

Received June 15, 2020, accepted July 9, 2020, date of publication July 27, 2020, date of current version August 5, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3011651

SDNMesh: An SDN Based Routing Architecture for Wireless Mesh Networks

**SYED SHERJEEL A. GILANI^{1,2}, AMIR QAYYUM², (Senior Member, IEEE),
RAO NAVEED BIN RAIS³, (Member, IEEE), AND MUKHTIAR BANO^{2,4}, (Member, IEEE)**

¹Computer Science Department, Riphah International University, Islamabad 44000, Pakistan

²Electrical Engineering Department, Capital University of Science and Technology, Islamabad 44000, Pakistan

³Electrical Engineering Department, Ajman University, Ajman, United Arab Emirates

⁴Software Engineering Department, Fatima Jinnah Women University, Rawalpindi 46000, Pakistan

Corresponding author: Syed Sherjeel A. Gilani (sherjeel.gilani@riphah.edu.pk)


The research of this article was funded by the Ajman University, UAE, through the Deanship of Graduate Studies and Research (DGSR) Program.

ABSTRACT Software Defined Networking (SDN) has been seen as a revolutionary and exciting network technology that aims to enable control and network management of various network types, whether wired or wireless. Nevertheless, SDN research focuses very little on wireless communication and, more specifically, on Wireless Mesh Networks (WMNs). Moreover, the issue of routing is vitally important in WMNs, but the legacy and traditional routing protocols cannot make the most of multiple paths between the source node and destination node due to the complexity and cost of the network. In this paper, we present SDNMesh, an SDN based routing architecture for WMNs. We combine SDN with WMN to allow mesh networks to meet current user requirements with several resources, coverage, and scalable high bandwidth capability. Apart from the mentioned capability, SDN's unified approach leads to better network capacity management. Experiments have been carried out using the Mininet-WiFi simulation tool to create a network environment that allows integration of the two networking paradigms, centralized, and decentralized. Simulation results show that our SDNMesh routing solution performs better in terms of network performance metric throughput, packet loss ratio, and delay while comparing with traditional routing approaches such as OLSR, BATMAN, and an SDN based Three-Stage routing protocols. Moreover, experimental results show that SDNMesh gives better results in terms of the mentioned performance metrics.

INDEX TERMS Control plane, data plane, programmable, routing, software defined networking, wireless mesh networks.

I. INTRODUCTION

Current users' demands and the rapidly increasing use of the network make the management of conventional legacy networks more complex and difficult to monitor. Nevertheless, different types of network technologies such as cloud computing, big data, and multimedia applications not only represent a major source of revenue generation but also result in significant operational and efficiency challenges for network operators [1]. Efficiency and flexibility remained a major requirement for current networks to tackle these challenges. Hence, Software Defined Networking (SDN) is one possible solution. The basic principle of operation in the SDN paradigm is the centralization of control and administration,

The associate editor coordinating the review of this manuscript and approving it for publication was Zonghua Gu .

a true solution for the management of networks, and the regulation of related issues. SDN breaks down the control plane data plane [2], [3]. SDN is a modern networking model, ensuring a significant reduction in network management complexity, facilitating innovative progress, and network programmability transformation [4], [5]. SDN is currently deployed in integration with various wireless environments and applications such as network virtualization, handover methods, interference management, load balancing, wireless network heterogeneity, routing strategies [6], [7]. The SDN architecture (Figure 1) is beneficial in the following respects over conventional network architecture;

1. Changing traffic engineering policies, for example, choosing a new outgoing traffic gateway, routing all traffic using any given firewall, routing some traffic differently

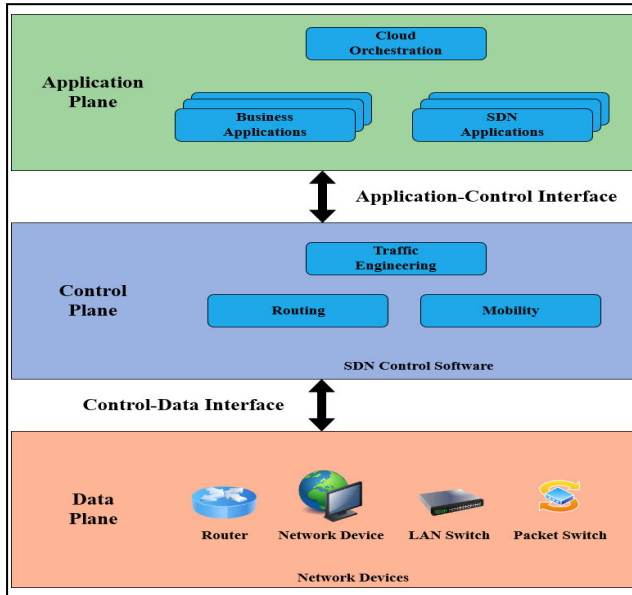


FIGURE 1. The architecture of Software Defined Networking (SDN).

2. Offline or runtime traffic such as finer granularity in timing for taking decisions to route this traffic and for the traffic flows affected

3. Creation of creative services such as bundles can be managed in different ways depending on the customer or the framework

4. Together with Network Function Virtualization such as dynamically activating network functions, services delivery, and allowing network operators to gain control of their network [8].

Wireless Mesh Networks (WMNs) architecture has been increasingly deployed in modern communication systems as well as applications for Internet access. Among other wireless architectures such as Adhoc networks, MANETs, etc., WMN remained a long-standing and reliable network architecture. The range of WMN network nodes such as switches, laptops, and routers, naturally makes it versatile. Nevertheless, association and disassociation of these nodes with the network can lead to topological dynamics and variability in communication requirements, which is a major cause for making WMNs difficult to manage [9], [10]. Besides, the limited number of gateway routers in WMN leads to the increasing problem of congestion. WMNs need to incorporate effective load management, traffic engineering, and resource allocation to alleviate these issues [11]. Furthermore, multi-hop setting induces network performance degradation and packet loss [12]. Likewise, other variables such as variable quality of links, network asymmetry, and traffic load are also major challenges that need to be addressed [13], [14].

Furthermore, their alteration can require complete or partial replacement of hardware equipment in WMNs network nodes once installed and deployed, resulting in significant increases in operating costs. Thus, making WMN programmable can allow changes to be implemented using

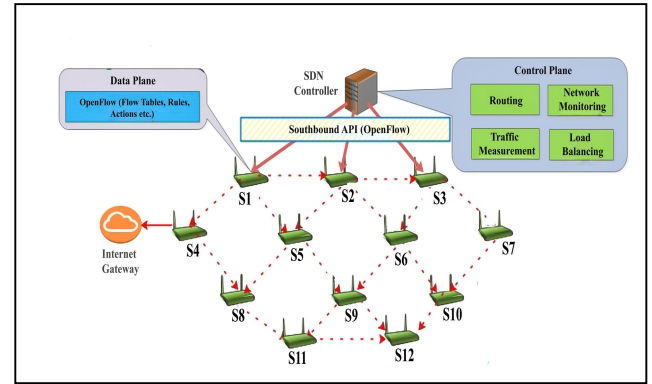


FIGURE 2. SDN enabled wireless mesh network.

software programs [15]. The routing protocols currently used in WMNs include DSR, AODV, DSDV, OLSR, and BATMAN. All of these routing approaches show a high decrease in terms of average throughput and do not provide support for mobility management [13]. However, OLSR and BATMAN protocols show a good performance for multi-hop communications. Generally, the OLSR protocol shows better performance than the BATMAN protocol [37], [38]. The foregoing problems of the design and implementation of WMNs can be effectively solved by applying the SDN approach. Using a logically centralized controller to remotely control and configure mesh nodes and to render them simple data forwarders (Figure 2) will achieve the goal of programmable WMNs. Additionally, congestion control and load balancing policies are implemented to improve traffic management and load balancing. However, a single controller may compromise the reliability of the network, as this fault tolerance must also be considered while adjusting the SDN paradigm [12]. Network operators can implement different QoS policies while keeping user and application requirements in view [17], [18]. Our proposed solution, SDNMesh presents an SDN based two-stage routing architecture, where the first phase finds the initial route from controller towards switches and the second phase does optimization of the inefficient routes found from the first phase. The proposed architecture also addresses link or node failure which is one of the most significant challenges of the wireless environment. The main contributions of this paper are as follows:

1. We present a two-phase routing architecture for SDN-enabled WMN that addresses the dynamic configurations for the SDN controller to react efficiently to topological changes of the network keeping in view the mobility of mesh nodes.
2. We propose a modification in the OpenFlow protocol by defining new messages to create efficient communication between SDN nodes and controller. The modification is orthogonal with the traditional behavior of the OpenFlow protocol and does not disturb its functionality.
3. We perform simulations using Mininet-Wifi to give the comparison of average throughput between the

proposed routing architecture with conventional routing protocols such as OLSR and BATMAN.

The rest of the paper is organized as follows; Section II presents the related work to the SDN based WMN routing architectures, Section III elaborates proposed routing architecture, and Section IV presents experimental evaluation and analysis of simulation results while Section V concludes the paper.

II. EXISTING SDN ENABLED WMN ROUTING ARCHITECTURES

This section presents background on SDN based solutions for WMN, particularly focusing on routing. Keeping in view the deployment method of SDN in WMN, the routing architectures are categorized into two major classes 1) Pure SDN-WMN Routing and 2) Hybrid SDN-WMN Routing.

A. PURE SDN-WMN ROUTING ARCHITECTURES

Pure SDN-WMN deployment includes SDN switches in topology along with controllers, with a focus on routing architecture. Rawat *et al.* [19] present a comprehensive study of different approaches for networks defined by wireless software, focusing on their design challenges and benefits. In terms of different types of wireless networks such as cellular networks, wireless sensor networks, wireless mesh networks, and home networks, the authors present SDN requirements. This study, however, neither provides a platform for implementation nor any evaluation. Patil *et al.* [20] propose a three-stage routing strategy for the efficient use of SDN technology in WMN. They extend the OpenFlow protocol, using three new messages to promote the routing technique suggested. Quality metric latency is used to equate a hybrid approach to the method they propose.

Sajjadi *et al.* [21] propose a fine-grained routing approach to optimize SDN-conscious WMN traffic engineering while addressing the issue of the connection between three main operations such as gateway selection, AP affiliation, and flow routing. This optimization aims to optimize network efficiency to meet predefined QoS constraints. Hakiri *et al.* [22] present a division of routing functionalities based on two layers; one is designed to support data transmission via OpenFlow protocol, and the other uses IP transmission through conventional OLSR routing protocol. For communication control policies, OpenFlow is used. Li *et al.* [23] introduce Protocol Oblivious Source Routing (POSR), an SDN-based routing approach that they claim to be a protocol-independent and bandwidth-efficient packet forwarding approach. The packet format is also planned in tandem with unicast and multicast packet processing pipelines as well as recovery from link failure. Besides, a testbed is designed for experimental evaluation of POSR. This method could reduce the use of flow tables and latency to determine the shortest path. Yaghoubi *et al.* [24] suggest an ambitious control policy for weather-stricken SDN networks. The routing optimization and optimal route series are also done through the policies.

Experimental testing suggests improved network capacity through the implementation of proposed policies and a wise selection of optimal routes. It also aims to work on the best routes for future Network states to have practical forecasts and decision-making. Bao *et al.* [25] propose a traffic control and mobility management solution for SDN-based WMNs by leveraging the centralized control features. The authors also suggest developing a two-layer Support Vector Machine (SVM) based connection failure prediction scheme which is a machine learning algorithm.

TABLE 1. Pure SDN-WMN routing architectures.

Architecture	Features	Control Traffic Routing	Data Traffic Routing	Challenges
Patil <i>et al.</i> [20]	Routing, Traffic Engineering	Open Flow	OpenFlow	Link failure while connection establishment
Sajjadi <i>et al.</i> [21]	Integrating cellular with Wi-Fi networks, Traffic	Open Flow	OpenFlow	No of Gateways, the symmetry of the grid and the position of stations
Hakiri <i>et al.</i> [22]	Routing, Traffic Engineering	Open Flow	OLSR	Integration of OpenFlow and OLSR
Li <i>et al.</i> [23]	Protocol independent forwarding,	Open Flow	OpenFlow	Differentiated services
Yaghoubi <i>et al.</i> [24]	Routing based on timing and sequence of paths of flows	Open Flow	OpenFlow	Rain attenuation
Bao <i>et al.</i> [25]	Traffic balancing, reduced response time, node mobility prediction	Open Flow	OpenFlow	Node mobility, link failure

Table 1 summarizes the SDN-WMN routing-based architectures while presenting implementation challenges including connection failure, OpenFlow integration, and conventional routing protocols, differentiated services, node mobility, etc. Besides, another issue is also highlighted regarding the number of gateways and stations.

B. HYBRID SDN-WMN ROUTING ARCHITECTURES

Presently, various hybrid routing schemes generally use SDN technology in WMNs to address the issues of load balancing, traffic engineering, and mobility management. Such kind of hybrid implementation involves topology SDN switches along with legacy switches. OpenFlow protocol is implemented in SDN switches whilst legacy routing protocols (AODV, OSPF, OLSR, etc.) are implemented in legacy switches (Figure 3). Panopticon [26] suggests hybrid SDN architecture, concentrating on the question of interconnectivity between legacy switches and SDNs. In this architecture,

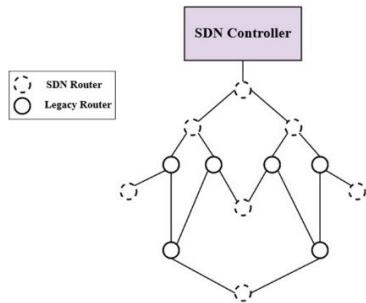


FIGURE 3. Hybrid SDN-WMN architecture.

communication of network, layers are not considered. Vissicchio *et al.* [27] propose various types of hybrid SDN models based on service type, class, and topology. The deployment architecture of HRFA [28] contributes to better traffic forwarding. The proposed architecture is based on OSPF [29] and OpenFlow protocol [30], while considering the use of links and the number of simulation performance metrics for hops. This provides a significant reduction in load balancing and network congestion. Guo *et al.* [31] propose an architecture that focuses on the transformation from the conventional to the SDN model, considering the use of traffic engineering. The author uses a genetic algorithm to classify the sequence of nodes migrated.

Labraoui *et al.* [32] introduces a hybrid routing architecture using OLSR SDN, intending to study that traditional controller-assisted routing would result in better performance and control. Reliability is also enhanced but the overhead effect of the network is increased linearly according to the scale of the network. Nevertheless, the distribution ratio of the throughput and the packets is increased. Wang *et al.* [33] propose an architecture addressing QoS Routing, which considers a multi-hop wireless network integrated with SDN using residual energy and hop count as performance metrics for selecting pathways. Multipoint Relay (MPR) principle is applied to optimize transmissions. The controller determines the shortest path and uses the Dijkstra Algorithm to get multiple paths. HEATE [34] proposes tackling the challenge of traffic engineering while considering energy efficiency. The design is a routing protocol based on OSPF. It introduces the splitting of traffic flow to avoid congestion and use multiple paths to forward the flow in parts. He *et al.* [35] proposes an architecture focused on network grouping into zones such as ground network follow IP routing and SDN routing follows space network follow. Hakiri *et al.* [36] present a Cyber-physical wireless communication system in smart cities that are scalable, reliable, and predictable to support real-time applications. The authors introduce a novel architecture based on a symbiotic relationship between wireless mesh networks (WMNs) and software-defined networking (SDN).

Table 2 summarizes the hybrid SDN-WMN routing schemes in which [26]–[27] and [31] consider the physical location of the controller in topology, the number of SDN nodes, and scalability of controllers as the most significant

TABLE 2. Hybrid SDN-WMN routing architectures.

Architecture	Features	SDN Routing Protocol	IP Routing Protocol	Challenge
Panopticon [26]	Load balancing, fault tolerance, link recovery	Open - Flow	Shortest path routing	Location and number of SDN switches
Vissicchio [27]	Flexibility, partial robustness	Open - Flow	Shortest path routing	Topological position and scalability
HRFA [28]	Load balancing, congestion control and making traffic fast-forwarding	Open - Flow	OSPF	Information exchange b/w SDN and IP nodes
Guo [31]	Traffic Engineering and link utilization	Open - Flow	OSPF	Migration sequence of SDN nodes
Labraoui [32]	Reliability and performance improvement	Open - Flow	OLSR	One hop communication b/w Controller and nodes
Wang [33]	QoS Routing, Single/multipath routing and disjoint multipath	Open - Flow	OLSR	Expansion in network size
HEATE [34]	Traffic engineering, Energy saving	Open - Flow	OSPF	Energy consumption by link,
He et al. [35]	Collaborative management,	Open - Flow	RIP and OSPF	Information exchange b/w SDN and IP nodes
Hakiri et al. [36]	Routing, Traffic Engineering	OpenFlow	OLSR	Integration of OpenFlow and OLSR

challenges for operating SDN controllers. These issues can be very critical, especially in large or highly dynamic networks where controllers may need to make quick decisions on high-frequency events such as connection failures, dynamic traffic demands, regular arrival of new flows, etc. [28] and [35] find the sharing of topological information between traditional routers and SDN switches to be the most critical issue for topology-based deployment.

III. PROPOSED SDN BASED WMN ROUTING ARCHITECTURE

SDNMesh extends the working of the OpenFlow protocol, enabling the current OpenFlow protocol to implement WMNs

TABLE 3. Messages used in SDNMesh.

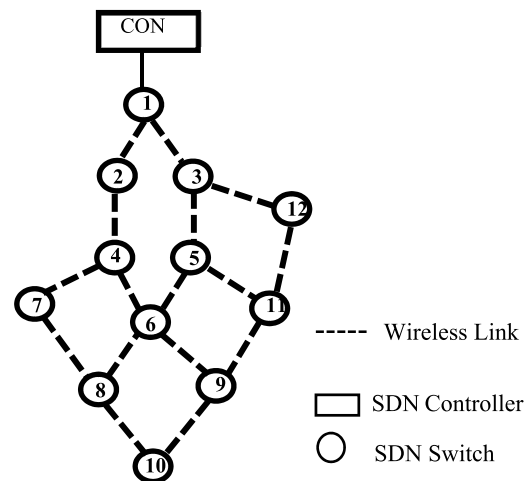
Messages	Main fields	Send mode	Pathway
PATH-REQ	Cont. id, Seq.	Broadcast	Controller - MAPs
PATH-RES	Device id, Seq, Neighbor list	Unicast	MAPs - Controller
PATH-UPD	Source IP, Source MAC, Dest. IP, Dest. MAC, Seq.	Unicast	Controller - MAPs
CURRENT-STAT	Device id, Total capacity of Interface, Current load (Queue length), Layer 2 protocol, Seq.	Unicast	MAPs - Controller
NEIG-TAB	Device id, Interface Id (IP), Neighbor MAC, Number of Neighbors	Unicast	MAPs - Controller
INT-LIS	Device id, Number of interfaces, Type of interfaces	Unicast	MAPs - Controller
THRESHOLD-SET	Device id, Interface id, Threshold Bandwidth	Unicast	Controller - MAPs
FED-BACK-CON	Device id, Message id, delay	Unicast	Controller - MAPs
ERROR	Device id, Neighbor list	Unicast	MAPs - Controller

with its dynamic nature. Modification of OpenFlow client has been done to achieve the appropriate working of proposed routing architecture. The optimization Algorithm is under consideration for adoption into future research. Options of using Dijkstra for maximum weight calculation, Round Robin, or Weighted average are available but currently, all possible combinations calculation is used with the assumption of ideal computational power is available for simplicity of the problem.

A. ROUTING SCHEME

SDNMesh routing approach is based on two phases, where the first phase finds the initial route from controller towards switches which may be inefficient in terms of delay, and the second phase does optimization of initial and inefficient routes found from the first phase while taking some more delay. This approach helps find the routes between SDN switches. Our proposed routing architecture also addresses link or node failure which is one of the most significant challenges of the wireless environment. OpenDaylight controller is added to the network. The controller broadcasts messages to all Mesh Access Points (MAPs), which are sensing for the information. The MAPs reply to the Controller with their information and the Controller forms a global network view based on that information. When a station wants to send data to some other node in the network, the directly connected MAP looks for the entry in its forwarding table; if the entry found, it forwards the packet to the required destination. If there is no entry, it requests the Controller to get the desired path. The stepwise working of the proposed architecture is as follows:

- i. **Controller to Switch Connection Establishment:** In Figure 4, it can be seen that the controller is connected with only one switch directly, hence the first task of the SDNMesh is to establish the connection between the controller and all the switches by applying the initial routing. Initially, the path towards all the switches is

**FIGURE 4.** Topology for SDNMesh architecture.

found through flooding by the SDN controller, and information about the directly connected switches is broadcast by the SDN controller. For this purpose, an OpenFlow based routing algorithm is used.

- ii. **Controller to Switch Shortest Path Establishment:** Routing paths are set up after the connection establishment between the SDN Controller and all the connected Switches. When the network starts, the connection is set up by establishing an initial path that is used to install a new alternative path, i.e., the shortest path. As the controller gets the knowledge of the network and a global view of the topology, it runs the Shortest Path Algorithm to acquire the shortest path between the Controller and SDN Switches. The decision making about choosing a path between the switch and the Controller among the given alternative paths (either the shortest or the initial) is made by the Controller. After deciding the specific path, the corresponding rules are installed in the respective switches to route packets

from the switches to the Controller by sending the information messages using the initial path.

- iii. **Switch to Switch Path Establishment:** The shortest path among switches is calculated by the controller using the Shortest Path Algorithm and corresponding forwarding rules are installed into switches using the established shortest path between controller and switch previously.

The complete list of messages (Table 3) used in the SDN-Mesh routing approach and their working is given as:

- i. The Controller broadcast information to every MAP using PATH-REQ, and the nodes establish the path to the Controller using the backward path.
- ii. The MAPs in the network send their Neighbor list to the Controller using PATH-RES. The Controller, on receiving the MAPs information, create the network overall topology view.
- iii. If the source node is to send data to the destination without path information, the MAP directly connected to the sender node (if it is not a Gateway) sends the data to its default Gateway and the Gateway subsequently sends a path information request to the Controller.
- iv. The Controller calculates the shortest path from sender to receiver by using hop count. For optimum path calculation, the Controller uses the network-wide map and the Dijkstra's algorithm. The calculated path information is sent to the sender node using PATH-UPD.
- v. CURRENT-STAT message is used by the MAPs to give information about the total capacity of the interfaces and the current load. The Controller, based on this information, suggests the threshold bandwidth to the MAPs using THRESHOLD-SET message. CURRENT-STAT is a periodic and a unicast message.
- vi. MAPs send the information of their interfaces using the INT-LIS message to the Controller. This message is helpful for the Controller to get the information of interfaces in a MAP and their types.
- vii. In the case of Gateway or MAP saturation, intermediate MAP failure, or Gateway failure situations, the Controller tries all possible combinations from the sender to the receiver nodes and selects the optimum path.
- viii. In case there is some change in the neighbor table of any node, the change is communicated to the Controller through NEIG-TAB.
- ix. The Controller, using FED-BACK-CON restricts the MAPs to stop sending any control messages for some specific time, if required.
- x. In case of any error, the MAPs send ERROR to the Controller so that the Controller takes further necessary action.

B. OPTIMIZATION ALGORITHM

The optimization algorithm aims to find a path towards the most under-utilized gateway link present in the network.

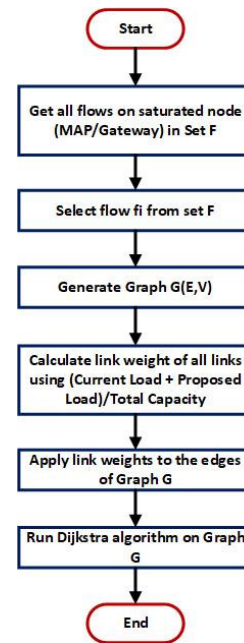


FIGURE 5. Optimization algorithm flow graph.

Figure 5 shows the flow graph of the Optimization Algorithm. The algorithm starts by selecting a source flow (s) on the saturated link responsible for initiating the optimization algorithm. The source flow is currently selected at random. Once the flow is selected link utilization of each link after application of the source flow is calculated. The resultant graph is called a link utilization (G) graph. From the link utilization graph all links which have weights greater than threshold value are removed. Once the links are removed a gateway (GW) a link is selected as the target and Dijkstra's algorithm is run on the graph G. If the removal of links has partitioned the graph such that no path can be found for source flow (s) towards GW then a new GW is selected from the set of Gateways. The process is iterated for all Gateways until a path is found. If no path can be found then the source flow (s) is deemed unmovable and a new source flow is selected from set S. If no source flow can be moved from the congested link then the network is considered to be optimal and not modified. The details of the algorithm are given as:

C. WORKFLOW OF SDNMesh MESSAGES

- a) PATH-REQ: In the beginning, this OpenFlow Path Request message is sent by the Controller to its directly connected switch (Figure 6) through wired or wireless connection. Upon receiving this message by the switch, it updates its path to the
- b) The controller based on source ID present in the received message. A new PATH_REQ is created by the respective switch using its source ID and it forwards the message to other switches except the one from which the request message is received. Every switch performs this step periodically with an interval of 2s,

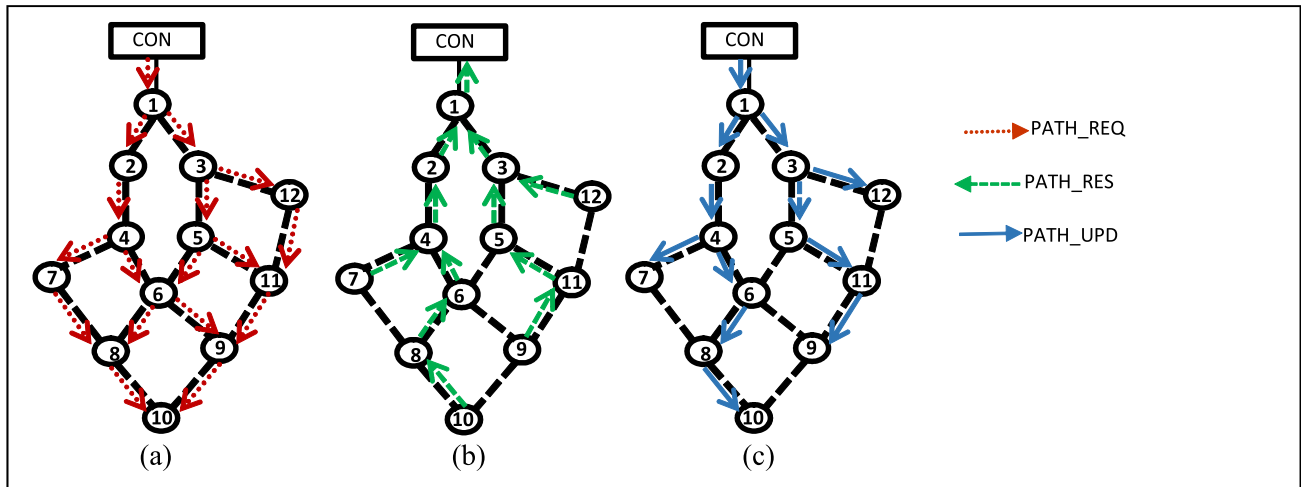


FIGURE 6. OpenFlow message exchange.

Algorithm 1 Algorithm: Optimization Algorithm

L is the set of links

P is the path

S is the set of Sources traversing the congested link

GW is a set of Gateway links for (s in S)

Calculate link utilization graph G

all links that are fully loaded are removed from graph G
fully loaded means more than the threshold (80%)
constant traffic

Sort GW links in ascending order

for (g in GW)

$P = \text{Run_Dijkstra}(G, s, g) // \text{Run_Dijkstra}(\text{Graph},$
 $\text{source}, \text{target})$

if P is found

then end for loop

else

continue for loop

end for loop

and in this way, switches can establish the initial path to the Controller, which may not be the shortest. The periodical broadcasting of the path request message addresses the issue of mobility, node, and link failure in the network.

- c) **PATH_RES**: This message is sent by the switch to the Controller in response to the earlier received **PATH_REQ**, it is forwarded to the neighbor from which the switch receives **PATH_REQ** first. This response message is sent using the initial path containing SSIDs of all the neighboring switches.
- d) **PATH_UPD**: The path established at the start between the Controller and switch is optimized by using this message, and is sent to the switches for updating their path to the Controller. After receiving this message, the switch installs a new path in its forwarding table.

Thus, whenever the switch sends any message to the Controller, it uses the shortest path. However, in case of failure of the shortest path, the initial path is used.

- e) **ERROR**: It is a path error message, initiated by a switch which finds any link failure along with its corresponding interface, and is forwarded towards the Controller for the modification of routes and the respective routing rules. It is triggered in case of delay critical applications and the data packets.

Figure 6 shows the connection establishment between the SDN Controller and SDN Switches using above OpenFlow messages. Initially, the Controller sends the **PATH_REQ** to its directly connected neighbor switch 1 (Figure 6(a)), which then forwards the same message after making its duplicate to its directly connected neighbor switches (2 and 3) through flooding. It leads to finding the initial path of every switch towards the controller. In response to **PATH_REQ**, each switch sends **PATH_RES** containing neighbors' information (Figure 6(b)), towards the Controller by using the initial path, which allows the Controller to get the information of neighbors of each switch. In this way, it learns the topology of the entire network. In Figure 6(a), switch 6 receives **PATH_REQ** from switch 4 and 5 respectively, however it sends **PATH_RES** through switch 4 as it receives **PATH_REQ** from switch 4 first. Similarly, switch 10 sends the **PATH_RES** message through switch 8. After getting the knowledge of the entire topology, the Controller derives the shortest path routes from each switch to itself and between the switches. Finally, the Controller installs the updated OpenFlow rules for the shortest path routes among all the switches using **PATH_UPD** through the shortest path as shown in Figure 6(c). The use of timers is only proposed to overcome the issue of lost control packets and the resulting divergence of the network. The control channel in SDNs is usually run on reliable protocols (TCP) and is proposed to be encrypted. This leaves very little room for any de-synchronization among network

elements. This means that we can have very large timer values, and can perform this synchronization sweep when the traffic load is low. We have currently not performed such a study due to the minimal impact of this scenario. Moreover, the time interval of 2s has been introduced for path request message to broadcast periodically by the controller.

IV. IMPLEMENTATION DETAILS

This section presents the implementation details of SDNMesh. The proposed network architecture is mainly based on backbone routing for SDN-enabled WMN. The topology under consideration comprises an SDN Controller connected with SDN nodes as forwarding elements (Figure 2). Each SDN node forwards OpenFlow messages using the OpenVSwitch soft router that implements a software pipeline based on flow tables. We assume the underlying wireless technology to be IEEE 802.11n, which provides an end throughput of ~150Mbps approx.

TABLE 4. Variables used in the mathematical model of SDNMesh.

S. No	Symbol	Variable
1	Mx	Mesh Access Point
2	C _{i,j}	Link capacity between node i and j
3	f _i	The flow of node i in kbps
4	H _{f_i}	Hop count of flow f _i
5	f _{z,m}	Flow present on access radio of MAP "m" in kbps
6	S _{f_i}	Mobility speed of flow f _i
7	Q _{mx}	Queue length at MAP Mx

Table 4 provides the details of notations that are used while modeling the solution mathematically. Where

- C_{i,j} < 150Mbps i.e. Capacity of the link between nodes i and j is less than 150 Mbps. We assume that our network will be limited to six hops to simplify the exposition.
- H_{f_i} < 6 where H_{f_i} is the hop count for flow f_i.
- To avoid the addition of unnecessary delay to traffic the hop count limit has been imposed and is, therefore, an objective of the final optimization algorithm.
- min (H_{f_i}); it is a minimum of hop count

Hence the approximate throughput capacity of 30 Mbps for the last hop is assumed to be guaranteed.

- Σ z = 0 Mx. f_z <= 30 Mbps; where Mx is the MAP under observation and f_z is a flow on the access radio.

For application layer throughput calculations, we are assuming the usage pattern of social media applications (like sharing of images and voice messages or advertising, etc.) as a use case and therefore assume a traffic load of approximately 150Kbps per user.

- max (f_i) < 150 Kbps

This yields a maximum of 200 users' capacity per MAP.

- Σ z = 0 Mx f_z / max (f_i)

For our exposition, we are assuming a campus network with an average of 20 nodes connected via each MAP. Since these nodes are mobile we can assume a possible density of up

to 40 nodes with any MAP. Node mobility speed is assumed to be 1.4 m/s which is considered to be average walking speed.

- S_{f_i} ≅ 1.4 m/sec where S_{f_i} is the mobility speed of flow f_i generated by a node.

All the nodes in the network send their statistics and hardware characteristics to the Controller. Using this information, the Controller easily figures out the Gateways. For instance, the Gateways normally have two live radios i.e. one for the network node connectivity and other for Internet connectivity. To assess the performance of the WMN powered SDNMesh routing architecture, we have identified five types of scenarios that investigate the deployment challenges facing SDN. In the specified scenarios, we assign data traffic to the multiple numbers of hops between a source-destination pair. At each step, we increase the size of the network to see the impact of the proposed routing architecture on efficiency. Due to interference, average UDP latency, and end-to-end delay, the number of hops between the source-destination pair can help us observe the decrease in the packet. This will also help us compare the conventional routing protocols OLSR and BATMAN with the SDN approach concerning their capacity in handling multiple hops. The details of the scenarios are given as:

1. Gateway saturated
2. Intermediate MAP Failure
3. MAP Saturated
4. Controller failure
5. Node Mobility

1) GATEWAY SATURATED

When the Gateway gets saturated, Controller detects the saturation through either of the two techniques, a) Through Link statistic (Periodic polling), and b) Event-based (Traps).

To examine saturation, the queue length variable is observed.

- Q_{Mx} < 5; where Q_{Mx} is the queue length at MAP Mx.

The controller decides which nodes traffic should be redirected to another Gateway, based on the source IP as the identifier of the node. As the gateways have WAN links that have much lower capacity than 802.11n radios we can expect these nodes to generate recalculation events much more frequently. By eliminating the hop count metric in the calculation of the optimum path, our optimization algorithm guarantees that all gateway nodes are fully utilized albeit at the cost of adding significant delay to the traffic.

It should be noted that if no alternate path is found then the controller will send the feedback control message to delay the possible re-initiation of the MAP saturation event.

2) INTERMEDIATE MESH ACCESS POINT (MAP) FAILURE

When the MAP fails, the Controller can detect the saturation through either of the two techniques;

- a) Through Link statistic (Periodic polling)
- b) (Traps)

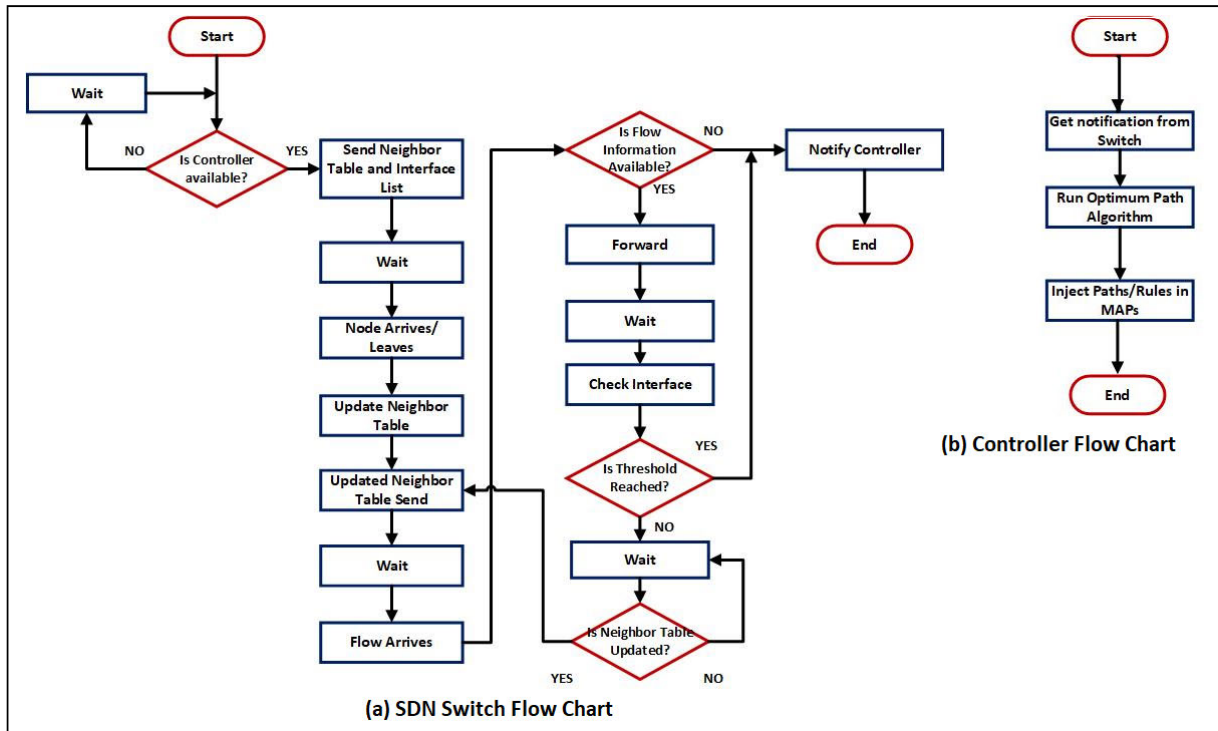


FIGURE 7. SDNMesh flow graph.

The neighbor nodes send the new neighbor table to the Controller upon receiving and association or disassociation request on any of its interfaces.

- Since S_i is defined as the mobility rate for flow f_i and the radio range of 802.11n is defined as $\sim 53\text{m}$. We can expect this event to occur every 37 secs for each node. For a 200-node network, we can expect an event at the controller every 0.18 secs or ~ 5 events per sec.
- This provides an upper bound of 0.18 sec to our run-time for each optimization cycle.

The controller decides which nodes traffic should be redirected to the same Gateway via a new path. The controller uses the Optimization algorithm for the recalculation of a new path. It should be noted that only up-stream nodes can send this information, down-stream nodes cannot do this as downstream nodes do not have a path towards the controller. These nodes have to wait for the periodic scan from the controller to discover an alternate path. It is noted that if no alternate path is found then the controller sends a feedback control message to delay the possible re-initiation of the MAP saturation event.

3) MESH ACCESS POINT (MAP) SATURATED

The resolution in this scenario is different as traffic involved is not in an immediate threat of service unavailability but has the potential of possible degradation in the short-term. The obvious solution is to load balance the traffic to suitable gateways. With our current network parameters, it is unlikely that the MAP will ever get saturated. However, we intend

to study the effect of our future investigations by applying more stringent QoS requirements and enhancing the available bandwidth to each node.

4) CONTROLLER FAILURE

To consider the event of a controller failure we propose to have a backup controller in place which will take up the network control once the primary controller. All Openflow nodes will have the backup controller path configured/installed by the primary controller. The secondary controller will perform a network state sync from the primary controller on periodic bases. Although we are using OpenDaylight's default HA cluster configuration, it is assumed that a traffic load of approximately 40 Kbps is allocated for this information exchange on the path between the two controllers. All nodes of the network must have the capability to install the backup controller path.

5) NODE MOBILITY

When a node becomes mobile in the vicinity of a MAP its Neighbor table experiences the change upon receiving an association/disassociation request. The MAP immediately triggers a neighbor change event bypassing the new updated neighbor table to the controller. The controller then performs a recalculation of the traffic paths using the Optimization algorithm.

V. EXPERIMENTAL EVALUATION

The implementation framework of SDNMesh is using Mininet-WiFi Simulator. We used the OpenDaylight SDN

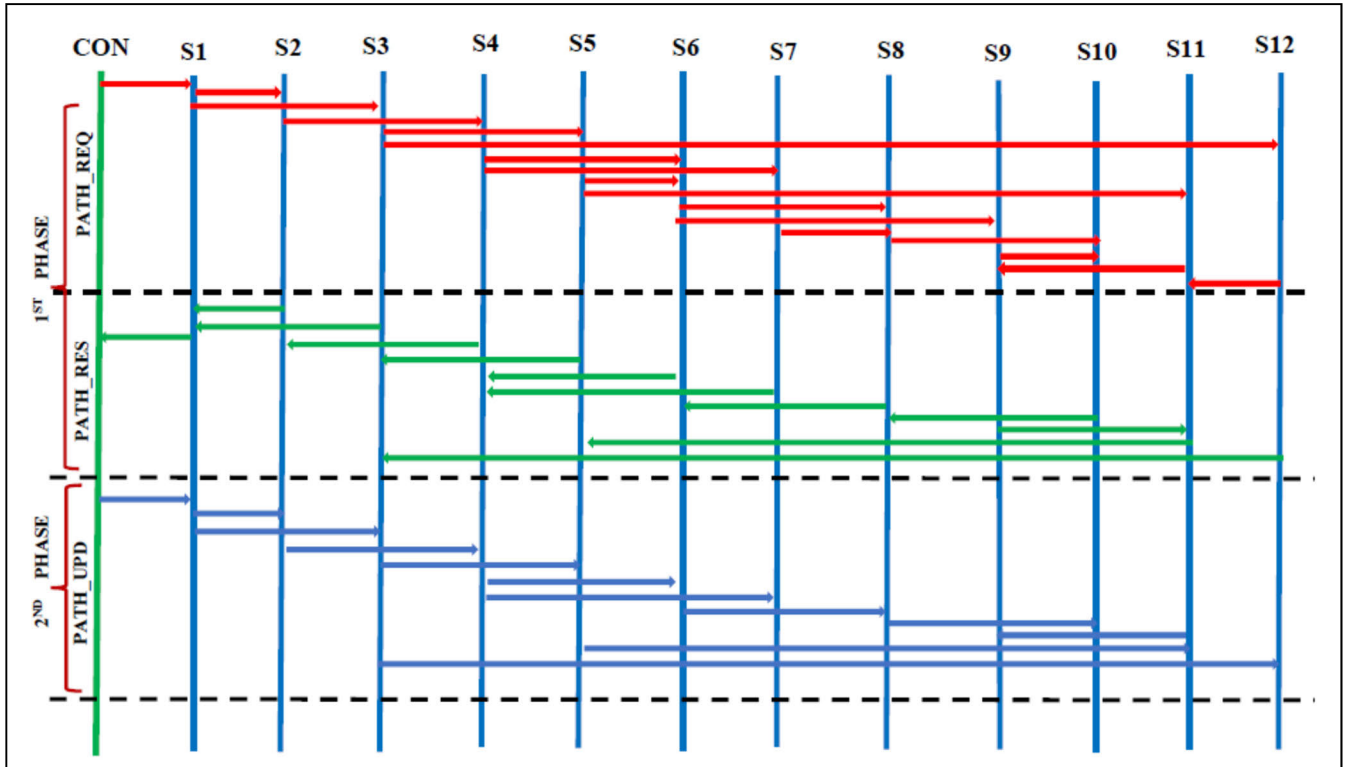


FIGURE 8. SDNMesh (Two-Phase) timeline.

Controller for this purpose and implemented our SDN-Mesh on the ODL controller as a network application. Figure 7 shows the SDNMesh flow graph whereas SDNMesh Timeline for its proposed two-phase strategy has been shown in Figure 8. We have evaluated our approach by comparing its performance with the traditional routing approaches OLSR [37] and BATMAN [38], and SDN based routing Three-Stage [20] using performance metrics Average UDP Throughput (Figure 9), Packet Loss Ratio (Figure 10) and Delay (Figure 11). Experiments are performed for the establishment of connection for (1) controller to switch and (2) switch to switch. Initially, the controller tries to establish a connection with all the switches using PATH_REQ and PATH_RES messages for finding the initial path which is used for the deriving shortest path.

Simulations have been conducted for varying numbers of SDN nodes in the topology, which are 10, 20, 50, 100, and 150 to get the results for three performance metrics that is average UDP throughput, packet drop ratio, and average End-to-End delay. To get better results, several experiments have been carried out that are approximately 10 experiments each for mentioned number of nodes have been performed while considering the Random walk mobility model to investigate support for node mobility in the algorithm. Moreover, to see the variations in results, standard deviations for the desired performance metrics have also been considered.

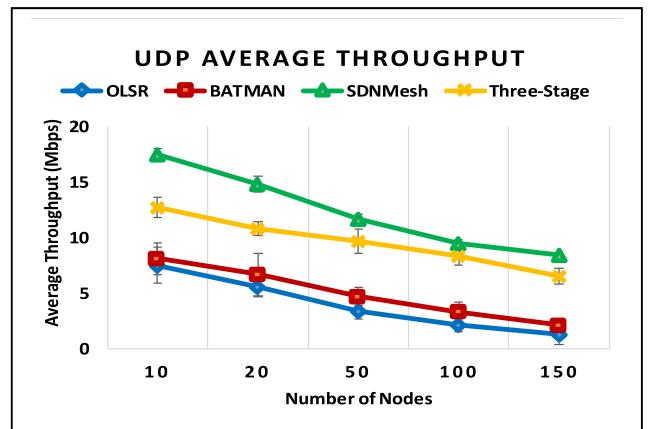


FIGURE 9. UDP average throughput.

Figure 9 shows the average UDP throughput for each protocol based on the number of the network nodes. We note that the SDNMesh routing approach measured throughput is higher compared to the OLSR, BATMAN, and Three-Stage performance. A fine-tuning of the flow distribution such as load balancing between alternative paths can be done using SDN with WMN.

Packet loss is a good metric for reflecting the efficiency of a routing protocol in optimizing internode device exchanges. Data about the dropped packets are obtained due to the unavailability of a routing rule. Figure 10 shows the Packet

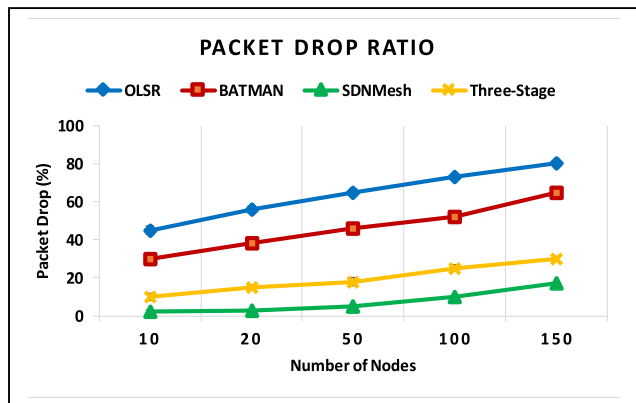


FIGURE 10. Packet loss ratio.

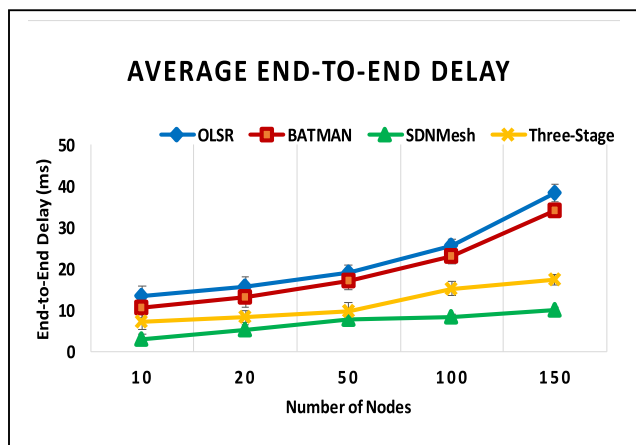


FIGURE 11. Average end-to-end Delay.

Loss Ratio for SDNMesh which is OLSR has the greatest dropped packet rate and therefore the slowest convergence process, therefore BATMAN comes due to a relatively large time interval between updates and the absence of an overhead optimization mechanism. We note it grows with the number of nodes. As a result, two key parameters can have a direct impact on the packet loss rate, which can be the ability to route protocols to optimize paths and the amount of overhead they incur. Nonetheless, interference has, in most situations, a direct impact on the rate of packet loss due to collisions and incorrect packets.

We find the end-to-end delay (Figure 11) to be the time from a packet to be sent from the source mesh client until the destination server which executes the services receives it. We performed this experiment several times and retained the latency average. Measuring one-way delay is not easy as packets experience different delays in the network, including delays in collection, delays in queuing, transmission, and propagation. We have therefore measured the Round Trip Time (RTT), which calculates the one-way latency by assuming half of the RTT. Besides, we measured the delay needed for the controller to send a packet until its router close is received. By using the periodical broadcasting of route request messages, the controller tries to overcome the issue of mobility, node, and link failure (Figure 12). Another method

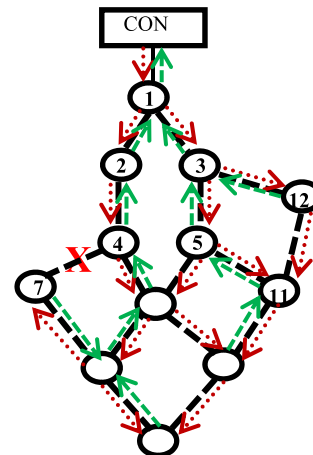


FIGURE 12. Link failure case.

is of triggered path error message ERROR that is generated by the corresponding switch (4 or 7) towards the controller to inform about link or node failure to make controller update routing rules accordingly. This error message is triggered as per the application requirement such as delay critical applications.

To summarize, by analyzing the simulation results we can conclude our analysis. They demonstrate that the centralized approach and the out-of-band signaling of the SDNMesh solution can help WMNs overcome certain limitations the present network is facing due to distributed routing. SDN routing is based on a centralized approach that is the connection between switches is established through the controller. When a switch needs to communicate with another switch, it requests the controller to provide path information towards that specific switch, which results in the installation of required routing rules. To this aspect, controller switch connection plays a significant role in establishing connectivity from one switch to another. However, routing rules do not change or update frequently in wired networks due to the static nature of network nodes, which results in no impact on the controller to switch connectivity. However, connectivity between switches is very much impacted due to frequent movement of nodes in wireless networks more specifically wireless mesh networks. The proposed routing architecture SDNMesh is more advantageous for WMN in terms of connection establishment between switches as compared to routing approach BATMAN, OLSR, and Three-Stage. The later approaches try to find the best route towards controller leading in higher delay in the controller to switch connectivity, causing the unavailability of the controller for a longer period.

VI. CONCLUSION

This paper proposed a two-phase routing strategy SDNMesh for SDN based Wireless Mesh Networks (WMNs), to make mesh access points more efficient and scalable by implementing the concept of programmability and centralized control through Software Defined Networking (SDN), a new

networking technology. SDNMesh is based on various types of messages using OpenFlow protocol, which is then evaluated by the Mininet-WiFi simulation tool compared to its performance with the OLSR, BATMAN, and Three-Stage routing approach using average UDP throughput, average packet drop ratio and end-to-end delay. Experiments are performed for (1) controller to switch connection and (2) switch to switch connection. Initially, the controller tries to establish a connection with all the switches for finding the initial path which is then used for the deriving shortest path from the controller to switch and between the switches.

REFERENCES

- [1] M. Jammal, "Software-defined networking: State of the art and research challenges," *Comput. Netw.* vol. 72, pp. 78–84, Oct. 2014.
- [2] D. Kreuzt, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Dec. 2014.
- [3] F. D. O. Silva, J. H. D. S. Pereira, P. F. Rosa, and S. T. Kofuji, "Enabling future Internet architecture research and experimentation by using software defined networking," in *Proc. Eur. Workshop Softw. Defined Netw.*, Oct. 2012, pp. 73–78.
- [4] A. Drescher, "A survey of software-defined wireless networks," Dept. Comput. Sci. Eng., Washington Univ. St. Louis, St. Louis, MO, USA, Tech. Rep., 2014, pp. 1–15.
- [5] J. F. Gonzalez Orrego and J. P. Urrea Duque, "Throughput and delay evaluation framework integrating SDN and IEEE 802.11s WMN," in *Proc. IEEE 9th Latin-American Conf. Commun. (LATINCOM)*, Nov. 2017, pp. 1–6.
- [6] C. Chaudet and Y. Haddad, "Wireless software defined networks: Challenges and opportunities," in *Proc. IEEE Int. Conf. Microw., Commun., Antennas Electron. Syst. (COMCAS)*, Oct. 2013, pp. 1–5.
- [7] A. Maleki, "An SDN perspective to mitigate the energy consumption of core networks-GEANT2," in *Proc. Int. SEEDS Conf.*, Sep. 2017, pp. 233–244.
- [8] L. Cominardi, C. J. Bernardos, P. Serrano, A. Banchs, and A. D. L. Oliva, "Experimental evaluation of SDN-based service provisioning in mobile networks," *Comput. Standards Interface*, vol. 58, pp. 158–166, May 2018.
- [9] T. Bakhshi, "State of the art and recent research advances in software defined networking," *Wireless Commun. Mobile Comput.*, vol. 2017, Jan. 2017, Art. no. 7191647.
- [10] H. Huang, P. Li, S. Guo, and W. Zhuang, "Software-defined wireless mesh networks: Architecture and traffic orchestration," *IEEE Netw.*, vol. 29, no. 4, pp. 24–30, Jul. 2015.
- [11] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *Proc. 10th Annu. Int. Conf. Mobile Comput. Netw.*, 2004, pp. 114–128.
- [12] I. T. Haque and N. Abu-Ghazaleh, "Wireless software defined networking: A survey and taxonomy," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2713–2737, Oct. 2016.
- [13] M. L. Sichitiu, "Wireless mesh networks: Opportunities and challenges," in *Proc. World Wireless Congr.*, vol. 2, 2005, p. 21.
- [14] M. Eslami, O. Karimi, and T. Khodadadi, "A survey on wireless mesh networks: Architecture, specifications and challenges," in *Proc. IEEE 5th Control Syst. Graduate Res. Colloq.*, Aug. 2014, pp. 219–222.
- [15] S. Y. Shahdad, A. Sabahath, and R. Parveez, "Architecture, issues and challenges of wireless mesh network," in *Proc. Int. Conf. Commun. Signal Process. (ICCCSP)*, Apr. 2016, pp. 557–560.
- [16] K. Liu, Y. Cao, Y. Liu, G. Xie, and C. Wu, "A novel min-cost qos routing algorithm for SDN-based wireless mesh network," in *Proc. 2nd IEEE Int. Conf. Comput. Commun. (ICCC)*, Oct. 2016, pp. 1998–2003.
- [17] 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," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 475–476, Sep. 2013.
- [18] V. Nascimento, M. Moraes, R. Gomes, B. Pinheiro, A. Abelem, V. C. M. Borges, K. V. Cardoso, and E. Cerqueira, "Filling the gap between software defined networking and wireless mesh networks," in *Proc. 10th Int. Conf. Netw. Service Manage. (CNSM) Workshop*, Nov. 2014, pp. 451–454.
- [19] D. B. Rawat and S. Reddy, "Recent advances on software defined wireless networking," in *Proc. SoutheastCon*, Mar. 2016, pp. 1–8.
- [20] P. Patil, A. Hakiri, Y. Barve, and A. Gokhale, "Enabling software-defined networking for wireless mesh networks in smart environments," in *Proc. IEEE 15th Int. Symp. Netw. Comput. Appl. (NCA)*, Oct. 2016, pp. 153–157.
- [21] D. Sajjadi, R. Ruby, M. Tanha, and J. Pan, "Fine-grained access provisioning via joint gateway selection and flow routing on SDN-aware Wi-Fi mesh networks," in *Proc. IEEE 13th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2017, pp. 1–10.
- [22] A. Hakiri, A. Gokhale, and P. Patil, "A software-defined wireless networking for efficient communication in smart cities," Tech. Rep., 2017.
- [23] S. Li, K. Han, N. Ansari, Q. Bao, D. Hu, J. Liu, S. Yu, and Z. Zhu, "Improving SDN scalability with protocol-oblivious source routing: A system-level study," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 1, pp. 275–288, Mar. 2018.
- [24] F. Yaghoubi, "Consistency-aware weather disruption-tolerant routing in SDN-based wireless mesh networks," *IEEE Trans. Netw. Service Manage.* vol. 15, no. 2, pp. 582–595, Dec. 2018.
- [25] K. Bao, J. D. Matyjas, F. Hu, and S. Kumar, "Intelligent software-defined mesh networks with link-failure adaptive traffic balancing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 2, pp. 266–276, Jun. 2018.
- [26] D. Levin, "Panopticon: Reaping the benefits of incremental SDN deployment in enterprise networks," in *Proc. USENIX Annu. Tech. Conf.*, 2014, pp. 333–345.
- [27] S. Vissicchio, L. Vanbever, and O. Bonaventure, "Opportunities and research challenges of hybrid software defined networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 70–75, Apr. 2014.
- [28] Y. Peng, L. Guo, Q. Deng, Z. Ning, and L. Zhang, "A novel hybrid routing forwarding algorithm in SDN enabled wireless mesh networks," in *Proc. IEEE 17th Int. Conf. High Perform.*, Aug. 2015, pp. 1806–1811.
- [29] J. Moy, *OSPF Protocol Analysis*, document 1245, RFC, 1991.
- [30] N. McKeown, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Apr. 2008.
- [31] Y. Guo, Z. Wang, X. Yin, X. Shi, J. Wu, and H. Zhang, "Incremental deployment for traffic engineering in hybrid SDN network," in *Proc. IEEE 34th Int. Perform. Comput. Commun. Conf. (IPCCC)*, Dec. 2015, pp. 1–8.
- [32] M. Labraoui, M. M. Boc, and A. Fladenmuller, "Software defined networking-assisted routing in wireless mesh networks," in *Proc. Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Sep. 2016, pp. 377–382.
- [33] J. Wang, "A software-defined network routing in a wireless multihop network," *J. Netw. Comput. Appl.* vol. 85, pp. 76–83, Dec. 2017.
- [34] Y. Wei, "Energy-aware traffic engineering in hybrid SDN/IP backbone networks," *J. Commun. Netw.* vol. 18, no. 4, pp. 559–566, 2016.
- [35] L. He, X. Zhang, Z. Cheng, and Y. Jiang, "Design and implementation of SDN/IP hybrid space information network prototype," in *Proc. IEEE/CIC Int. Conf. Commun. China*, Jul. 2016, pp. 1–6.
- [36] A. Hakiri, A. Gokhale, and P. Berthou, "Software-defined wireless mesh networking for reliable and real-time smart city cyber physical applications," in *Proc. 27th Int. Conf. Real-Time Netw. Syst.*, 2019, pp. 165–175.
- [37] T. Clausen, P. Jacquet, C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L. Viennot, "Optimized link state routing protocol (OLSR)," Tech. Rep., 2003.
- [38] D. Johnson, N. Ntlatlapa, and C. Aichele, "Simple pragmatic approach to mesh routing using BATMAN," Tech. Rep., 2008.



SYED SHERJEEL A. GILANI received the M.S. degree in computer science from Riphah International University, Islamabad, Pakistan, in 2011. He is currently pursuing the Ph.D. degree in computer engineering with the Capital University of Science and Technology, Islamabad. From 2001 to 2009, he was a Lecturer with the PAF College JC Chaklala, Rawalpindi, Pakistan. From 2009 to 2013, he has worked as a Manager IT with Foundation University, Pakistan. He is also working as a Senior Lecturer with the Faculty of Computing, Riphah International University. His research interests include software-defined networks, wireless mesh networks, the Internet of Things, and routing in wireless networks.



AMIR QAYYUM (Senior Member, IEEE) received the bachelor's degree in electrical engineering from the University of Engineering and Technology, Lahore, in 1991, and the M.S. and Ph.D. degrees from the University of Paris-Sud, France, in 1996 and 2000, respectively. He is currently working as a Professor with the Capital University of Science and Technology (CUST), Islamabad, Pakistan. He completed his research work with INRIA, Rocquencourt, France. He is also the Chair of the IEEE Islamabad Section. He was associated with INRIA Rocquencourt, France, for research on MANETs. He has coauthored *Optimized Link State Routing (OLSR) protocol RFC at the Internet Engineering Task Force (IETF)*. He is also the Head of the Center of Research in Networks and Telecom (CoReNeT). He has led several national and international and funded research projects in the domain of wired and wireless networks. He has several publications in international conferences and journals. His research interests include wireless and mobile computer networks, multimedia networks, software-defined networking, vehicular and mobile ad hoc networks, and next-generation protocols.



RAO NAVEED BIN RAIS (Member, IEEE) received the B.E. degree in computer systems from the National University of Sciences and Technology, Pakistan, in 2002, and the M.S. and Ph.D. degrees from the University of Nice, Sophia Antipolis, France, in 2007 and 2011, respectively. He completed his research work with INRIA, Sophia Antipolis, France. He has also worked with COMSATS University and the Capital University of Science and Technology, Islamabad, Pakistan.

He is currently working as an Associate Professor with Ajman University, Ajman, UAE. He carries industrial research and development, academic, and research experience for several years. He has several publications in renowned international conferences and journals. His research interests include next-generation computer networks, protocol design and development, software-defined networking, information-centric networking, and opportunistic networking.

MUKHTIAR BANO (Member, IEEE) received the M.S. degree in computer engineering from the Center for Advanced Studies and Engineering (CASE), Islamabad, UET Taxila, Pakistan, in 2011. She is currently pursuing the Ph.D. degree in computer engineering with the Capital University of Science and Technology, Islamabad. From 2001 to 2005, she has worked as an IT Officer with Pakistan Air Force, Pakistan. She is also working as an Assistant Professor with the Software Engineering Department, Fatima Jinnah Women University, Rawalpindi, Pakistan. Her research interests include software-defined networks, wireless mesh networks, network monitoring, and routing in wireless networks.

...