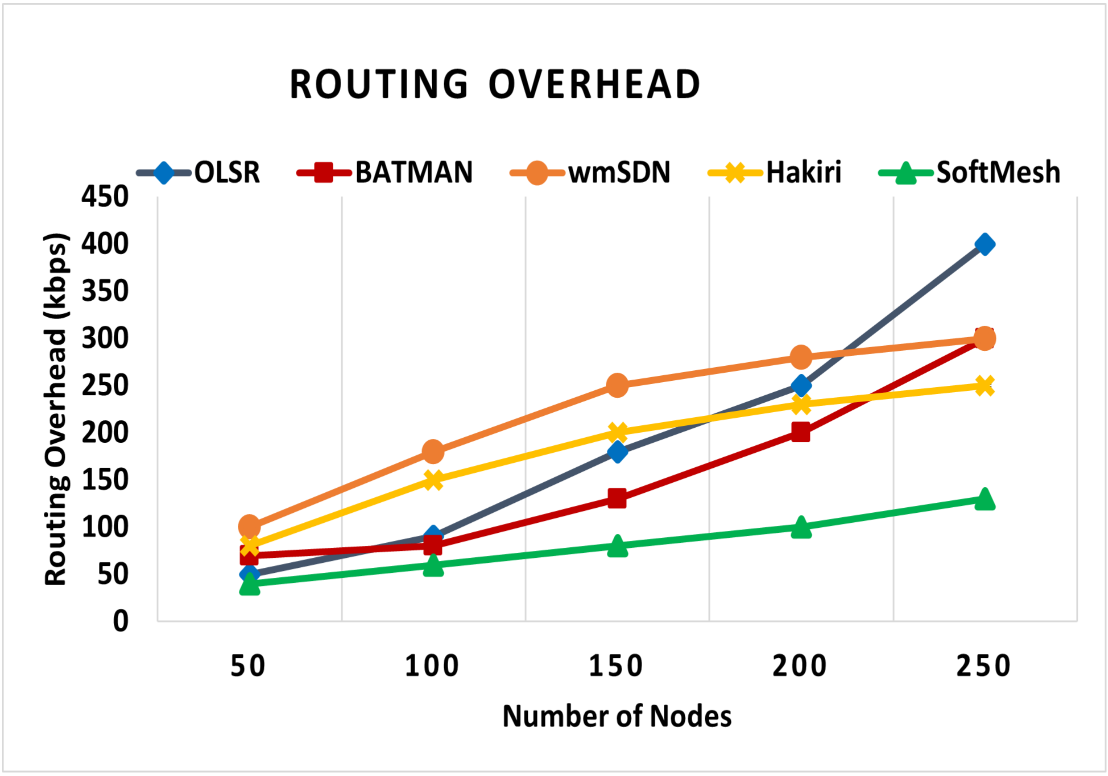


FIGURE 4.22: Routing Overhead - Scenario 02 (25% of SDN hybrid nodes and 75% of legacy nodes)

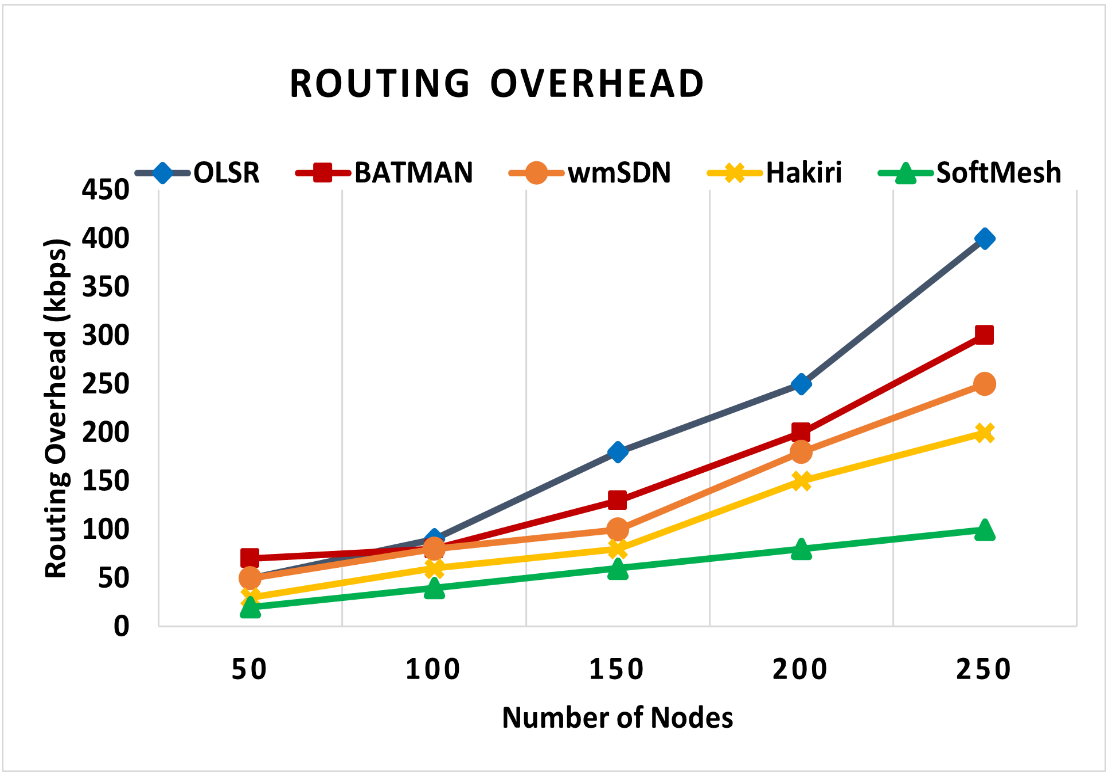


FIGURE 4.23: Routing Overhead - Scenario 03 (50% of SDN hybrid nodes and 50% of legacy nodes)

We evaluate the performance overhead of each mesh node to OLSR for IP routing and OpenFlow forwarding when applying our hybrid routing scheme. Every gateway has an Internet connection, and each node's routing table receives the default route, 0.0.0.0/0, via OLSR. Each node also has access to OpenvSwitch, which can forward OpenFlow messages and is bridged to IP forwarding via the br network interfaces. In this case, it is possible to perform flow-based forwarding operations using our hybrid routing approach, while still routing those flows between different mesh nodes using OLSR to more efficiently utilize IP networks' capacity to route packets to the shortest path between the source and destination.

After the initiation phase, OpenFlow generates control traffic when new rules are added by the SDN controller to the pertinent node. The OLSR traffic remains steady as expected, but as new rules are introduced, the OpenFlow traffic increases. OpenFlow increases control traffic by about 4 Mbps, and overall traffic is six times higher than when OLSR is used as the routing protocol. The controller's ability to inject new flow entries into the node means that all of the new OpenFlow rules have been installed, which reduces the amount of OpenFlow control traffic. Unlike OLSR and OpenFlow, our method does not add any extra control flow while installing new rules. Because of this, our method does not raise node overhead. Estimating the overhead of the extra infrastructural components is essential. Thus, we assessed the volume of control traffic that was sent between the controller and the underlying nodes in order to calculate the controller overhead. We also contrasted this traffic with data traffic exchanges, in which the controller updates the flow tables of the nodes.

The simulations are carried out roughly 10 times, and the evaluation is based on the average values. There were three separate matching actions in the controller traffic that was gathered: OpenFlow packets, Ethernet packets (like ARP), and data packets (i.e., TCP packets). Wireshark is used to record the controller traffic passing across the nodes, while Tcpdump is employed for analysis. Control around 15% of the overall traffic exchanged in the wireless network is OpenFlow traffic, whereas 75% of it is data traffic and 10% is Ethernet traffic. To provide host reachability between remote hosts, Ethernet traffic must be exchanged during the startup phase. In reality, the early hosts broadcast *PACKET_OUT* messages to every node in the network using ARP requests. To determine the source port mapping, the nodes analyze these requests. After the controller gets ARP responses, it becomes aware of all Ethernet addresses and the source MAC address will be linked to the port. The ports of the underlying node can now receive *FLOW_MOD* messages from the controller. When compared to TCP data traffic, the control overhead of broadcasting OpenFlow messages is minor, being around twice that of Ethernet traffic. The overhead brought on by the control traffic is thus negligible.

4.3.3 Evaluation of Controller Failure Handling

We carried out simulations to address controller failure also, as demonstrated by Figure 4.24. Its goal is to demonstrate the efficiency of the OLSR-to-OpenFlow (O_2O) module (shown by Figure 4.25) in handling such situations in addition to continuing the routing process in the event of controller failure.

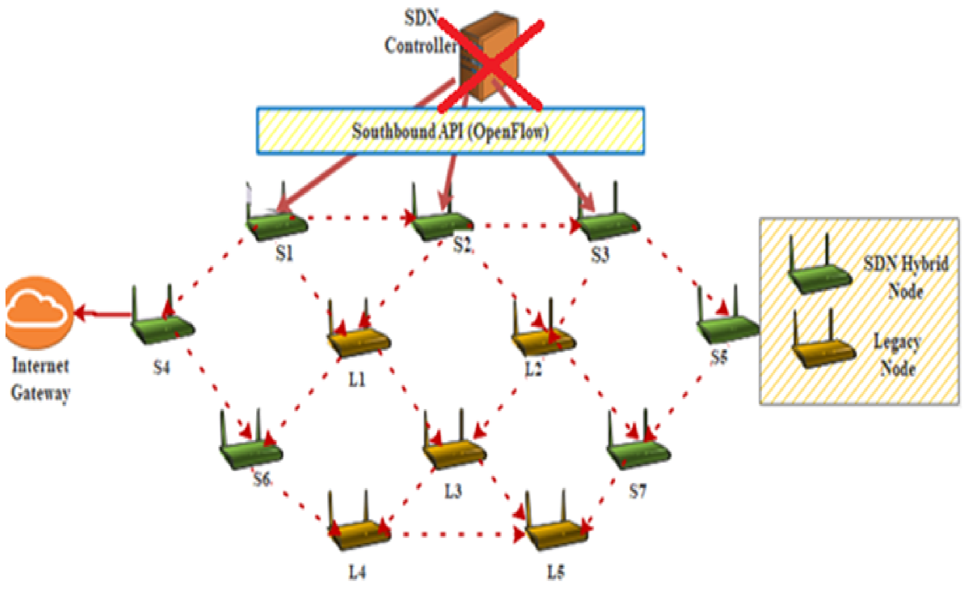


FIGURE 4.24: Controller Failure in Soft-Mesh

For this, we take into account the network topology in Scenario 02 (25% of SDN hybrid nodes and 75% of legacy nodes) and Scenario 03 (50% of SDN hybrid nodes and 50% of legacy nodes). TCP flows are selected for testing and sent to

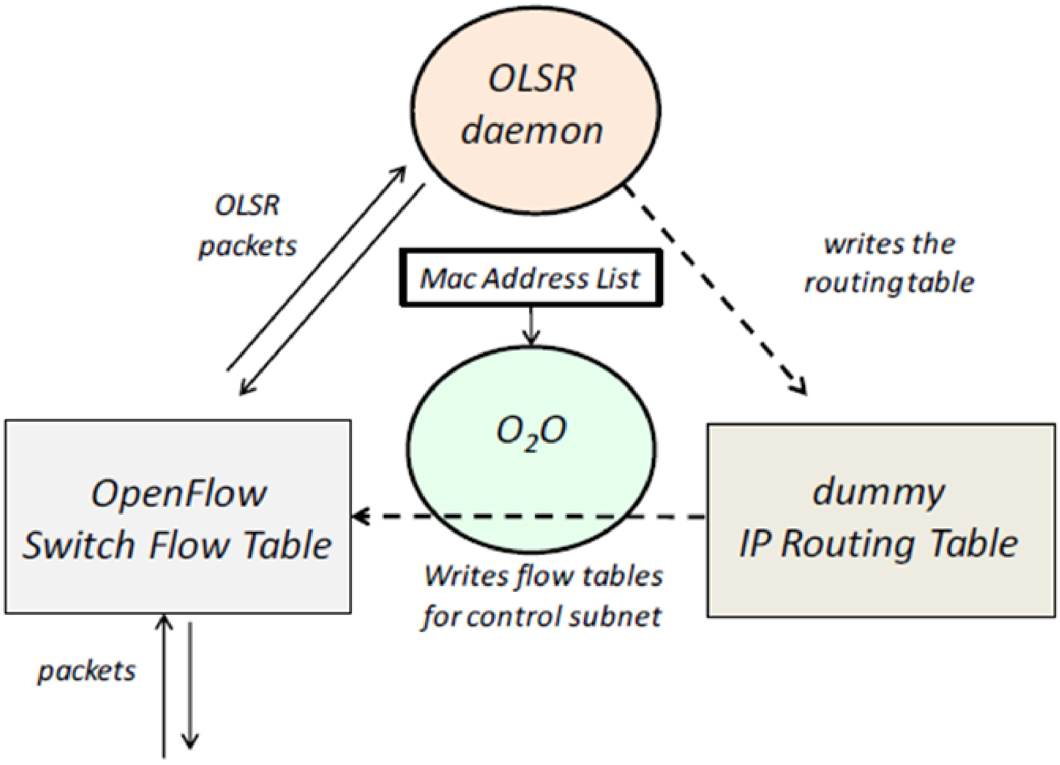


FIGURE 4.25: OpenFlow and OLSR Interaction

clients on the Internet via a public server source. The flows begin at time 0, and during the simulation's 100-second running time, the controller experiences a 40-second outage that we imitate by terminating the relevant ODL process. The O2O module dumps all of the OLSR routes, including the default route of 0.0.0.0/0, transmits the flows, and the corresponding throughputs are reduced by half during the controller outage. It is crucial to keep in mind that the timeout we set up to make sure the controller is functioning properly results in a delay of roughly 10 seconds between the controller failing and the O2O module actually responding. When the outage time for the controller is ended, the O2O module detects the controller's presence and removes any previously added OLSR routes that don't impact the control subnet from the flow tables. As a result, a flow setup phase takes place, during which the controller assigns flows to SDN hybrid nodes, and performance levels return to those attained at the test's beginning, this also takes the delay of approximately 10 seconds in the resumption of controller.

Furthermore, it is evident from the results as displayed in Figures 4.26, 4.27, 4.28

and 4.29 average throughput behavior, packet drop ratio, end-to-end delay, and routing overhead for the Scenario 02 that comprised of 25% SDN hybrid nodes and 75% legacy mesh nodes. It is noteworthy that, being the distributed routing protocols the traditional OLSR and BATMAN do not use controller, thus not included in this particular simulation scenario. Furthermore, among the hybrid SDN routing protocols, Hakiri [104] does not address the issue of controller failure. Thus, simulation results show that our proposed approach provides extremely similar results to that of OLSR with a little increase due to the rules already installed by controller at the setting up of network. Such an improvement is brought about by the cohabitation of OpenFlow and OLSR in the SDN hybrid nodes architecture.

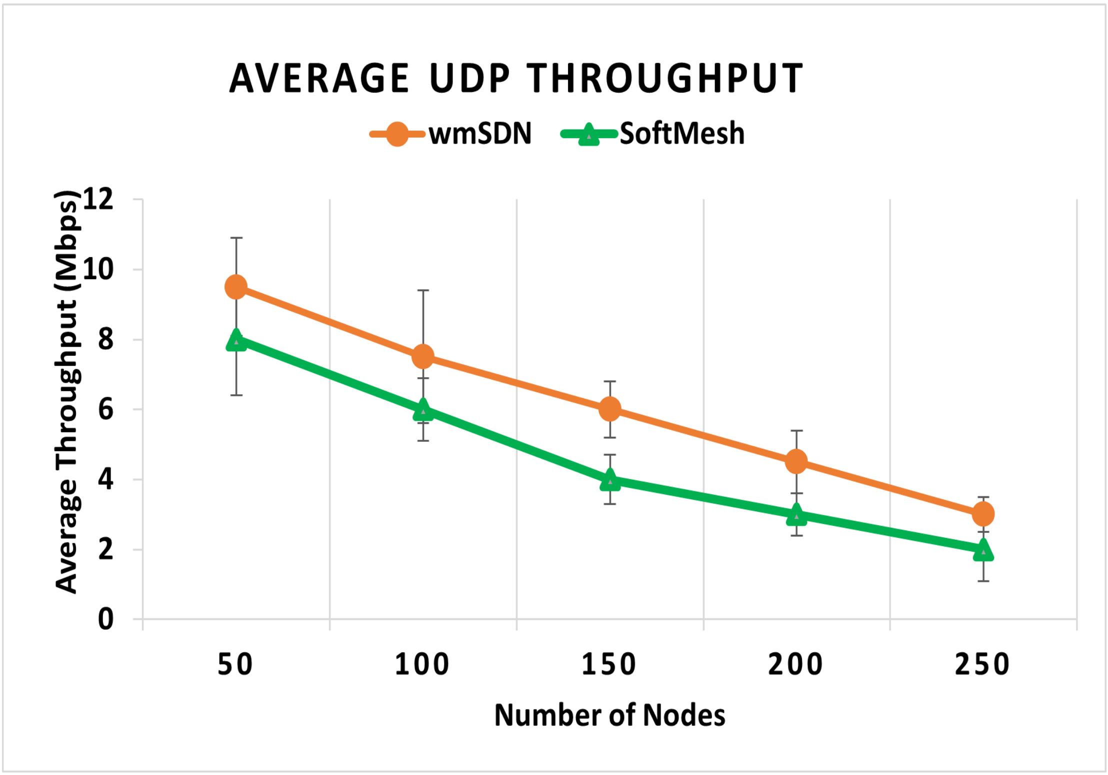


FIGURE 4.26: Average UDP Throughput - Scenario 02 (25% of SDN hybrid nodes and 75% of legacy nodes)

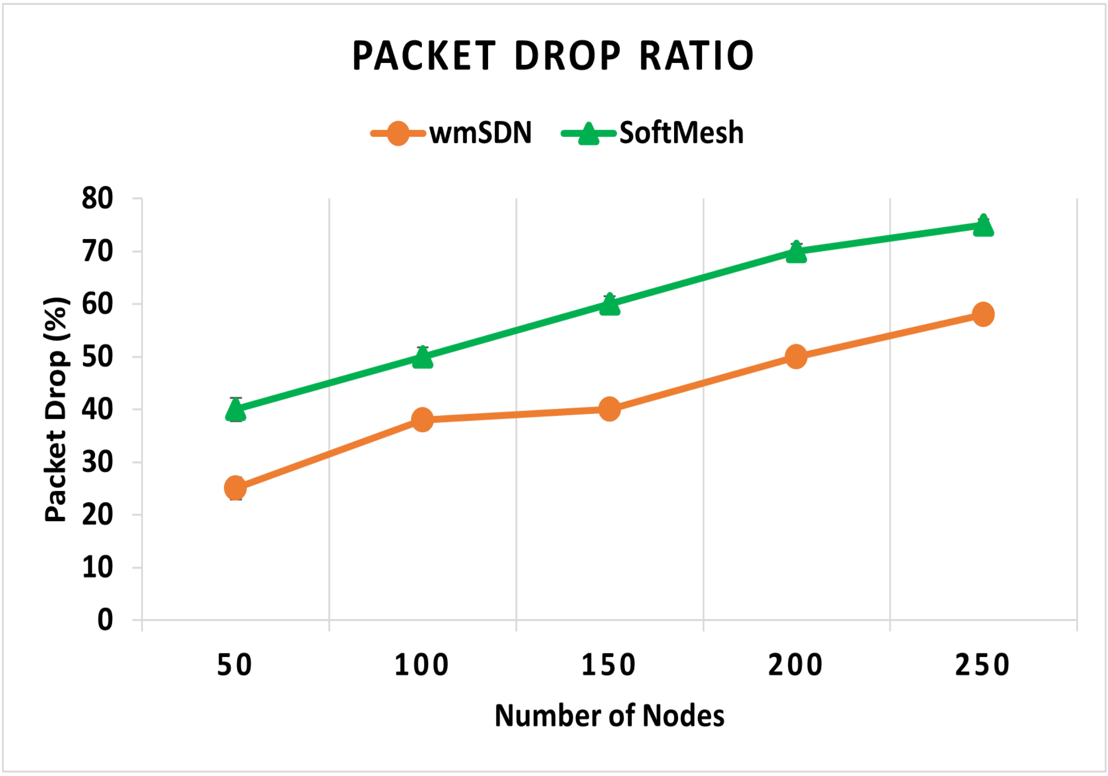


FIGURE 4.27: Packet Drop Ratio - Scenario 02 (25% of SDN hybrid nodes and 75% of legacy nodes)

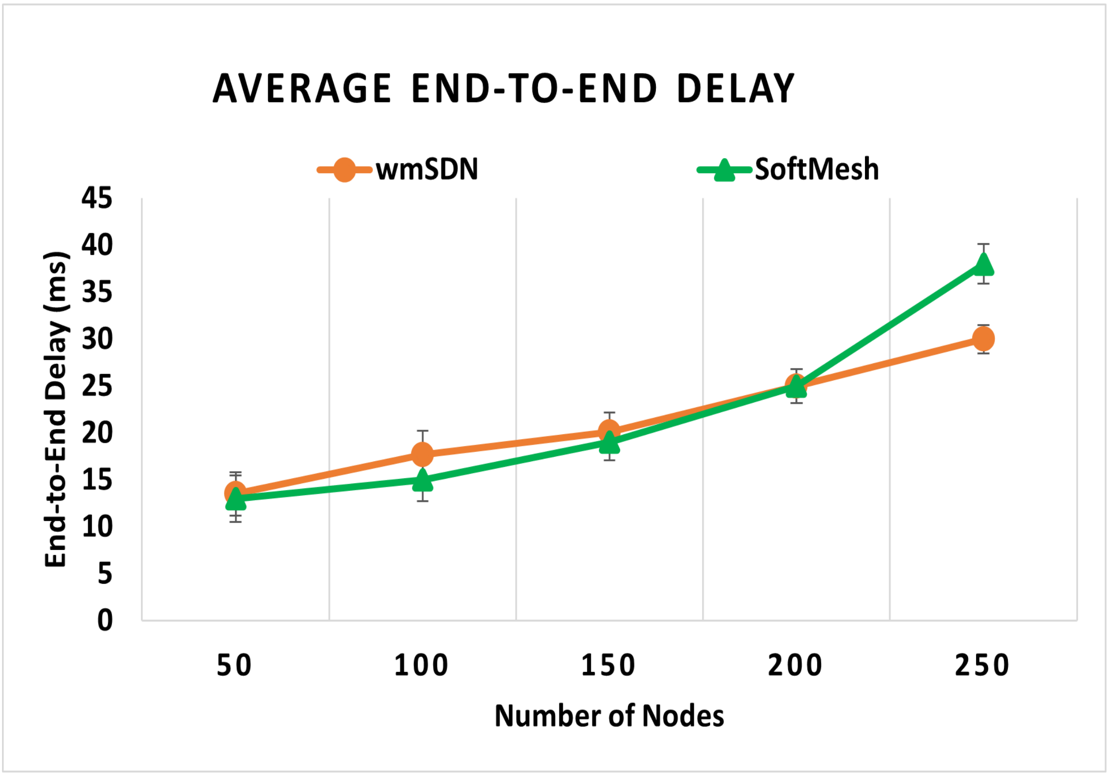


FIGURE 4.28: Average End-to-End Delay - Scenario 02 (25% of SDN hybrid nodes and 75% of legacy nodes)

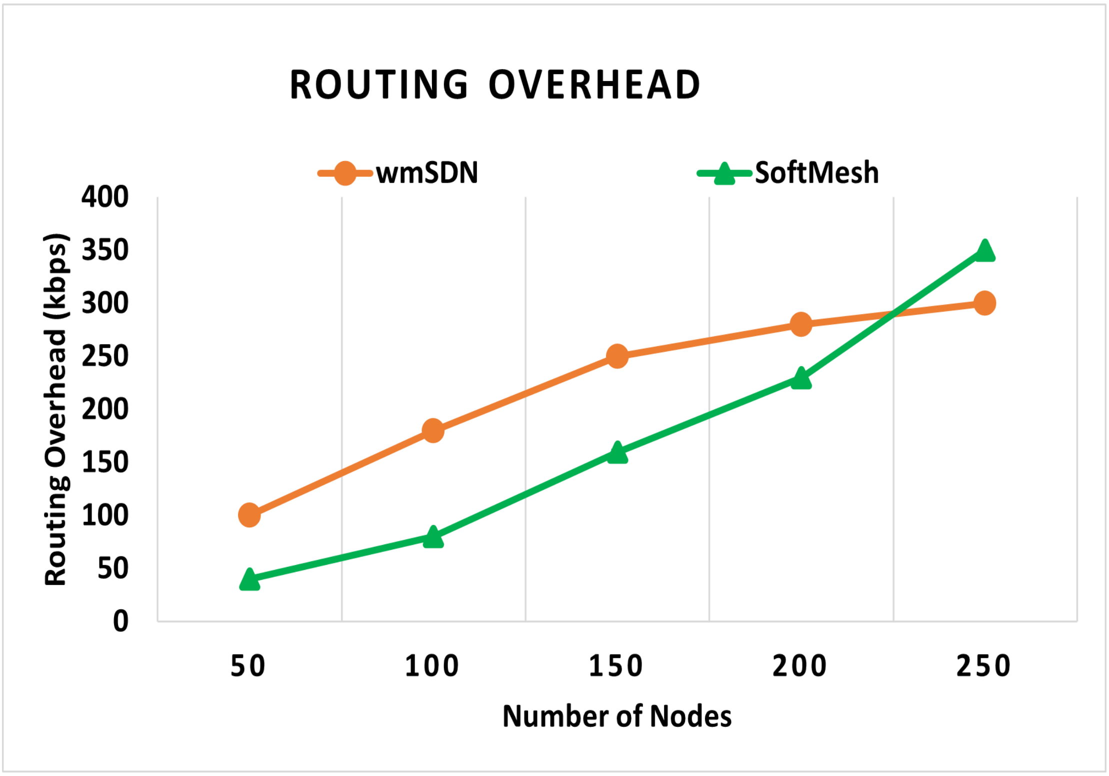


FIGURE 4.29: Routing Overhead - Scenario 02 (25% of SDN hybrid nodes and 75% of legacy nodes)

Figures 4.30, 4.31, 4.32, and 4.33 show the simulation results for scenario 03 with 50% SDN hybrid nodes and 50% legacy mesh nodes. It gives better results of average throughput behavior, packet drop ratio, end-to-end delay, and routing overhead as compared to the scenario 02 due to increase in number of SDN hybrid nodes.

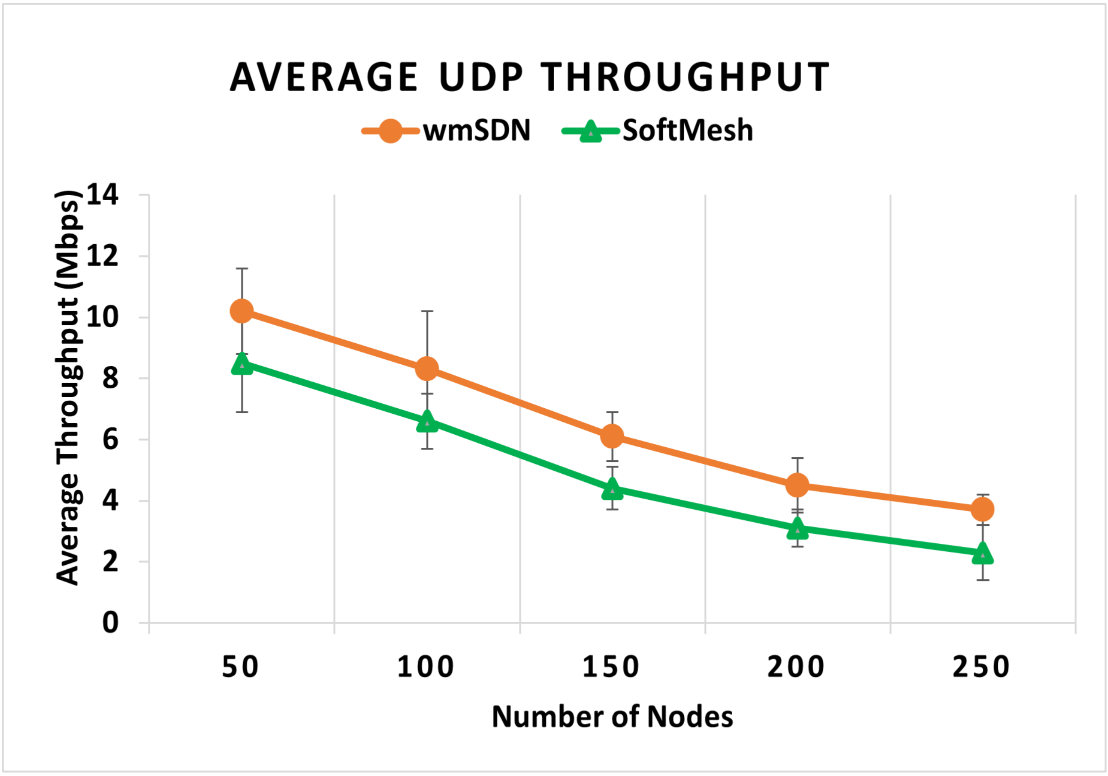


FIGURE 4.30: Average UDP Throughput - Scenario 03 (50% of SDN hybrid nodes and 50% of legacy nodes)

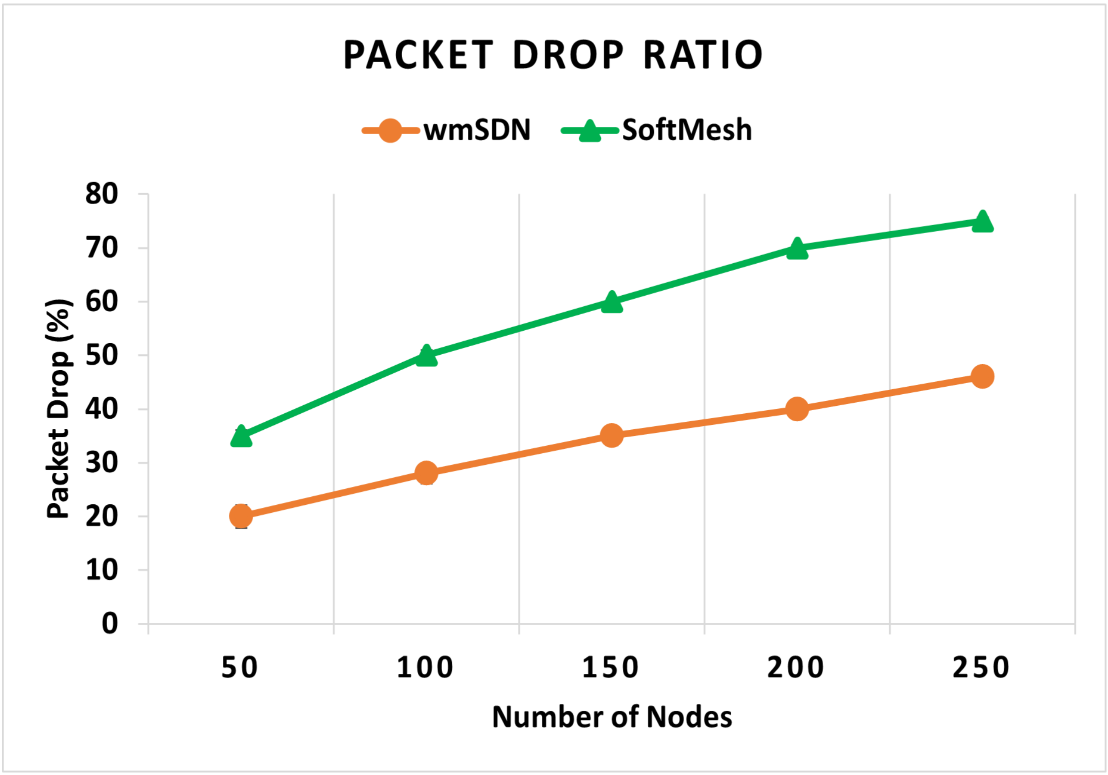


FIGURE 4.31: Packet Drop Ratio - Scenario 03 (50% of SDN hybrid nodes and 50% of legacy nodes)

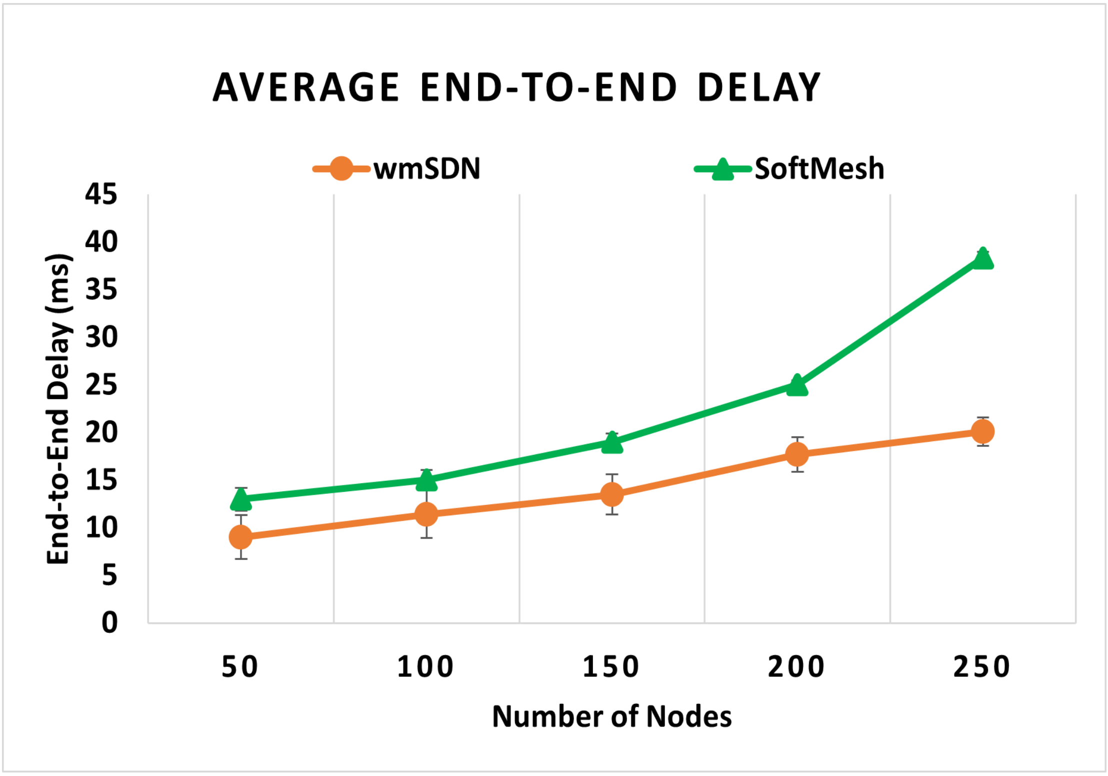


FIGURE 4.32: End-to-End Delay - Scenario 03 (50% of SDN hybrid nodes and 50% of legacy nodes)

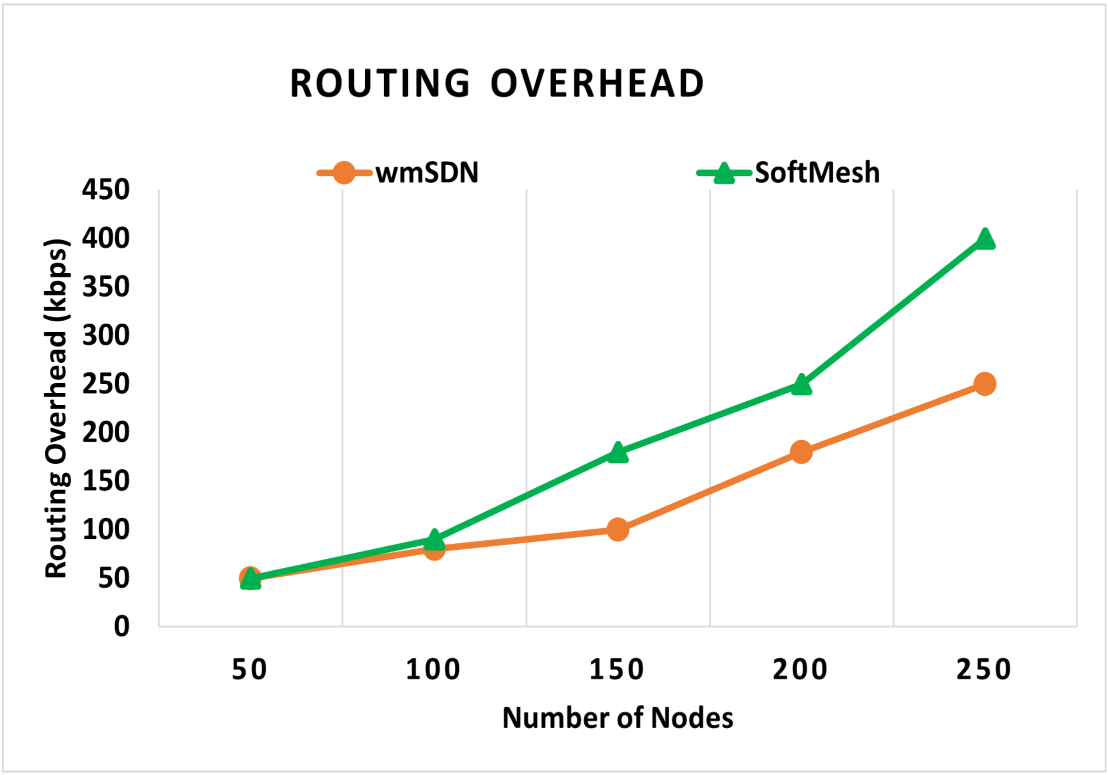


FIGURE 4.33: Routing Overhead - Scenario 03 (50% of SDN hybrid nodes and 50% of legacy nodes)

The analysis of simulation data demonstrates that the centralized approach and out-of-band signaling of proposed routing architecture 'Soft-Mesh' allow WMNs to get around the drawbacks of dispersed routing. Based on a centralized strategy, SDN routing establishes communication between nodes through the controller. The controller is asked for information regarding the way to a certain node when a node needs to connect to another node, which results in the implementation of the proper routing rules. In this regard, communication to the controller node is crucial for creating connections between nodes. Due to the static nature of network nodes in wired networks, routing rules do not alter or update frequently, hence the controller's node connectivity is unaffected. However, frequent node migration in wireless networks has a significant impact on node communication, particularly in wireless mesh networks. In comparison to conventional BATMAN and OLSR routing protocols and other hybrid (SDN/IP) routing approaches, the suggested Soft-Mesh routing architecture is more beneficial for WMN. The subsequent approaches, which seek to determine the best path to the controller, cause connections to be established between the controller and the node to take longer, allowing the controller to be unavailable for longer periods of time. Table 4.2 shows the comparison between the proposed Soft-Mesh architecture and existing traditional and hybrid (SDN/IP) routing systems.

TABLE 4.2: Comparison of Soft-Mesh with existing Traditional and Hybrid (SDN/IP) Routing Approaches

Features	**OLSR Traditional**	**BATMAN Routing**	**wmSDN Hybrid (SDN/IP)**	**Hakiri Routing**	**Proposed Architecture (Soft Mesh)**
Scalability	Easily Scalable	Easily Scalable	Possibly Scalable	Possibly Scalable	Easily Scalable
Reliability	High	High	Possibly High	Possibly High	Very High
Controller Failure Solution	No Controller exists	No Controller exists	Switch to traditional routing	No solution proposed	Switch to traditional routing

Table 4.2: Comparison of Soft-Mesh with existing Traditional and Hybrid (SDN/IP) Routing Approaches

Features	**OLSR Traditional**	**BATMAN Routing**	**wmSDN Hybrid (SDN/IP)**	**Hakiri Routing**	**Proposed Architecture (Soft Mesh)**
Resource Optimization	Not at all	Not at all	Possibly Optimized	Moderately Optimized	Highly optimized
Congestion Control Mechanism	Not at all	Not at all	No congestion control	Load balancing mechanism	Load balancing mechanism
Network Programmability	Not at all	Not at all	Programmable	Programmable	Programmable
Network Monitoring	Not at all	Not at all	Simple Network Monitoring	Simple Network Monitoring	(Few nodes need to be replaced with SDN hybrid nodes)
Cost-effectiveness	Effective (All nodes are legacy nodes)	Effective (All nodes are legacy nodes)	Highly Expensive (All nodes need to be replaced with SDN hybrid nodes)	Highly Expensive (All nodes need to be replaced with SDN hybrid nodes)	Effective

4.4 Mathematical Model for the Proposed Routing Architecture

This section presents the mathematical model of the proposed architecture which provides the basis for computing aggregated flows.

The graph of WMN is represented by
G = (V, E)
where V and E represent the set of all vertices (i.e., nodes) and edges (i.e., links), respectively.

Let TD$_v$ denote the traffic demand of the flows generated in node v, Let TD$_v$ = {f$^{ID:1}$, f$^{ID:2}$, f$^{ID:3}$, ..., f$^{ID:n}$} denote the traffic demands for n flows,
Given a traffic distribution △v, the outgoing flows' demands in different paths of node v can be obtained by using the below model:

$$\{f^{p1}A1, f^{p2}A2, ..., f^{pn}An\} = \triangle_v \times TD_v \qquad \forall_v \in V \tag{4.3}$$

where $f^{p1}A1$ implies the rate of flow A that is sent out from node v on path p. As the aggregated flows must be less than their capacity c(e) on each connection in the time slot τ, we have:

$$\Sigma_{p\in f(e)} f^p \leq c(e) \qquad \forall_e \in E \tag{4.4}$$

The network monitoring and traffic measurement module makes use of aggregated flow parameters to get the statistics of network traffic on each node.

4.5 Conclusion

The proposed routing architecture Soft-Mesh improves the routing efficiency and scalability of wireless mesh networks by systematically and gradually migrating WMNs to SDNs in an effective manner. The major goal of this research is to merge SDN with WMN using a hybrid topology, as well as to investigate routing issues and their consequences while modifying the SDN node architecture. The suggested hybrid routing architecture is taken into account by legacy nodes and SDN hybrid nodes. The proposed routing architecture combines the features two categories of hybrid wireless mesh and software defined networks that include coexistence-based and cohabition-based hybrid approaches. In order to establish smooth interoperability between SDN and legacy nodes, SDN nodes are made hybrid where coexist OLSR routing for IP-based forwarding, and the OpenFlow protocol for SDN forwarding.

Simulations are carried out on topologies with varying number of nodes such as 50, 100, 150, 200, and 250 SDN hybrid nodes and legacy nodes, respectively. Furthermore, the percentage of SDN hybrid nodes and legacy nodes in the network topology is configurable to produce three alternative simulation scenarios, with 10%, 25%, and 50% of SDN hybrid nodes and 90%, 75%, and 50% of legacy nodes in the network topology. The OLSR and BATMAN routing systems are simulated because they are regarded to be more stable than any other standard routing approaches for wireless mesh networks. Soft-Mesh architecture has been compared to wmSDN and Hakiri for hybrid approaches. In terms of different performance measures, such as average UDP throughput, end-to-end delay, packet drop ratio, and routing overhead, the suggested routing Soft-Mesh gives improved results. For the incremental fraction of SDN hybrid nodes, Soft-Mesh improves the results by 70%. As a result, our findings suggest that the SDN method will benefit the distributed routing protocol's operations. Furthermore, the proposed routing architecture also provides a routing solution in case of controller goes down or failure occurs, this particular solution makes our routing approach robust and reliable. Results show that our proposed routing architecture's average throughput behavior, packet drop ratio, end-to-end delay, and routing overhead are extremely similar to that of OLSR with a little increase due to the rules already installed by controller at the setting up of network. Such an improvement is brought about by the cohabitation of OpenFlow and OLSR in the SDN hybrid nodes architecture.

Chapter 5

AdNetMon: An Adaptive Network Monitoring Architecture for Hybrid Wireless Mesh and Software Defined Networks

5.1 Introduction

In network administration, monitoring is very challenging. The applications require accuracy with getting statistics in time for managing the resources of network while performing aggregation at different levels. Conversely, the overhead of network must be minimized for collecting the data. The majority of the tasks related to manage the networks that include enforcement of SLAs, load balancing, accounting, precise and in time statistics collection, traffic engineering, intrusion detection, etc. The use of management applications is getting significant in order to monitor the resources of network with aggregation of statistics at different levels. For instance, network applications generally used for load balancing to address the issue of congestion control require traffic statistics per second for each port. However, the rate of packet drops at interfaces used for incoming and outgoing traffic are needed by the application for enforcement of security measures.

A number of aggregation levels, reduced delay, and accuracy should be available for management applications to choose from in a well-designed network monitoring architecture. The monitoring architecture's responsibility is to choose network resources and poll for statistics collection. The monitoring architecture should prevent unnecessary monitoring overhead while gathering, processing, and disseminating monitored data at the designated aggregation level and frequency. Today's IP network monitoring solutions are ad hoc and challenging to implement, for a smooth network administration and management, an efficient monitoring, precise and accurate application is required. There are three techniques presented by [155–157] for sample-based and direct monitoring in conventional networks, however, these techniques bear a large overhead. Additionally, some network equipment suppliers have their own methods for gathering traffic statistics [155, 157]. Because there is a lack of openness and compatibility across different approaches and technologies, collecting network traffic data in traditional IP networks has become a challenging task.

According to SDN, the management applications are built by using an interface of programming that is provided by control plane rather than an interface for configuration for fine-tuning network characteristics. The enhanced programmability allows for the simplification of network administration duties and the simplification of distributed configuration [158]. A built-in interface for the communication between control-data plane pair has been created that is known as OpenFlow protocol. At the controller, OpenFlow provides primitives for collecting statistics on each flow. The controller can get information on current flows by polling switches. Instead, it may ask a switch to report flow statistics on a regular basis (upon flow timeout). The controller's viewpoint of the network is shown. The capabilities of an OpenFlow Controller can be leveraged to build sophisticated and efficient monitoring systems. However, under the present situation, the control plane is integrated with the SDN based network management application as opposed to existing application independent of such an integration.

In this chapter, we propose an SDN based adaptive network monitoring architecture that provides several benefits for developing an architecture to manage the network resources, and implemented with the available platform used by SDN controller. The proposed network monitoring architecture provides network overview and an algorithm for collection of nodes' statistics. Generally, there exists a trade-off between accuracy and resources such as network bandwidth, CPU, memory etc; used for the monitoring of network resources, thus an adaptive architecture has been proposed for monitoring of networks as well as scheduling the collection of statistics.

5.2 Network Monitoring Architecture

Figure 5.1 displays our monitoring architecture along with presenting the stack of software for installation of SDN based network. Typically, OpenFlow controllers offer a development environment for creating specialized network applications independent network technologies and complexities. The North- bound API is a programming interface that an OpenFlow controller provides to network applications. Each application has a different requirement for the level of detail needed for statistics collection. Aggregate statistics are required by some applications whereas per-flow data are desired by others. An ISP's user billing application, for instance, would anticipate getting use information for each traffic flow occurring over the customer's home network. The OpenFlow API and the present controller implementations do not support these aggregate levels (e.g., NOX, POX, and Floodlight). A controller's implementation will become significantly more complex if monitoring functionalities are added. As a result, a distinct layer is required to separate network applications and controller design from monitoring complexity.

To achieve this, we provide an adaptive and effective architecture for network monitoring that obtains network statistics by using RESTful API, that is used by the controller to communicate with application layer as a northbound API. The high-level monitoring needs for the applications are translated by the monitoring architecture. Additionally, it hides the intricacies of storage organization and data collection.

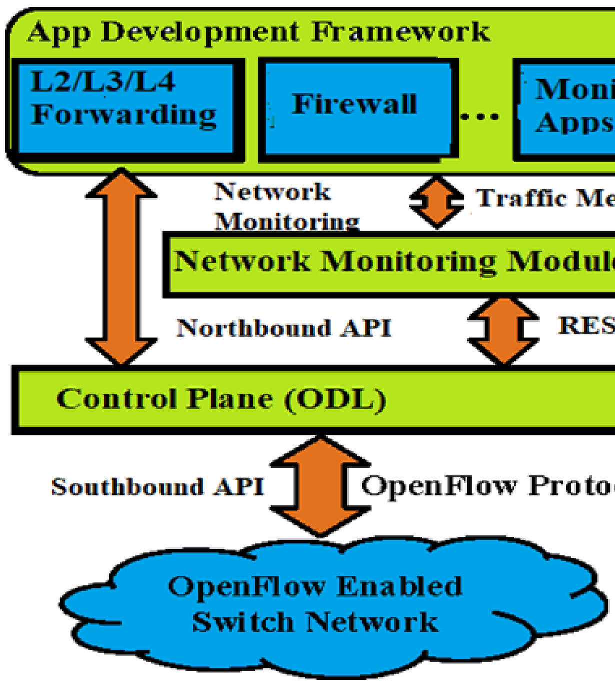


FIGURE 5.1: SDN Software Stack

Figure 5.1 illustrates software stack of SDN whereas Figure 5.2 shows the monitoring architecture with its components. These components are described in further details below:

- **Request Analyzer**: This component transforms application-level high-level primitives into flow-level primitives. This module is in charge of connecting to other modules.

- **Scheduler**: It is employed to schedule the network's switches for polling in order to compile statistics. For each flow, queue, and port, aggregate data is available on switches with OpenFlow support. A scheduling mechanism is used to determine the polling time.

- **Node Selector**: When a statistics collection event is scheduled, one or more switches must be identified and selected for statistics collecting. This component chooses which switches must be polled in order to collect the right information at the appointed time. Simple formulas can be used only the ingress switch needs to be queried in order to learn about a flow.

- **Aggregator and Data Store**: These are used to gather and store raw data from switches that have been preset. At the necessary levels of aggregation, network monitoring statistics are calculated using the acquired data. Data store is a representation of the storage system. Examples include relational databases, standard files, and key-value storage.

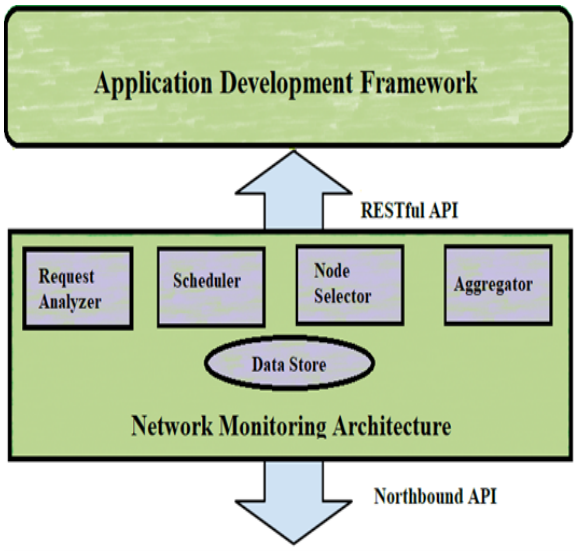


FIGURE 5.2: Network Monitoring Architecture

5.3 An Adaptive Network Monitoring Algorithm

An adaptive monitoring strategy to maintain tabs on network resources is presented in this section. The production of accurate and timely information while consuming the least amount of network capacity is our aim. The communication from switch to controller is supposed to be carried out via the OpenFlow protocol. OpenFlow uses the fields gathered from the headers of the packets for identification of flows at different layers that layer 2 – layer 4. In case, such a flow received by a switch not matching with any of the existing rules available in forwarding table, a *PACKET_IN* message is sent to the controller. The relevant rules for forwarding are installed by the controller in the respective node using *FLOW_MOD* message. An idle timeout is specified by the controller for forwarding of rules. After this point, a forwarding rule (and any associated flows) are deactivated from the node.

In case of ejection of a flow, the node used to transmit a *FLOW_REMOVED* message towards the controller. It contains two pieces of information: 1) the flow's duration and 2) the number of matching bytes.

For connection consumption monitoring, the previous methodologies advise only tracking the *PACKET_IN* and *FLOW_REMOVED* messages. This method does have a noticeable average lag in retrieving subsequent statistics, though. The ability to monitor traffic peaks is also problematic. The controller can, however, use these messages to send the switch a *FLOW_STATISTICS_REQUEST* message in order to get information about a particular flow. The switch transmits to the controller two type of information in a *FLOW_STATISTICS_REPLY* message, that is byte count (total number of bytes) and time duration it takes for that flow. Flow statistics can be easily gathered by polling the switches with the *FLOW_STATISTICS_REQUEST* message at regular intervals. Extremely accurate data are produced by polling often (i.e., with a small polling interval). However, there will be significant network monitoring overhead as a result. In order to balance network overhead and statistics collection accuracy, we suggest a variable frequency flow statistics collecting approach.

A new entry is added into the active flow table coupled with a τ milliseconds as initial statistics collection timeout, on receiving a *PACKET_IN* message by the controller. In addition, it will further receive the flow's statistics in a *FLOW_REMOVED* message in case, within τ milliseconds the flow gets expired. Otherwise, a *FLOW_STATISTICS_R* message will be sent to the relevant switch by the controller to collect statistics on that flow. 'α, a small constant multiplies with the timeout, if there is no significant change in the collected statistics results in no change in the number of bytes received during this time period and bytes do not exceed a threshold '$\triangle$'. This operation is iteratively used until timeout value reaches its maximum limit τ_{max}. In contrast, if significant change in the collected statistics results in a big change in the number of bytes received and bytes exceed a threshold '$\triangle$'. This operation is iteratively used until timeout value reaches its minimum limit τ_{min}, another constant β is used to divide the flow's scheduling timeout. For flows that greatly increase link utilization at that time, polling frequency is kept higher, however polling frequency is kept lower for flows reducing link utilization. If their contribution increases, the proposed methodology will alter the scheduling timeout, changing the polling frequency to reflect the increase in traffic. By batching *FLOW_STATISTICS_REQUEST* messages for flows with the same timeout, we further enhance this method. By doing this, network monitoring traffic will be reduced and variable-frequency polling effectiveness will be preserved. Algorithm 5.1 shows the pseudocode for the description of proposed flow statistics collection method whereas simulation parameters of Adaptive Network Monitoring Algorithm are illustrated by Table 5.1.

Table 5.1: Simulation Parameters of Adaptive Network Monitoring Algorithm

Symbol	**Unit**	**Description**
τ	Millisecond	Initial statistics collection timeout
$\triangle$	Byte	Byte Count Threshold for τ_{min} & τ_{max}
α	Constant	Constant Value for τ_{max}
β	Constant	Constant Value for τ_{min}
τ_{max}	Millisecond	Maximum timeout value
τ_{min}	Millisecond	Minimum timeout value

Algorithm 5.1: Flow Statistics Collection

```
Globals: $active\_flows$ //Currently Active Flows
         $schedule\_table$ //Associative table of active flows // indexed by poll frequency
         $U$ // Utilization Statistics. Output of this algorithm
if $e$ is Initialization event then
  $active\_flows \leftarrow \phi$, $schedule\_table \leftarrow \phi$, $U \leftarrow \phi$
end if
if $e$ is a PacketIn event then
  $f \leftarrow (e.switch, e.port, \tau_{min}, 0)$
  $schedule\_table[\tau_{min}] \leftarrow schedule\_table[\tau_{min}] \cup f$
else if $e$ is timeout $\tau$ in schedule_table then
    for all flows $f \in schedule\_table[\tau]$ do
        send a FlowStatisticsRequest to $f.switch$
    end for
else if $e$ is a FlowStatisticsReply event for flow $f$
then
     $diff\_byte\_count \leftarrow e.byte\_count - f.byte\_count$
     $diff\_duration \leftarrow e.duration - f.duration$
     $checkpoint \leftarrow current\_time\_stamp$
     $U[f.port][f.switch][checkpoint] \leftarrow (diff\_byte\_count, diff\_duration)$
     if $diff\_byte\_count < \Delta$ then
         $f.\tau \leftarrow \min(f.\tau\alpha, \tau_{max})$
         Move $f$ to $schedule\_table[f.\tau]$
    else if $diff\_byte\_count > \Delta$ then
         $f.\tau \leftarrow \max(f.\tau/\beta, \tau_{min})$
         Move $f$ to $schedule\_table[f.\tau]$
     end if
end if
```

5.3.1 Link Utilization Monitoring

Monitoring of link utilization has been performed by using the OpenDaylight (ODL) controller platform as a tangible application of our suggested architecture and monitoring methodology.

It's worth noting that the architecture we've proposed is designed to run simulations and demonstrate the efficacy of our approach. As a result, without sacrificing generality, we established the following simplification assumption concerning flow identification and matching. The simulations were carried out in UDP mode using Iperf [159]. There was also some DHCP activity on the underlying network, which utilizes UDP as well. It's worth mentioning that our suggested monitoring architecture's components aren't completely in place yet. As a result, we turned to an ODL module to develop the link utilization monitoring application.

The *PACKET_IN* and *FLOW_REMOVED* messages are investigated for the purpose of monitoring the additions and deletions of flows from switches. The scheduling timeout value served as an index in a hash table that was also preserved. There is a list of active flows in each bucket with a timeout of milliseconds that needs to be polled every millisecond. Every millisecond, a worker thread for each bucket in the hash table wakes up and sends a *FLOW_STATISTICS_REQUEST* message to the switches responsible for the flows in that bucket. The monitoring module receives the *FLOW_STATISTICS_REPLY* messages asynchronously. This establishes a benchmark for measuring each reply message. Divide the difference in bytes from the previous checkpoint by the difference in time from the previous checkpoint to determine a flow's contribution. The monitoring module evaluates the measurement checkpoints on the pertinent link and, if necessary, adjusts the consumption of earlier checkpoints. Depending on the change in the number of bytes since the previous measurement checkpoint, the active flow entries are transferred between the buckets of the hash table with lower or higher timeout values. As of right now, we have a fundamental REST API that provides a way to obtain link statistics for any link in the network. On the other hand, in the long run, we would like to build a REST API that would enable third-party apps to register a specific flow for analytics and monitoring. Although some flow identification and matching assumptions are made throughout the simulation, this does not change how universal our suggested approach is. Our long-term goal is to have a fully operational monitoring system that makes it possible to gather flow data efficiently. The statistics made available by our approach will make it much easier to construct network monitoring applications. The scheduler component of this design is at the core of the proposed scheduling approach, and the only presumptions made here are for the development of a link utilization monitoring application that uses our architecture. The Openflow messages used in the suggested network monitoring architecture are described in detail in Table5.2.

Table 5.2: Openflow Messages for Adaptive Network Monitoring Algorithm

OpenFlow Message Type	**Message Description**
PACKET_IN	A packet-in message is sent to the controller for all packets that do not have a matching flow entry or if a packet matches an entry with a send to controller action. It gives the packet's control over to the controller.
FLOW_MOD	It alters the flow entry message. This message is sent by the controller to change the flow table.
FLOW_REMOVED	This message alerts the controller that a flow entry has been removed from a flow table. When the flow expires, the flow modify message additionally indicates whether the switch should send a flow removal message to the controller.
FLOW_STATISTICS_REQUEST	It's a request for individual flow statistics. This message is used by the controller to query individual flow statistics.
FLOW_STATISTICS_REPLY	It's a response message for an individual flow statistic. In response to a request for individual flow statistics, the switch sends this message.

5.3.2 Evaluation

The performance of the proposed network monitoring architecture is presented in this section. The simulation is run for a baseline situation in which the controller polls the switches at a fixed interval to collect link utilization data. Mininet-Wifi simulator was used to simulate a network that included a controller, SDN hybrid nodes, and traditional routers. Section 5.3.2.1 describes the simulation model, while Section 5.3.2.2 covers the assessment metrics. Finally, in Section 5.3.2.3, the results are provided in comparison to the existing network monitoring architectures periodic polling and FlowSense [141].

5.3.2.1 Simulation Model

A full mesh topology is considered to be investigated for this work, as shown in Figure 5.3. Iperf was used to generate UDP flows between hosts for a total of 100s. Generally, a number of network applications use UDP protocol for communication instead of TCP protocol due to its inherent advantages such as rapid delivery, efficiency, less overhead etc. Therefore, we investigated UDP flows during our study. The idle timeout for the active flows of switches is taken 5 seconds. We have also intentionally established pauses between traffic flows that vary in time in order to assess various scenarios. As seen in the figure, between the 28th and 30th second, as well as between the 33rd and 35th second, a pause less than the soft timeout was included to see how the proposed scheduling algorithm and the FlowSense [141] respond to abrupt traffic spikes. For our scheduling method, we set the minimum and maximum polling intervals to 500ms and 5s respectively. A one-second polling period was used by the constant. The byte count threshold for τ_{min} and τ_{max} parameter $\triangle$ were set to 100MB. Finally, after considering the concept of additive-increase multiplicative-decrease (AIMD) algorithm used for TCP congestion control we have set α and β to 2 and 6, respectively, as indicated in Section 4.3. To quickly react and adapt to any changes in traffic, β was set to a higher value than the value of α as it plays an active role in reaching the minimum timeout value τ_{min}. Thus, higher value of β will result in minimum value of timeout.

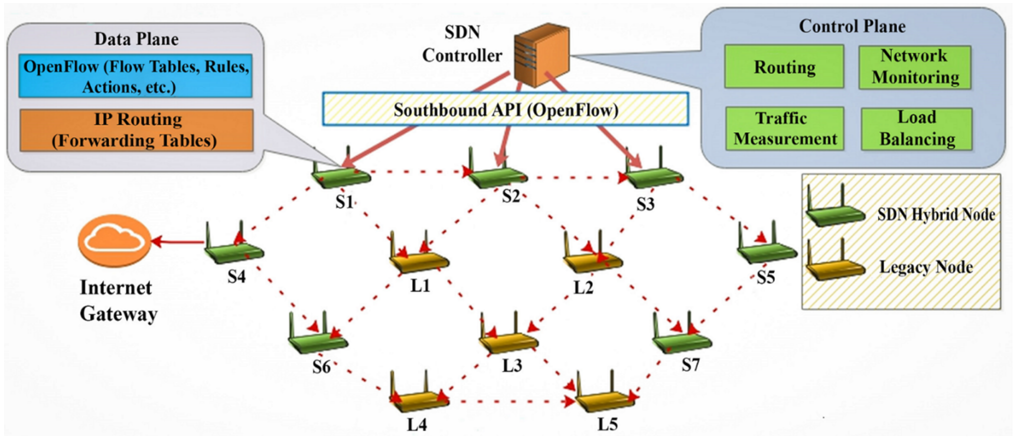


FIGURE 5.3: Topology for Network Monitoring Architecture

5.3.2.2 Evaluation Metrics

1. **Link Utilization**: It is referred to as the link's immediate throughput and is expressed in megabits per second (Mbps). It is reported that the link between switches S1 and S4 is used (Figure 5.3), exhibits a large variety of usage variations, is a component of all flows, and is the most often utilized one In order to examine how the accuracy-to-monitoring overhead trade-off

is impacted, we additionally test various minimum polling interval (min) options.

2. **Overhead**: The controller's total number of *FLOW_STATISTICS_REQUEST* messages is utilised to determine overhead.

5.3.2.3 Results

1. **Link Utilization**: Figure 5.4 shows three independent techniques' measurements of the S1 and S4 link's consumption over the course of the simulation. Periodic polling is the most like the traffic in the default setup. Because of its extremely fine-grained measurement, FlowSense [141] is unable to detect spikes in traffic. Since less traffic pauses than the soft timeout value, FlowSense [141] displays a lower usage than is actually the case. On the other hand, the use pattern discovered through routine polling is remarkably similar to our suggested technique. Despite not completely recording the initial traffic surge, it quickly adjusted and was able to successfully capture the next one.

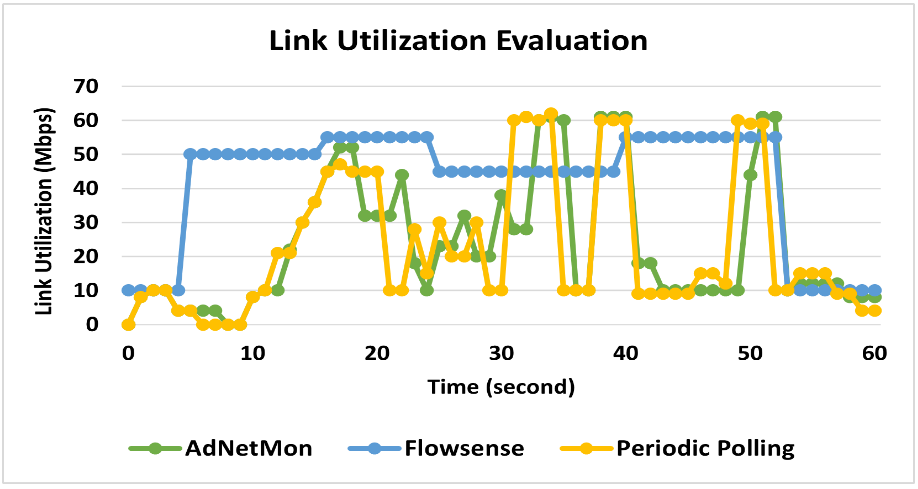


FIGURE 5.4: Link Utilization Measurement

2. **Overhead**: Figure 5.5 displays the communications overhead of the first scenario and the method we recommend. FlowSense[141] has no messaging overhead and is therefore not represented in the diagram because it does not transmit *FLOW_STATISTICS_REQUEST* messages. The fixed polling technique polls every active flow when the specified timeout expires causing injection of a number of control messages into the network. However, the

adaptive feature of our proposed architecture reduces the injection of unnecessary number of messages in the network. In some cases, our method delivers nearly 50% fewer signals than the periodic polling method even though it has more query points across the timeline and collects information about flows with fewer network messages each time. Although FlowSense[141] has no measurement overhead, it is far less precise than our adaptive scheduling method. Averaging at 5:5 messages per second as opposed to periodic polling's 10:5 messages per second, the suggested network monitoring architecture also creates a very little amount of monitoring traffic. In conclusion, the proposed technique for scheduling flow statistics can achieve accuracy comparable to the constant periodic polling approach while utilising less messaging overhead.

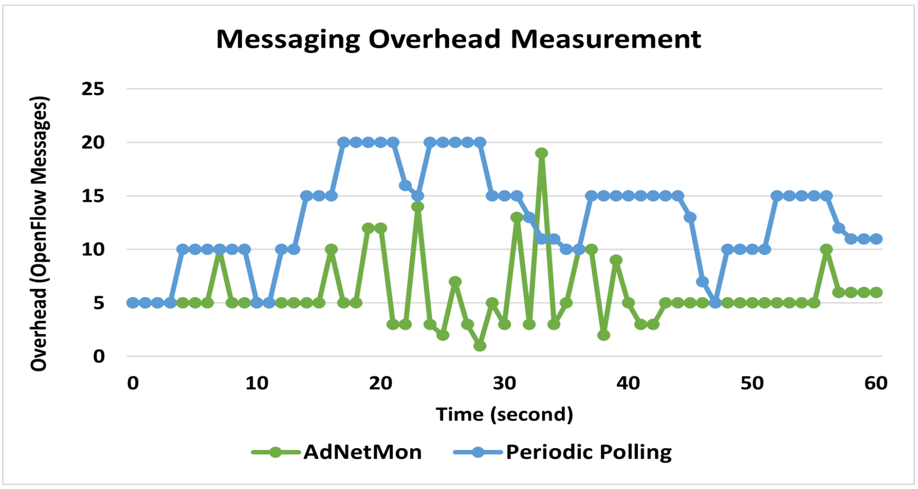


FIGURE 5.5: Messaging Overhead Measurement

3. **Minimum Polling Frequency Effect,** τ_{min}: Our scheduling system adapts to the traffic pattern, as shown in Flow Statistics Collection Algorithm. The polling frequency swiftly falls and reaches τ_{min} in response to a rapidly changing traffic spike. The effect of τ_{min} on monitoring accuracy is seen in Figure 5.6. For τ_{min} = 250ms, the accuracy of monitoring data is high, but a steadily decrease can be observed with increasing the values of τ_{min} values. But as in Figure 5.7, monitoring precision can be achieved at the cost of network overhead. The different values ofτ_{min} are used to calculate RMS error with monitoring accuracy, that varies depending on application

requirement. In addition to that, we compared CPU utilization of the proposed architecture with FlowSense [141] and Periodic Polling [127], initially the proposed network monitoring architecture gives 30% CPU utilization, then remains stable at 20% CPU utilization. Moreover, FlowSense[141] and Periodic Polling [127] give 50% and 90% CPU utilization respectively at initial and then get stable at 30% and 60% CPU utilization respectively. The reason for higher CPU utilization in the beginning is that initially it takes time to setup the flow statistics collection environment and resources allocation. The analysis shows that the proposed network monitoring architecture is adaptive and polling occurs on the basis of requirement, thus gives better results as compared with FlowSense[141] and Periodic polling [127], and outperforms in terms of CPU utilization.

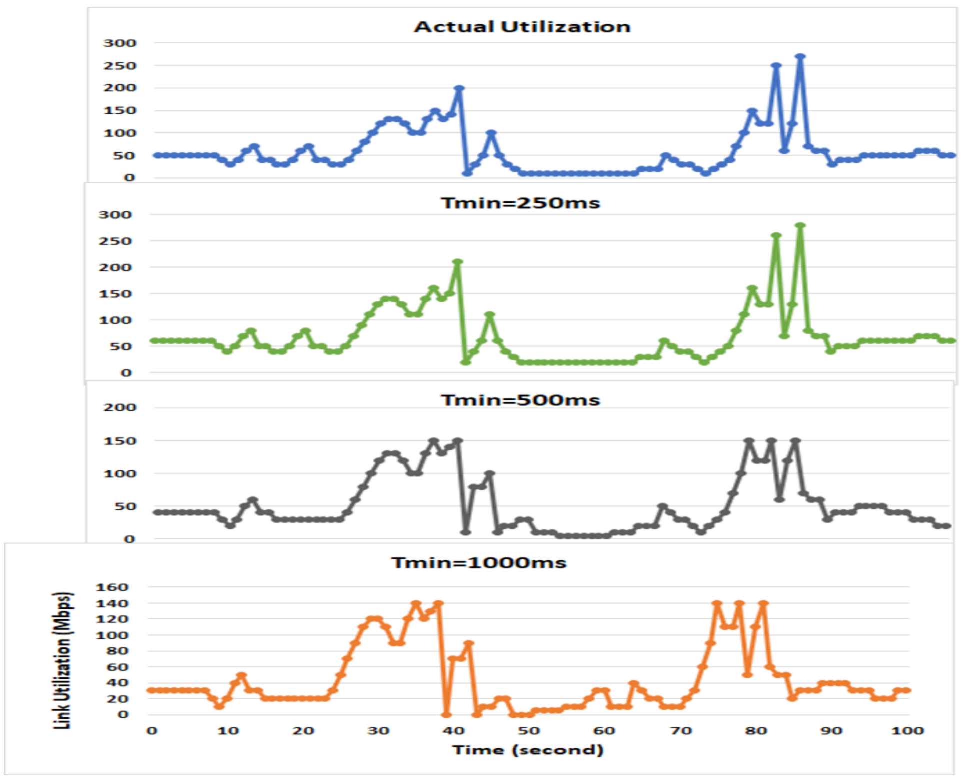


FIGURE 5.6: Minimum Polling Frequency Effect Measurement for Link Utilization

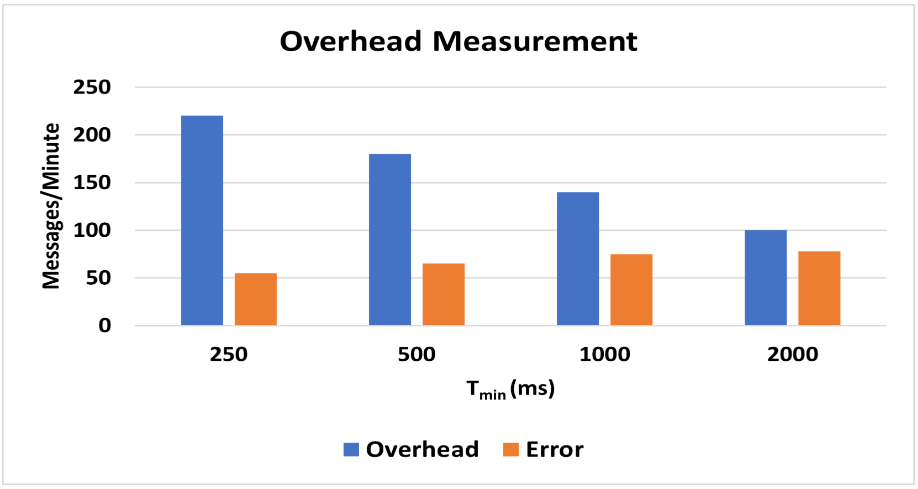


FIGURE 5.7: Minimum Polling Frequency Effect Measurement for Messaging Overhead

5.4 Conclusion

The proposed network monitoring architecture is a flexible, adaptable, and expandable architecture for hybrid SDN network, that is introduced in this chapter. The generic RESTful API can be used to specify almost every aspect of monitoring. Furthermore, new techniques can replace the basic components in the proposed architecture without affecting the other components. We introduced an adaptive scheduling approach for flow statistics gathering to demonstrate the effectiveness of the proposed architecture. Using this approach, we created a specific use case of monitoring connection consumption. We assessed and compared it to the performance of FlowSense [141] and a Periodic polling [127] method. We found that the suggested method can collect statistics more precisely than FlowSense [141]. The analysis shows that the proposed network monitoring architecture gives better results as compared with FlowSense[141] and Periodic polling [127], and outperforms in terms of CPU utilization.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Soft-Mesh, a robust routing architecture for hybrid SDN and wireless mesh networks is presented in this study. The major goal of this research is to merge SDN with WMN using a hybrid topology, as well as to investigate routing issues and their consequences while modifying the SDN node architecture. The suggested hybrid routing architecture is taken into account by legacy nodes and SDN hybrid nodes. The proposed routing architecture combines the features two categories of hybrid wireless mesh and software defined networks that include coexistence-based and cohabition-based hybrid approaches. In order to establish smooth interoperability between SDN and legacy nodes, SDN nodes are made hybrid where coexist OLSR routing for IP-based forwarding, and the OpenFlow protocol for SDN forwarding. It should be emphasized that our research does not support the use of SDN routing to replace traditional routing methodologies. The purpose of this research is to understand that an SDN-based architecture may help with WMN routing to improve its performance rather than arbitrarily equating legacy routing techniques using in-band signaling, and with the centralized approach using out-of-band signaling. It is explored how legacy protocols can be used with SDN networking architecture and to what extent the former can help the latter. In addition, when compared to other hybrid routing architectures, Soft-Mesh architecture delivers a cost-effective solution with smooth interoperability between conventional and SDN nodes.

Simulations are carried out on topologies with varying number of nodes such as 50, 100, 150, 200, and 250 SDN hybrid nodes and legacy nodes, respectively. Furthermore, the percentage of SDN hybrid nodes and legacy nodes in the network topology is configurable to produce three alternative simulation scenarios, with 10%, 25%, and 50% of SDN hybrid nodes and 90%, 75%, and 50% of legacy nodes in the network topology. The scale of the topology has also been kept configurable, with 500m*500m topologies for 50 and 100 nodes and 1000m*1000m topologies for 150, 200, and 250 nodes. The OLSR and BATMAN routing systems are simulated because they are regarded to be more stable than any other standard routing approaches for wireless mesh networks. Soft-Mesh architecture has been compared to wmSDN and Hakiri for hybrid approaches. In terms of different performance measures, such as average UDP throughput, end-to-end delay, packet drop ratio, and routing overhead, the suggested routing Soft-Mesh gives improved results. For the incremental fraction of SDN hybrid nodes, Soft-Mesh improves the results by 70%. As a result, our findings suggest that the SDN method will benefit the distributed routing protocol's operations. Furthermore, the proposed routing architecture also provides a routing solution in case of controller goes down or failure occurs, this particular solution makes our routing approach robust and reliable. Results show that our proposed routing architecture's average throughput behavior, packet drop ratio, end-to-end delay, and routing overhead are extremely similar to that of OLSR with a little increase due to the rules already installed by controller at the setting up of network. Such an improvement is brought about by the cohabitation of OpenFlow and OLSR in the SDN hybrid nodes architecture.

The proposed network monitoring architecture 'AdNetMon', is a flexible, adapt able, and expandable architecture for hybrid SDN network, that is introduced in this chapter. The generic RESTful API can be used to specify almost every aspect of monitoring. Furthermore, new techniques can replace the basic components in the proposed architecture without affecting the other components. We introduced an adaptive scheduling approach for flow statistics gathering to demonstrate the effectiveness of the proposed architecture. Using this approach, we created a specific use case of monitoring connection consumption. We assessed and compared it to the performance of Flowsense and a periodic polling method. We discovered that the proposed approach can collect statistics with more accuracy than Flowsense. Despite this, the incurred communications cost is up to 50% higher than in a comparable periodic poling technique.

6.2 Future Work

We propose to use SDN-based WMNs solutions to address the major difficulties in the IoT area, emphasizing their benefits while exposing their flaws. There are numerous fresh areas for further study. Moreover, our proposed architecture can be evaluated using distributed controllers' applications. Resource management for data centers and wireless network resources has been the main focus of recent developments. Second, current wireless devices require a variety of modulation methods to conform to a specific radio interface, which limits their ability to adapt to the increasing demands on bandwidth and frequency spectrum resources. SDN, on the other hand, makes the data plane programmable. The coexistence of SDN and Software Defined Radio (SDR), which may necessitate a cross-layer design, can, in our opinion, combine network resource management and radio resource management. Future projects will also incorporate a time-sensitive networking element. Our long-term goal for this project is to develop a hybrid SDN monitoring architecture that is community-driven and open source. This should provide a full abstraction layer for the SDN control platform on top of which network monitoring applications can be quickly developed. We want to build an independent QoS policy enforcement application on top of the suggested network monitoring architecture, then carry out comprehensive OpenFlow tests to assess the performance of our solution. Our long-term objective is to enable interoperability between the proposed approach and platforms that employ distributed controllers.

Moreover, our future objective is to create an open source and community driven hybrid SDN monitoring architecture. This ought to give the SDN control platform a complete abstraction layer on top of which applications for network monitoring can be easily built. On top of the suggested network monitoring architecture, we want to construct an autonomous QoS policy enforcement application, and then conduct extensive experiments to evaluate the performance of our design using OpenFlow.

Bibliography

[1] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," *Computer Networks*, vol. 72, pp. 74–98, 2014.

[2] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2014.

[3] F. de OliveiraSilva, J. H. de Souza Pereira, P. F. Rosa, and S. T. Kofuji, "Enabling future internet architecture research and experimentation by using software defined networking," in *2012 European Workshop on Software Defined Networking*, pp. 73–78, IEEE, 2012.

[4] A. Drescher, "A survey of software-defined wireless networks," *Dept. Comput. Sci. Eng., Washington Univ. St. Louis, St. Louis, MO, USA, Tech. Rep*, pp. 1–15, 2014.

[5] J. F. G. Orrego and J. P. U. Duque, "Throughput and delay evaluation framework integrating sdn and ieee 802.11 s wmn," in *2017 IEEE 9th Latin-American Conference on Communications (LATINCOM)*, pp. 1–6, IEEE, 2017.

[6] C. Chaudet and Y. Haddad, "Wireless software defined networks: Challenges and opportunities," in *2013 IEEE International Conference on Microwaves, Communications, Antennas and Electronic Systems (COMCAS 2013)*, pp. 1–5, IEEE, 2013.

[7] S. S. A. Gilani, A. Qayyum, R. N. B. Rais, and M. Bano, "Sdnmesh: An sdn based routing architecture for wireless mesh networks," *IEEE Access*, vol. 8, pp. 136769–136781, 2020.

[8] T. Bakhshi, "State of the art and recent research advances in software defined networking," *Wireless Communications and Mobile Computing*, vol. 2017, 2017.

[9] A. Maleki, M. Hossain, J.-P. Georges, E. Rondeau, and T. Divoux, "An sdn perspective to mitigate the energy consumption of core networks–géant2," in *International SEEDS conference*, pp. 233–244, 2017.

[10] L. Cominardi, C. J. Bernardos, P. Serrano, A. Banchs, and A. de la Oliva, "Experimental evaluation of sdn-based service provisioning in mobile networks," *Computer Standards & Interfaces*, vol. 58, pp. 158–166, 2018.

[11] H. Huang, P. Li, S. Guo, and W. Zhuang, "Software-defined wireless mesh networks: architecture and traffic orchestration," *IEEE network*, vol. 29, no. 4, pp. 24–30, 2015.

[12] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *Proceedings of the 10th annual international conference on Mobile computing and networking*, pp. 114–128, 2004.

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

[14] M. L. Sichitiu, "Wireless mesh networks: opportunities and challenges," in *Proceedings of World Wireless Congress*, vol. 2, p. 21, Citeseer, 2005.

[15] M. Eslami, O. Karimi, and T. Khodadadi, "A survey on wireless mesh networks: Architecture, specifications and challenges," in *2014 IEEE 5th Control and System Graduate Research Colloquium*, pp. 219–222, IEEE, 2014.

[16] S. Y. Shahdad, A. Sabahath, and R. Parveez, "Architecture, issues and challenges of wireless mesh network," in *2016 International Conference on Communication and Signal Processing (ICCSP)*, pp. 0557–0560, IEEE, 2016.

[17] A. O. Ajayi, A. A. Adigun, and W. O. Ismaila, “A review of routing protocols for practical rural wireless mesh networks (wmns),” *International Journal of Computer Applications*, vol. 114, no. 16, 2015.

[18] N. Feamster, J. Rexford, and E. Zegura, “The road to sdn: an intellectual history of programmable networks,” *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, 2014.

[19] R. Alvizu, G. Maier, N. Kukreja, A. Pattavina, R. Morro, A. Capello, and C. Cavazzoni, “Comprehensive survey on t-sdn: Software-defined networking for transport networks,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2232–2283, 2017.

[20] M. Mousa, A. M. Bahaa-Eldin, and M. Sobh, “Software defined networking concepts and challenges,” in *2016 11th International Conference on Computer Engineering & Systems (ICCES)*, pp. 79–90, IEEE, 2016.

[21] S. Schaller and D. Hood, “Software defined networking architecture standardization,” *Computer standards & interfaces*, vol. 54, pp. 197–202, 2017.

[22] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “Openflow: enabling innovation in campus networks,” *ACM SIGCOMM computer communication review*, vol. 38, no. 2, pp. 69–74, 2008.

[23] Y.-D. Lin, Y.-K. Lai, C.-Y. Wang, and Y.-C. Lai, “Ofbench: Performance test suite on openflow switches,” *IEEE Systems Journal*, vol. 12, no. 3, pp. 2949–2959, 2017.

[24] Y.-D. Lin, Y.-C. Lai, H.-Y. Teng, C.-C. Liao, and Y.-C. Kao, “Scalable multicasting with multiple shared trees in software defined networking,” *Journal of Network and Computer Applications*, vol. 78, pp. 125–133, 2017.

[25] A. Rostami, T. Jungel, A. Koepsel, H. Woesner, and A. Wolisz, “Oran: Openflow routers for academic networks,” in *2012 IEEE 13Th international conference on high performance switching and routing*, pp. 216–222, IEEE, 2012.

[26] Z. Kerravala, “Configuration management delivers business resiliency,” *The Yankee Group*, 2002.

[27] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. Gude, N. McKeown, and S. Shenker, “Rethinking enterprise network control,” *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1270–1283, 2009.

[28] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, “Simple-fying middlebox policy enforcement using sdn,” in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pp. 27–38, 2013.

[29] A. Arcuri, “Restful api automated test case generation with evomaster,” *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 28, no. 1, pp. 1–37, 2019.

[30] M. Paliwal, D. Shrimankar, and O. Tembhurne, “Controllers in sdn: A review report,” *IEEE access*, vol. 6, pp. 36256–36270, 2018.

[31] D. S. Rana, S. A. Dhondiyal, and S. K. Chamoli, “Software defined networking (sdn) challenges, issues and solution,” *International journal of computer sciences and engineering*, vol. 7, no. 1, pp. 884–889, 2019.

[32] P. Pinyoanuntapong, *Software defined wireless mesh networks: from theory to practice.* PhD thesis, Wichita State University, 2017.

[33] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, “Research challenges for traffic engineering in software defined networks,” *IEEE Network*, vol. 30, no. 3, pp. 52–58, 2016.

[34] P. Lin, J. Hart, U. Krishnaswamy, T. Murakami, M. Kobayashi, A. Al-Shabibi, K.-C. Wang, and J. Bi, “Seamless interworking of sdn and ip,” in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pp. 475–476, 2013.

[35] V. Nascimento, M. Moraes, R. Gomes, B. Pinheiro, A. Abelém, V. C. Borges, K. V. Cardoso, and E. Cerqueira, “Filling the gap between software defined networking and wireless mesh networks,” in *10th International Conference*

on Network and Service Management (CNSM) and Workshop, pp. 451–454, IEEE, 2014.

[36] Y. Guo, Z. Wang, X. Yin, X. Shi, and J. Wu, "Traffic engineering in sdn/ospf hybrid network," in *2014 IEEE 22nd International Conference on Network Protocols*, pp. 563–568, IEEE, 2014.

[37] C. E. Rothenberg, A. Vidal, M. Salvador, *et al.*, "Hybrid networking towards a software defined era," in *Network Innovation through OpenFlow and SDN: Principles and Design*, pp. 153–198, CRC Press, 2014.

[38] M. Bano, S. S. A. Gilani, and A. Qayyum, "A comparative analysis of hybrid routing schemes for sdn based wireless mesh networks," in *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 1189–1194, IEEE, 2018.

[39] M. Bano, R. N. B. Rais, S. S. A. Gilani, and A. Qayyum, "Sdnhybridmesh: A hybrid routing architecture for sdn based wireless mesh networks," in *Workshops of the International Conference on Advanced Information Networking and Applications*, pp. 366–375, Springer, 2020.

[40] Y. Chai and X.-J. Zeng, "The development of green wireless mesh network: a survey," *Journal of Smart Environments and Green Computing*, vol. 1, no. 1, pp. 47–59, 2021.

[41] R. Amin, M. Reisslein, and N. Shah, "Hybrid sdn networks: A survey of existing approaches," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3259–3306, 2018.

[42] M. Markovitch and S. Schmid, "Shear: A highly available and flexible network architecture marrying distributed and logically centralized control planes," in *2015 IEEE 23rd International Conference on Network Protocols (ICNP)*, pp. 78–89, IEEE, 2015.

[43] Y. Sinha, K. Haribabu, *et al.*, "A survey: Hybrid sdn," *Journal of Network and Computer Applications*, vol. 100, pp. 35–55, 2017.

[44] J. Galán-Jiménez, "Exploiting the control power of sdn during the transition from ip to sdn networks," *International Journal of Communication Systems*, vol. 31, no. 5, p. e3504, 2018.

[45] H. Park, B. Cho, I.-s. Hwang, and J. R. Lee, "Study on the sdn-ip–based solution of well-known bottleneck problems in private sector of national r&e network for big data transfer," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 1, p. e4365, 2018.

[46] M. Tanha, D. Sajjadi, R. Ruby, and J. Pan, "Traffic engineering enhancement by progressive migration to sdn," *IEEE Communications Letters*, vol. 22, no. 3, pp. 438–441, 2018.

[47] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, *et al.*, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[48] C.-Y. Hong, S. Mandal, M. Al-Fares, M. Zhu, R. Alimi, C. Bhagat, S. Jain, J. Kaimal, S. Liang, K. Mendelev, *et al.*, "B4 and after: managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined wan," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pp. 74–87, 2018.

[49] D. E. Henni, A. Ghomari, and Y. Hadjadj-Aoul, "A consistent qos routing strategy for video streaming services in sdn networks," *International Journal of Communication Systems*, vol. 33, no. 10, p. e4177, 2020.

[50] S.-H. Tseng, A. Tang, G. L. Choudhury, and S. Tse, "Routing stability in hybrid software-defined networks," *IEEE/ACM Transactions On Networking*, vol. 27, no. 2, pp. 790–804, 2019.

[51] H. Wang, Y. Yu, and Q. Yuan, "Application of dijkstra algorithm in robot path-planning," in *2011 second international conference on mechanic automation and control engineering*, pp. 1067–1069, IEEE, 2011.

[52] S. Aditi, K. Mahadev, S. Prasad, S. Eswaran, and P. Honnavalli, "Opendaylight as software defined networking controller: Shortcomings and possible solutions," in *2022 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pp. 1–6, IEEE, 2022.

[53] R. Durner, A. Blenk, and W. Kellerer, "Performance study of dynamic qos management for openflow-enabled sdn switches," in *2015 IEEE 23rd International Symposium on Quality of Service (IWQoS)*, pp. 177–182, IEEE, 2015.

[54] L. S. Peter, H. Kobo, and V. M. Srivastava, "A comparative review analysis of openflow and p4 protocols based on software defined networks," *Data Intelligence and Cognitive Informatics: Proceedings of ICDICI 2022*, pp. 699–711, 2022.

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

[56] S. Ali, M. Pandey, and N. Tyagi, "Sdfog-mesh: A software-defined fog computing architecture over wireless mesh networks for semi-permanent smart environments," *Computer Networks*, vol. 211, p. 108985, 2022.

[57] R. Rosen, "Linux containers and the future cloud," *Linux J*, vol. 240, no. 4, pp. 86–95, 2014.

[58] W. J. Lee, J. W. Shin, H. Y. Lee, and M. Y. Chung, "Testbed implementation for routing wlan traffic in software defined wireless mesh network," in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 1052–1055, IEEE, 2016.

[59] T. Osiński and H. Tarasiuk, "New approaches to data plane programmability for software datapaths in the nfv infrastructure," in *2023 IEEE 9th International Conference on Network Softwarization (NetSoft)*, pp. 320–325, IEEE, 2023.

[60] D. Levin, M. Canini, S. Schmid, F. Schaffert, and A. Feldmann, "Panopticon: Reaping the {Benefits} of incremental {SDN} deployment in enterprise networks," in *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pp. 333–345, 2014.

[61] M. Canini, A. Feldmann, D. Levin, F. Schaffert, and S. Schmid, "Software-defined networks: Incremental deployment with panopticon," *Computer*, vol. 47, no. 11, pp. 56–60, 2014.

[62] M. Swamy, K. Haribabu, A. Bhatia, *et al.*, "Achieving waypoint enforcement in multi-vlan hybrid sdn," in *2018 10th International Conference on Communication Systems & Networks (COMSNETS)*, pp. 519–521, IEEE, 2018.

[63] R. C. Sofia, "A survey of advanced ethernet forwarding approaches," *IEEE Communications surveys & tutorials*, vol. 11, no. 1, pp. 92–115, 2009.

[64] M. Yu, J. Rexford, X. Sun, S. Rao, and N. Feamster, "A survey of virtual lan usage in campus networks," *IEEE Communications Magazine*, vol. 49, no. 7, pp. 98–103, 2011.

[65] O. Tilmans and S. Vissicchio, "Igp-as-a-backup for robust sdn networks," in *10th International Conference on Network and Service Management (CNSM) and Workshop*, pp. 127–135, IEEE, 2014.

[66] J. Rekhter, "Nsfnet backbone spf based interior gateway protocol," in *Internet Request for Comments Series RFC 1074*, Citeseer, 1990.

[67] S. Vissicchio, L. Cittadini, O. Bonaventure, G. G. Xie, and L. Vanbever, "On the co-existence of distributed and centralized routing control-planes," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 469–477, IEEE, 2015.

[68] S. Vissicchio, L. Vanbever, and O. Bonaventure, "Opportunities and research challenges of hybrid software defined networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 70–75, 2014.

[69] S. Vissicchio, L. Vanbever, L. Cittadini, G. G. Xie, and O. Bonaventure, "Safe update of hybrid sdn networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1649–1662, 2017.

[70] J. Schot and A. Rip, "The past and future of constructive technology assessment," *Technological forecasting and social change*, vol. 54, no. 2-3, pp. 251–268, 1997.

[71] D. Kim, "A framework for hierarchical clustering of a link-state internet routing domain," in *International Conference on Information Networking*, pp. 839–848, Springer, 2003.

[72] L. Vanbever and S. Vissicchio, "Enabling {SDN} in old school networks with {Software-Controlled} routing protocols," in *Open Networking Summit 2014 (ONS 2014)*, 2014.

[73] S. Vissicchio, O. Tilmans, L. Vanbever, and J. Rexford, "Central control over distributed routing," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pp. 43–56, 2015.

[74] K.-T. Foerster, S. Schmid, and S. Vissicchio, "Survey of consistent software-defined network updates," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1435–1461, 2018.

[75] R. Hartert, S. Vissicchio, P. Schaus, O. Bonaventure, C. Filsfils, T. Telkamp, and P. Francois, "A declarative and expressive approach to control forwarding paths in carrier-grade networks," *ACM SIGCOMM computer communication review*, vol. 45, no. 4, pp. 15–28, 2015.

[76] Y. Peng, L. Guo, Q. Deng, Z. Ning, and L. Zhang, "A novel hybrid routing forwarding algorithm in sdn enabled wireless mesh networks," in *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety*

and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems, pp. 1806–1811, IEEE, 2015.

[77] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM computer communication review*, vol. 38, no. 2, pp. 69–74, 2008.

[78] Y. Guo, Z. Wang, X. Yin, X. Shi, J. Wu, and H. Zhang, "Incremental deployment for traffic engineering in hybrid sdn network," in *2015 IEEE 34th international performance computing and communications conference (IPCCC)*, pp. 1–8, IEEE, 2015.

[79] Y. Guo, Z. Wang, X. Yin, X. Shi, and J. Wu, "Traffic engineering in hybrid sdn networks with multiple traffic matrices," *Computer Networks*, vol. 126, pp. 187–199, 2017.

[80] Y. Guo, Z. Wang, X. Yin, X. Shi, and J. Wu, "Traffic engineering in sdn/ospf hybrid network," in *2014 IEEE 22nd International Conference on Network Protocols*, pp. 563–568, IEEE, 2014.

[81] M. Labraoui, M. M. Boc, and A. Fladenmuller, "Software defined networking-assisted routing in wireless mesh networks," in *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 377–382, IEEE, 2016.

[82] M. Labraoui, C. Chatzinakis, M. M. Boc, and A. Fladenmuller, "On addressing mobility issues in wireless mesh networks using software-defined networking," in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 903–908, IEEE, 2016.

[83] M. Labraoui, M. M. Boc, and A. Fladenmuller, "Opportunistic sdn-controlled wireless mesh network for mobile traffic offloading," in *2017 International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT)*, pp. 1–7, IEEE, 2017.

[84] M. Labraoui, M. Boc, and A. Fladenmuller, "Self-configuration mechanisms for sdn deployment in wireless mesh networks," in *2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1–4, IEEE, 2017.

[85] J. Wang, Y. Miao, P. Zhou, M. S. Hossain, and S. M. M. Rahman, "A software defined network routing in wireless multihop network," *Journal of Network and Computer Applications*, vol. 85, pp. 76–83, 2017.

[86] N. Makariye, "Towards shortest path computation using dijkstra algorithm," in *2017 International Conference on IoT and Application (ICIOT)*, pp. 1–3, IEEE, 2017.

[87] L. He, X. Zhang, Z. Cheng, and Y. Jiang, "Design and implementation of sdn/ip hybrid space information network prototype," in *2016 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, pp. 1–6, IEEE, 2016.

[88] L. He, X. Zhang, Z. Cheng, and Y. Jiang, "Design and implementation of sdn/ip hybrid space information network prototype," in *2016 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, pp. 1–6, IEEE, 2016.

[89] P. Jakma and D. Lamparter, "Introduction to the quagga routing suite," *IEEE Network*, vol. 28, no. 2, pp. 42–48, 2014.

[90] K. Poularakis, G. Iosifidis, and L. Tassiulas, "Sdn-enabled tactical ad hoc networks: Extending programmable control to the edge," *IEEE Communications Magazine*, vol. 56, no. 7, pp. 132–138, 2018.

[91] K. Poularakis, Q. Qin, K. M. Marcus, K. S. Chan, K. K. Leung, and L. Tassiulas, "Hybrid sdn control in mobile ad hoc networks," in *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 110–114, IEEE, 2019.

[92] S. Khuller, A. Moss, and J. S. Naor, "The budgeted maximum coverage problem," *Information processing letters*, vol. 70, no. 1, pp. 39–45, 1999.

[93] H. Xu, J. Fan, J. Wu, C. Qiao, and L. Huang, "Joint deployment and routing in hybrid sdns," in *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, pp. 1–10, IEEE, 2017.

[94] H. Xu, X.-Y. Li, L. Huang, H. Deng, H. Huang, and H. Wang, "Incremental deployment and throughput maximization routing for a hybrid sdn," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1861–1875, 2017.

[95] M. Caria and A. Jukan, "The perfect match: Optical bypass and sdn partitioning," in *2015 IEEE 16th International Conference on High Performance Switching and Routing (HPSR)*, pp. 1–6, IEEE, 2015.

[96] M. Caria, T. Das, A. Jukan, and M. Hoffmann, "Divide and conquer: Partitioning ospf networks with sdn," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 467–474, IEEE, 2015.

[97] M. Caria, A. Jukan, and M. Hoffmann, "Sdn partitioning: A centralized control plane for distributed routing protocols," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 381–393, 2016.

[98] J. Nunez-Martinez, J. Baranda, and J. Mangues-Bafalluy, "A service-based model for the hybrid software defined wireless mesh backhaul of small cells," in *2015 11th International Conference on Network and Service Management (CNSM)*, pp. 390–393, IEEE, 2015.

[99] M. Osman, J. Núñez-Martínez, and J. Mangues-Bafalluy, "Hybrid sdn: Evaluation of the impact of an unreliable control channel," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 242–246, IEEE, 2017.

[100] P. Dely, A. Kassler, and N. Bayer, "Openflow for wireless mesh networks," in *2011 proceedings of 20th international conference on computer communications and networks (ICCCN)*, pp. 1–6, IEEE, 2011.

[101] J. Vestin, P. Dely, A. Kassler, N. Bayer, H. Einsiedler, and C. Peylo, "Cloudmac: towards software defined wlans," in *Proceedings of the 18th annual*

international conference on Mobile computing and networking, pp. 393–396, 2012.

[102] D. Nguyen and P. Minet, “Analysis of mpr selection in the olsr protocol,” in *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, vol. 2, pp. 887–892, IEEE, 2007.

[103] 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.

[104] U. Ashraf, “Placing controllers in software-defined wireless mesh networks,” in *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, pp. 1–4, IEEE, 2018.

[105] S. Salsano, G. Siracusano, A. Detti, C. Pisa, P. L. Ventre, and N. Blefari-Melazzi, “Controller selection in a wireless mesh sdn under network partitioning and merging scenarios,” *arXiv preprint arXiv:1406.2470*, 2014.

[106] H. Elzain and W. Yang, “Qos-aware topology discovery in decentralized software defined wireless mesh network (d-sdwmn) architecture,” in *Proceedings of the 2018 2nd international conference on computer science and artificial intelligence*, pp. 158–162, 2018.

[107] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, *et al.*, “Onos: towards an open, distributed sdn os,” in *Proceedings of the third workshop on Hot topics in software defined networking*, pp. 1–6, 2014.

[108] O. Sefraoui, M. Aissaoui, M. Eleuldj, *et al.*, “Openstack: toward an open-source solution for cloud computing,” *International Journal of Computer Applications*, vol. 55, no. 3, pp. 38–42, 2012.

[109] S. Salsano, P. L. Ventre, F. Lombardo, G. Siracusano, M. Gerola, E. Salvadori, M. Santuari, M. Campanella, and L. Prete, "Mantoo-a set of management tools for controlling sdn experiments," in *2015 Fourth European Workshop on Software Defined Networks*, pp. 123–124, IEEE, 2015.

[110] S. Salsano, P. L. Ventre, L. Prete, G. Siracusano, M. Gerola, and E. Salvadori, "Oshi-open source hybrid ip/sdn networking (and its emulation on mininet and on distributed sdn testbeds)," in *2014 Third European Workshop on Software Defined Networks*, pp. 13–18, IEEE, 2014.

[111] S. Salsano, P. L. Ventre, F. Lombardo, G. Siracusano, M. Gerola, E. Salvadori, M. Santuari, M. Campanella, and L. Prete, "Hybrid ip/sdn networking: open implementation and experiment management tools," *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 138–153, 2015.

[112] A. Hakiri, A. Gokhale, and P. Patil, "A software defined wireless networking for efficient communication in smart cities," *Distrib. Object Comput.(DOC) Group, Vanderbilt Univ., Nashville, TN, USA, Tech. Rep*, 2017.

[113] A. Hakiri, B. Sellami, S. B. Yahia, and P. Berthou, "A sdn-based iot architecture framework for efficient energy management in smart buildings," in *2020 Global Information Infrastructure and Networking Symposium (GIIS)*, pp. 1–6, IEEE, 2020.

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

[115] N. Shastry and T. Keerthan Kumar, "Enhancing the performance of software-defined wireless mesh network," in *International Conference on Communication, Computing and Electronics Systems*, pp. 1–14, Springer, 2020.

[116] H. Elzain and Y. Wu, "Software defined wireless mesh network flat distribution control plane," *Future Internet*, vol. 11, no. 8, p. 166, 2019.

[117] E. Kuznetsova, Y. Avakyan, V. Dai Pham, and R. Kirichek, "Applying the concept of software-defined networking in wireless mesh network," in *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*, pp. 28–38, Springer, 2020.

[118] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using openflow: A survey," *IEEE communications surveys & tutorials*, vol. 16, no. 1, pp. 493–512, 2013.

[119] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, vol. 3, pp. 10–5555, 2010.

[120] T. Omizo, T. Watanabe, T. Akiyama, and K. Iida, "Resilientflow: Deployments of distributed control channel maintenance modules to recover sdn from unexpected failures," *IEICE Transactions on Communications*, vol. 99, no. 5, pp. 1041–1053, 2016.

[121] N. Katta, H. Zhang, M. Freedman, and J. Rexford, "Ravana: Controller fault-tolerance in software-defined networking," in *Proceedings of the 1st ACM SIGCOMM symposium on software defined networking research*, pp. 1–12, 2015.

[122] M. Hartung and M. Körner, "Softmon-traffic monitoring for sdn," *Procedia Computer Science*, vol. 110, pp. 516–523, 2017.

[123] H. I. Kobo, G. P. Hancke, and A. M. Abu-Mahfouz, "Towards a distributed control system for software defined wireless sensor networks," in *IECON 2017-43rd annual conference of the IEEE industrial electronics society*, pp. 6125–6130, IEEE, 2017.

[124] M. Osman and J. Mangues-Bafalluy, "Hybrid sdn performance: Switching between centralized and distributed modes under unreliable control communication channels," *Journal of Sensor and Actuator Networks*, vol. 10, no. 3, p. 57, 2021.

[125] N. L. Van Adrichem, C. Doerr, and F. A. Kuipers, "Opennetmon: Network monitoring in openflow software-defined networks," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, pp. 1–8, IEEE, 2014.

[126] M. Malboubi, L. Wang, C.-N. Chuah, and P. Sharma, "Intelligent sdn based traffic (de) aggregation and measurement paradigm (istamp)," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pp. 934–942, IEEE, 2014.

[127] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "Opentm: traffic matrix estimator for openflow networks," in *International Conference on Passive and Active Network Measurement*, pp. 201–210, Springer, 2010.

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

[129] M. Yu, L. Jose, and R. Miao, "Software {Defined}{Traffic} measurement with {OpenSketch}," in *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pp. 29–42, 2013.

[130] M. Moshref, M. Yu, R. Govindan, and A. Vahdat, "Dream: dynamic resource allocation for software-defined measurement," in *Proceedings of the 2014 ACM conference on SIGCOMM*, pp. 419–430, 2014.

[131] M. Moshref, M. Yu, and R. Govindan, "Resource/accuracy tradeoffs in software-defined measurement," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pp. 73–78, 2013.

[132] M. Hartung and M. Körner, “Softmon-traffic monitoring for sdn,” *Procedia Computer Science*, vol. 110, pp. 516–523, 2017.

[133] H. Yahyaoui and M. F. Zhani, “On providing low-cost flow monitoring for sdn networks,” in *2020 IEEE 9th International Conference on Cloud Networking (CloudNet)*, pp. 1–6, IEEE, 2020.

[134] P. Bellavista, A. Dolci, C. Giannelli, and D. D. Padalino Montenero, “Sdn-based traffic management middleware for spontaneous wmns,” *Journal of Network and Systems Management*, vol. 28, no. 4, pp. 1575–1609, 2020.

[135] J. Haxhibeqiri, P. H. Isolani, J. M. Marquez-Barja, I. Moerman, and J. Hoebeke, “In-band network monitoring technique to support sdn-based wireless networks,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 627–641, 2020.

[136] S. Vissicchio, L. Vanbever, and O. Bonaventure, “Opportunities and research challenges of hybrid software defined networks,” *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 70–75, 2014.

[137] Y. Guo, Z. Wang, X. Yin, X. Shi, and J. Wu, “Traffic engineering in hybrid sdn networks with multiple traffic matrices,” *Computer Networks*, vol. 126, pp. 187–199, 2017.

[138] 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.

[139] A. Hakiri, A. Gokhale, and P. Berthou, “Software-defined wireless mesh networking for reliable and real-time smart city cyber physical applications,” in *Proceedings of the 27th International Conference on Real-Time Networks and Systems*, pp. 165–175, 2019.

[140] Y. Zhang, “An adaptive flow counting method for anomaly detection in sdn,” in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, pp. 25–30, 2013.

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

[142] Y. Chai and X.-J. Zeng, "The development of green wireless mesh network: a survey," *Journal of Smart Environments and Green Computing*, vol. 1, no. 1, pp. 47–59, 2021.

[143] N. L. Van Adrichem, C. Doerr, and F. A. Kuipers, "Opennetmon: Network monitoring in openflow software-defined networks," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, pp. 1–8, IEEE, 2014.

[144] M. Malboubi, L. Wang, C.-N. Chuah, and P. Sharma, "Intelligent sdn based traffic (de) aggregation and measurement paradigm (istamp)," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pp. 934–942, IEEE, 2014.

[145] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "Opentm: traffic matrix estimator for openflow networks," in *International Conference on Passive and Active Network Measurement*, pp. 201–210, Springer, 2010.

[146] M. Moshref, M. Yu, R. Govindan, and A. Vahdat, "Dream: dynamic resource allocation for software-defined measurement," in *Proceedings of the 2014 ACM conference on SIGCOMM*, pp. 419–430, 2014.

[147] W. Wang, C.-H. Wang, and T. Javidi, "Reliable shortest path routing with applications to wireless software-defined networking," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2018.

[148] D. Sajjadi, Z. Zheng, R. Ruby, and J. Pan, "Randomized single-path flow routing on sdn-aware wi-fi mesh networks," in *2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pp. 184–192, IEEE, 2018.

[149] Y. Wang, T.-Y. C. Tai, R. Wang, S. Gobriel, J. Tseng, and J. Tsai, "Optimizing open vswitch to support millions of flows," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–7, IEEE, 2017.

[150] T. Clausen and P. Jacquet, "Optimized link state routing protocol (olsr)," tech. rep., 2003.

[151] D. E. Henni, A. Ghomari, and Y. Hadjadj-Aoul, "A consistent qos routing strategy for video streaming services in sdn networks," *International Journal of Communication Systems*, vol. 33, no. 10, p. e4177, 2020.

[152] A. Eftimie and E. Borcoci, "Sdn controller implementation using opendaylight: experiments," in *2020 13th International Conference on Communications (COMM)*, pp. 477–481, IEEE, 2020.

[153] D. Johnson, N. S. Ntlatlapa, and C. Aichele, "Simple pragmatic approach to mesh routing using batman," 2008.

[154] A. Tirumala, "Iperf: The tcp/udp bandwidth measurement tool," *http://dast. nlanr. net/Projects/Iperf/*, 1999.

[155] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a better netflow," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 245–256, 2004.

[156] J. Reves and S. Panchen, "Traffic monitoring with packet-based sampling for defense against security threats," *InMon Technology Whitepaper*, vol. 78, 2002.

[157] A. C. Myers, "Jflow: Practical mostly-static information flow control," in *Proceedings of the 26th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pp. 228–241, 1999.

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

[159] Y.-T. Han, E.-M. Lee, H.-S. Park, J.-Y. Ryu, C.-C. Kim, and M.-W. Song, "Test and performance comparison of end-to-end available bandwidth measurement tools," in *2009 11th International Conference on Advanced Communication Technology*, vol. 1, pp. 370–372, IEEE, 2009.