International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *12 August 2023* ***Revised:*** *08 October 2023* ***Accepted:*** *24 October 2023*

Impact of Docker Container Virtualization On Wireless Mesh Network by Using Software-Defined Network

Sanjay Shahaji Kadam[1*], Dr. Dayanand R. Ingle[2]
[1]Research Scholar
Bharati Vidyapeeth College of Engineering
Navi Mumbai, India
Email: sanjaykadam23@gmail.com
[2]Professor
Bharati Vidyapeeth College of Engineering
Navi Mumbai, India
Email: dringleus@gmail.com

Abstract: In today's advanced digital world era, it is extremely difficult for small enterprises or organizations to merge traditional or legacy computer network devices/equipment and wireless mesh networking devices with the latest digital computer network technology with respect to the expense of buying and maintaining expensive branded networking devices. However, today, by applying the neatly Software-defined networking, the OpenFlow protocol along with virtualization such as docker containers, which is a pack of their specific libraries, configured files, and software, provides advantages over proprietary or branded computer networking devices with respect to purchasing expenditure, operational expenditure, and improved performance in computer networking. Redistribution of routing protocol is very essential when using various autonomous systems in wireless mesh networks. Docker containers of frr and quagga give an edge over traditional or branded physical router devices, some docker containers are used as wired and wireless hosts/clients in the wireless mesh network. The novel idea used in this paper is on how to use the different software-defined controllers (Ryu and Pox controller) in a docker containerized wireless mesh network to analyse with respect to packet transfer, jitter in transmission, minimum delay in transmission, maximum delay in transmission, the average delay in transmission, delay standard deviation bit-rate, send packets, average packets drop, dropped packets along-with average loss-burst size in Mininet Wi-Fi testbed at the different scenario and the result shows that by using the docker container virtualization along with software-defined network two different controllers improves the performance and optimize the wireless mesh network. In addition, it shows that by using containerization and virtualization, capital expenditure and operational expenditure can be reduced in designing and developing wireless mesh network topologies.

Ethical Compliance: All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki Declaration and its later amendments or comparable ethical standards.

Keywords: SDN, Docker Container, Redistribution, Mininet-Wifi, Openflow, Virtualization, Wireless Network

I. INTRODUCTION

Computer networking combines or communicates various computer units/nodes/clients to share information or resources using various media such as wired networking, wireless mesh networking, wireless networking, and mobile networking devices/equipment. A wireless mesh network (WMN) is a combination of local area networks and wireless devices, commonly based on 802.11 technologies. Wireless mesh networks are considered dynamic and distributed networks. It is very difficult to maintain or configure the legacy networking devices/equipment because of very complicated networking commands, so when it comes situation of updating or integrating vertically internet protocol of traditional wireless mesh computer network of any organization or small enterprise it is very difficult to upgrade or configure existing wireless mesh networking devices because of very tedious configuration, complicated networking commands, incompatible policies which takes more times as well as more cost to upgrade new devices or purchase compatible networking devices. Traditional wireless mesh networking devices are combined control and data planes, where the data plane is nothing but the forwarding plane that manages or collects arriving n/w packets and advances to the designated target created on routing table information, whereas the strategies related to networking are handled by the control panel and task of packets dropping, forwarding, rejecting the data plane, and so on. The use of WMN in various fields such as transportation, university campuses, enterprise networks, colleges, and public safety services is nothing but the next generation of computer networking. There is vast communication between various mesh nodes, mesh routers, and gateways. A WMN offers numerous redundant interconnection routes in the network. Wireless mesh networks are measured with respect to various parameters such as bandwidth, latency, and RTT; therefore, these important parameters are responsible for the selection of routing protocols to achieve the goal of optimization and lowering the operating and capital costs of the

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
***Article Received:** 12 August 2023 **Revised:** 08 October 2023 **Accepted:** 24 October 2023*

wireless mesh network design of organizations or enterprise networks. Wireless mesh networks are organized into three categories: topology, architecture, and protocol.

Software-defined networking is the latest and most promising architecture and is active, vibrant, easily administrable, and less expensive, making it a natural choice for implementation in wireless mesh networking applications. Data packet transmission engineering is a technique for optimizing the execution of a WMN by dynamically evaluating, adapting, and controlling the data transfer over the network. A wireless mesh network (WMN) or infrastructure network creates an entry of source and destination addresses in routing tables and shortest paths, but the shortest path is often congested when the data transmission traffic increases. To overcome this problem, a software-defined network provides a brilliant routing technique using dynamic data-flow tables.

The high availability of different emergency services is based on this goal, which requires a data plane and control plane centralized administrated SDN (Software Defined Network) and NFV (Network Function virtualization), which are used to implement various costly networking devices, which will play a significant role; however, for this, we require proper tuning of these two technologies' complementary features, such as centralized control and virtualization of various devices.

Thus, the need for an hour is to reduce the capital expenditure on wireless mesh networks with a new paradigm, that is, software-defined network and network function virtualisation (dockers and containers), by using these two new advanced technologies [1] [25]. One is SDN with the same central data plane and control plane topologies, SDN with OpenFlow (of) protocol accomplishes network devices more dynamically and flexibly [1], and network function virtualization, which is used to virtualize various expensive networking devices with the help of docker containers with a specific library, configured files, and software for routing in computer networks such as LAN, wireless mesh network (WMN), and metro area network (MAN). SDN provides the central network state. SDN delivers elasticity to control, secure, handle, and enhance WMN using compatible applications [1]. With maximum intersections in the network, various gateways delivering interconnections to nearby networks have become compulsory. SDN knows how to manage these various gateways and align data flows appropriately. Software-defined wireless networks (SDWN) [9] remain the practical phase of utilizing software-defined networking (SDN) ideas in radio receiver(wireless) networks. SDWN attempts to accede to (wired) SDN elasticity; it dissociates radio control and radio data forward roles from each other, to help wireless networks become further adaptable by summarizing the fundamental wireless infrastructure network from applications and services through top-level application programming interfaces.

Docker is a software virtualization platform that supports users in designing, building, utilizing, controlling, and operating applications in containers with dependent libraries on the uppermost part of the structural engine [24]. Here, the engine is the critical component that receives every request to ensure that all containers execute properly with specific applications at the top. Docker containers are lightweight-specific applications. Docker changes the perception of network function virtualization such that it only works in virtual machines, which comes with the operating system, and the virtual network function (VNF) can execute the docker without the support of virtual machines, thus reducing the extra use of important resources such as memory, CPU, and bandwidth. Docker images stored in the cloud can be implemented by users based on their choice. During the early days of docker container development, there were concerns regarding security; however, today, this concern has been resolved, so it is always better to use docker containers instead of virtual machines.

The Docker network container is another feature of modern computer networks. The concept of a docker network is that it can deliver all network application capabilities and deploy them with full functionalities. There are several pluggable networking drivers, some of which are default drivers and cores in the dockers' networking application functionality [21]. The bridge is the default driver with standalone containers, the host is another standalone container, the overlay network driver is used to connect more than one docker, ipvlan network drivers give advantages to implement IPv4 and IPv6 addressing plan, macvlan used to allocate medium access control address to the existing docker container, and none of the network drivers used concurrence with a routine n/w driver, and n/w plugins user implement and use 3rd-party plugins through the docker[21].

Redistribution of routing protocols in topologies with the help of SDN and NFV is essential to design an algorithm, interior gateway, external gateway, and routing protocol at proper locations in topologies using SDN with Docker containers [22] [23]. A wrong selection of routing protocols in topologies can consume more CPU resources and memory along with bandwidth, which is required for the proper functioning of the WMN design. In this paper, we propose a new algorithm for organization and enterprise wireless mesh networks with a focus on bandwidth, latency, throughput, data packet transfer speed, and loss of packets with the help of redistributed routing protocols at topology with respect to hope count, path cost, and convergence time. Some routing protocols use hop count, and some use the path cost in topology.

The theory is that the advantages of SDN attained in wired networks can be realized, at least partly, in Wireless Mesh Networks, resulting in improved performance, better programmability, and decreased complexity via network function virtualization. We employed static nodes as docker containers in this case, and we focused on infrastructure WMNs,

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *12 August 2023* ***Revised:*** *08 October 2023* ***Accepted:*** *24 October 2023*

where mesh nodes may be deemed static. As a result, we may assume a dependable control channel between the SDN controller and the Software Defined Network switches.

Wireless mesh network routers gather data from local mesh nodes and send them to networking gateways or other clients based on the positions of the source and destination client servers. The inner mesh data flow always remains inside the same mesh, whereas beyond the mesh dataflow, it must pass through computer network gateways. Therefore, it shows a simplified and decentralized mesh infrastructure because every client/node should transfer to the extent next client/node, and every client/node transfers the packet from the client/node to the host communicated to itself, but also from the host communicated to every client/node in the existing mesh network. The main aim is to design a wireless mesh network that is self-maintainable, self-troubleshooting, self-monitoring, and that reduces the burden on network administrators.

When there is a limitation of real-time network topology design with physical advanced devices and equipment to test the experiments, it is very costly, so mininet-Wi-Fi emulates the same wireless topology with OpenFlow protocols and SDN, so we can perform high-reliability tests that reproduce the actual network working atmosphere [15]. Mininet-Wi-Fi enhances the mininet emulator with various virtualized Wi-Fi stations and access points by possessing the same software-defined capabilities. Container net is a branch of the well-known Mininet network emulator and permits the use of Docker containers as hosts in emulated mininet-Wi-Fi network topologies. This allows exciting functionalities to shape networking virtualisation and testbeds [17].

II. LITERATURE REVIEW

Wm-SDN [1] discusses the routing challenge in an SDN-based WMN, explained in the first section of this paper, by applying hybrid protocols. The conventional AODV protocol was used for switch-control joining. Later, when the supervisor-switch joining is recognised, OpenFlow protocols are applied to switch over to routing evaluations. In this design, every switch must strengthen the software-defined network OpenFlow and the conventional routing protocols. Therefore, the method is theoretically not a natural SDN-based methodology given that the switch is concerned with routing decisions through AODV. Furthermore, it involves a switch to support complicated hardware and software compared with SDN imagines. In addition, network function virtualisation was at an early stage; therefore, they did not discuss it.

In this study, the authors [2] discussed a new unique architecture to create the SD-WMN idea. The authors concentrated on dealing with the challenges of constructing divided control and data networks via spectrum division and preventing the congestion that arises in the combined medium. Three new unique spectrums and traffic scheduling algorithms–fixed-band non-sharing, non-fixed-band non-sharing, and non-fixed-band sharing–are suggested and assessed by vast simulations. Here, complex topology changes and different interactions may be necessary for the communication of mesh nodes with the wireless mesh network, which is the main cause of the difficulty in operating wireless mesh networks.

In this article, the authors [3] discussed the importance of a new open architecture SDN that permits active and appropriate management of the conduct of network hardware in complicated large computer networks. The authors examined SDN TE solutions from the perspectives of flow administration, load balancing, fault tolerance, topology, and traffic analysis. The latest state and research challenges of SDN TE are discussed by focusing on important SDN functioning metrics in terms of scalability, availability, reliability, consistency, and accuracy. Be there a modern and hopeful architecture, SDN afore-stated routing and controlling challenges of wireless mesh networks by introducing responsiveness and Flexibility.

In this paper, the authors discuss how the OpenFlow protocol is executed in SDN nodes, whereas traditional routing protocols (OLSR, OSPF, and AODV) are executed in traditional/legacy nodes. Panoptic on [4] suggested a mixed SDN architecture that concentrates on solving interconnectivity problems between legacy and SDN nodes. In this design, the network-layer communication problems were not considered. In addition, they did not discuss virtualisation.

The multiple types of hybrid SDN designs suggested by Vissicchio et al. [5] are classified according to topology, class, and service, but they do not include RTT; comparison of OpenFlow with other protocols is also avoided. Limited research has been conducted on this topic because of the lack of prominence in specific execution.

HRFA [6], which is based on SDN forwarding and the OSPF protocol, was designed. This hybrid routing scheme divides the network nodes into traditional nodes and SDN forwarding elements (SDN-FE) and selects efficient forwarding approaches for various network components. This arrangement takes the number of hops and link consumption paths into account to further the data to attain the WMN load balance and decrease the network congestion. The HRFA model performance improved in terms of the typical end-to-end output, median end-to-end delay, and packet failure ratio compared with the conventional routing protocol. HRFA [6] suggests a steady introduction of SDN nodes by steadily expanding the number of nodes, which results in improved data forwarding. They used specific OSPF traditional routing protocols and OpenFlow protocols [7].

OpenFlow [7] is a rational negotiation that allows investigators to execute or conduct tests on various switches and routers in a consistent manner, without the necessity for vendors to reveal the inner mechanisms of their products or for investigators must study vendor-precise control hardware or software.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
***Article Received:** 12 August 2023 **Revised:** 08 October 2023 **Accepted:** 24 October 2023*

Guo et al. [8] proposed a combined SDN and OSPF architecture that concentrates on migrating from traditional to SDN concepts using data transportation engineering as a methodology. This system uses a genetic algorithm to detect repositioning structures.

Labraoui et al. [9] presented a mixed routing architecture with OLSR using SDN to investigate the improved function and administration of legacy routing established by the controller. Consistency is often improved, but it affects the network operating cost, which directly increases the growth in the volume of the network linearly. The proportions of throughput and packet distribution were also improved.

Wang et al. [10] proposed a multi-hop wireless network combined with SDN using hop count as the efficiency value for route selection and suggested a design that addresses QoS routing. To improve the broadcasts, the concept of a multipoint relay (MPR) is employed. The shortest path is defined by the controller and several paths are achieved using the Dijkstra algorithm.

Another hybrid design explained in HEATE [11], addresses the questions of transportation engineering while considering energy efficiency. The design was established using shortest path routing and the OSPF protocol. It proposes the distribution of traffic flow; however, traffic flows accumulated to conserve electricity.

In this paper [12], the authors designed and executed a topology-centred SDN/IP mixture space data network prototype. Satellite and ground networks are two parts of the model. Significant efforts have been made to create satellite networks. In a satellite network, OpenFlow switches as the data plane, their lone capability to forward packets giving to the flow entries mounted by the controller, and OpenFlow controller as the main logic plane, which is accountable for the combination of data and controlling the conduct of the data plane by adjusting the flow entries in the respective switch.

III. RESEARCH METHODOLOGY

The research methodology used in this study employed experimental methods. SDN-based routing in Wireless Mesh Networks is prototyped. This section describes our experiments' major tools, components, platforms, and testbeds. Our prototype uses the SDN controller platform such as Network Operating System, Ryu, POX, FloodLight Trema, and Beacon are among the many SDN controller systems. We prototyped POX and Ryu. Because of their modularity, simplicity of use, support, and quality, they are frequently utilized in research.

The proposed diagram [fig. 1] this research used to check the Impact of Docker container virtualization on wireless mesh networks by using Software Defined Network. In the above literature review section many analyses have been performed on wireless mesh networks and SDN. From this analysis, we conclude that the redistribution of routing protocols is vital for accurate load balancing, routing, topology discovery, data, and control planes in WMN. In the proposed system[Fig.1], the 802.11 standards were used for the WMN. Compared with wired network links, there is a fluctuation between nodes owing to environmental reasons. To overcome these limitations in wireless links, we need a structure with a substantial number of factors to determine a link. In 802.11, the parameters are the service set identifier (SSID), broadcast power, and broadcast frequency. Topology discovery is crucial for efficient action in SDN-dependent applications and network services. In OpenFlow devices, there is no standard topology discovery; however, the OpenFlow discovery protocol is used. OFDP controls the link layer discovery protocol (LLDP), which permits switches in a wireless mesh network (IEEE 802) to advertise their abilities to each other.

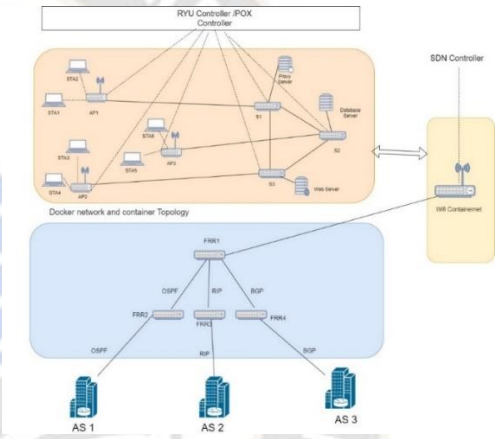

Figure 1. Proposed System Architecture

In Figure 1, the Docker network and container topology are particularly important in the proposed system architecture. Here wireless mesh topology is used with two controllers, in this docker network and container topology, the FRR routing [19] container is used for routing purposes, which redistributes the OSPF protocols from different legacy/traditional network protocols. As mentioned earlier, frr is the fork of Quagga routing software. The three autonomous systems, AS1, AS2, and AS3, are based on a legacy computer network. Three different protocols and three different topologies are used here, which use Docker containers, use a docker bridge network with an FRR container, which is a fork of the quagga. When data from different protocols (RIP, OSPF, and BGP) come in three different frr (frr2, frr3, frr4) docker containers, packets are forwarded to the next frr container frr1, which redistributes bgp and rip into the ospf protocols. It then transfers packets to the container net (mininet-Wi-Fi docker, controlled by the SDN controller). This is followed by communication between pox-controlled Wi-Fi networks [15] [17]. Therefore, the full wireless mesh network communicates with the outside world and intrabuilding servers, such as a database server, web server, and storage server, as well as communication between wireless

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *12 August 2023* ***Revised:*** *08 October 2023* ***Accepted:*** *24 October 2023*

clients. Therefore, Pox is a controller that functions as a control plane. Switches S1, S2, and S3 function as programmable switches, and they are nothing but the data plane in a software-defined network. As mentioned earlier the NFV (network function virtualization) is used in the proposed network. Here, container virtualised equipment came to be released and created many benefits for the network function virtualisation infrastructure. Ryu [18] and Pox [19] controllers were used [21].

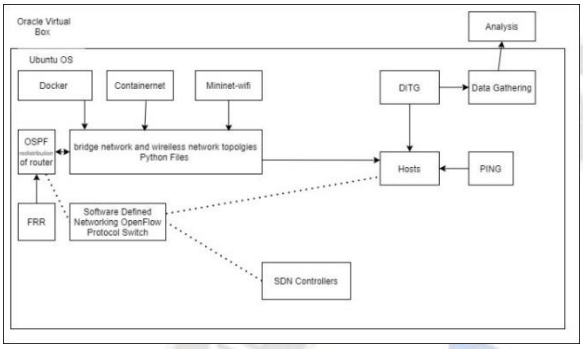

Figure 2. Mininet Wi-Fi and Containernet Environment

Figure 2 illustrates the environment. Software-defined network wireless networks(mininet-Wi-Fi) and containernet mixed with traditional networks also bridge the network of a docker container with different routing protocols used in different autonomous systems to analyze the effect of redistribution of routing protocols. For effects on wireless mesh networks with outside Autonomous systems of various (OSPF, RIP, BGP) redistributed routing protocols, distinct sizes of packets with a contrasting number were generated and transmitted from various hosts of wireless mesh networks to docker container of networks, Docker containernet hosts and latency and overall packet transmission were examined. PING commands have been used to observe the communication between different hosts. Matching, integrating, and improving the functioning of virtual devices, such as routers, switches, and bridge networks of dockers, with traditional network devices and topologies for two or more domain routing resolves were clearly described by the numerical values and evaluation methods required for software-defined networks and traditional networks. Mininet(Containernet) and Mininet-Wifi emulated their hosts as Linux machines. The Pox controller is used as the SDN controller on the same machine [21].

The overall Pseudocode algorithm for the above architecture is given below:

```
SETUP:
  Initialize POX and Ryu software-defined controllers
  Create a Docker container for each application/service to be deployed
  Create three FRR Docker containers (frr2, frr3, frr4)
  Start Mininet-Wi-Fi with a virtual wireless network topology
  Create a Docker bridge network with the FRR container frr1

CONFIGURE:
  Configure controllers to handle traffic flow to/from Docker containers
  Configure controllers for wireless network management (e.g., SSID)
  Connect controllers to Mininet-Wi-Fi environment
  Configure FRR Docker containers for routing protocols (RIP, OSPF, BGP)
  Connect the three different FRR containers (frr2, frr3, frr4) to the Docker bridge network
  Connect frr1 to the Docker bridge network as well

DEPLOY:
  Start Docker containers and deploy applications/services
  Launch Mininet-Wi-Fi CLI for network configuration and testing
  Start the FRR containers and configure routing protocols

MONITOR:
  Monitor network and Docker containers for proper communication and flow of traffic
  Implement load balancing and failover mechanisms
  Continuously update controllers and Docker containers with security patches and software updates
  Monitor wireless-specific metrics (e.g., packet transfer, jitter in transmission, minimum delay in
transmission, maximum delay in transmission, the average delay in transmission, delay standard
deviation bit-rate, send packets, average packets drop, dropped packets along-with average loss-
burst size, throughput)
  Monitor routing protocols' performance and updates

REDISTRIBUTION:
  When data from different protocols (RIP, OSPF, BGP) come in three different FRR containers (frr2,
frr3, frr4)
  Packets are forwarded to the next FRR container frr1
  frr1 redistributes BGP and RIP into the OSPF protocols
  Continuously monitor the redistribution process to ensure proper functioning

RESOURCE_MONITORING:
  Monitor resource utilization of Docker containers, controllers, Mininet-WiFi, and FRR containers
  Take action if necessary to address issues with resource utilization, network flow or security

TROUBLESHOOT:
  Monitor and troubleshoot any issues that arise
  Adjust algorithm as needed
```

Using D-ITG, three UDP flows were created with varying packet sizes, that is, 256, 512, and 1024, and packet rates for 20 s. More log files are created for the source and destination nodes. Here, the destination node is a docker container host that is connected to outside mininet-Wi-Fi cli and from inside mininet-Wi-Fi; there are five Wi-Fi hosts, one Wi-Fi host with a docker container, one wired host, and three access points as well as two open-flow virtual switches: at the destination node(docker container wired node) log file, which contains the values for various running pointers such as bytes received, jitter in transmission, minimum delay in transmission, maximum delay in transmission, the average delay in transmission, delay standard deviation bit-rate, send packets, average packet drops, and dropped packets along-with average loss-burst size. When interpreting the log file, we obtained a flow of data comprising effective statistics. The parameter values were obtained from the log file on the destination nodes. Various one-way flows were sent for 20 s among a couple of clients by adjusting the communication rate as well as the size of the packet to examine the functioning of the network in minimum and maximum loads.

IV. DATA ANALYSIS

In this section, an data analysis of the experimental setup (mininet-wifi, containernet, and docker redistribution routing) has been performed with three cases: the first case, where only two pox controllers are used with a docker and at the wireless hosts and wireless docker container; in the second case, one ryu controller is used at the docker side and one pox controller is used at the wireless hosts and wireless docker container; and the third case, where two ryu controllers are used in both the docker

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
***Article Received:** 12 August 2023 **Revised:** 08 October 2023 **Accepted:** 24 October 2023*

container side and at the wireless hosts and wireless docker container. All the wireless hosts were located in fixed locations. Ubuntu: A trusting container is used for both containers. D-ITG was used to generate log files. The packet sizes were 256, 512, and 1024, with constant transmission rates of 50, 25, and 12 per second, respectively. All three flows are delivered in the traffic flow via the UDP to a docker container host d1, which acts as a server, and the other nodes act as hosts. Finally, the log files of flows produced from five Wi-Fi hosts and one wired host in the emulator and software-defined networks were treated in the form of CSV files using a Python script. Then, all data from the CSV file were converted and stored in a table format [Table 1, 2, 3] for proper comparison and analysis. Here, each graph represents five Wi-Fi hosts (sta1, sta2, sta3, sta4, and sta5) and one wired host(h1) with two pox controllers, five Wi-Fi hosts (sta1-r, sta2-r, sta3-r, sta4-r, and sta5-r) and one wired host(h1-r) with one ryu controller and one pox controller, five Wi-Fi hosts (sta1-2r, sta2-2r, sta3-2r, sta4-2r, and sta5-2r), and one different wired host(h1-2r) with two pox controllers. The positions of access points ap1 are x-axis 10 mtr y-axis 10 mtr and z-axis 0 mtr with ssid 1, and the positions of the ap2 x-axis are 30 y-axis 10 mtr and z-axis 0 mtr and ssid 2. The positions of the ap3 x-axis are 40 mtr, the y-axis is 10 mtr, and the z-axis is 0 mtr and ssid 2. Here the wifi hosts sta1 x-axis 1 mtr y-axis 1 mtr and z-axis 0 mtr, Wifi host sta2 x-axis 31 mtr y-axis 11 mtr and z-axis 0 mtr, sta3 x-axis 36 mtr y-axis 18 mtr and z axis 0 mtr, sta4 x-axis 38 mtr y-axis 15 mtr and z axis 0 mtr and final docker container wifi host position is x-axis 41 y axis 18 and the z-axis is 0. sta1 and sta3 are connected to ap1, and sta2 and sta4 are connected to ap2. The last docker container sta5 is connected to ap3.

V. RESULTS

The following data were collected from the CSV data file and converted into a table format for further analysis. Here table 1 shows the result when two POX controllers are used, here at the wifi-docker sta host there is no activity, it shows null values means the pox controller is not useful in the docker wifi host.

TABLE I. TWO POX CONTROLLER WIRELESS MESH TOPOLOGY DATA

Flow 1	Wifi Sta1	Wifi Sta2	Wifi Sta3	Wifi Sta4	Wifi-Docker Sta	Docker host	Unit
flows sent	3	3	3	3	3	3	
Total Time	19.953264	19.882242	19.938975	19.917144		19.995665	S
packets	1603	1611	1607	1587		1609	
Min. delay	0.000958	0.001002	0.001031	0.001057		-0.000001	S.
Max. delay	0.087881	0.143643	0.109433	0.164055		0.035435	S.
Avg. delay	0.004669	0.005035	0.005148	0.005731		0.000758	S.
Avg. jitter	0.002833	0.003174	0.002979	0.003746		0.000519	S.
Delay std. deviation	0.00555	0.009333	0.00724	0.010709		0.001598	S.
Bytes received	711680	715008	712960	706816		713984	
Avg. bitrate	285.33878	287.69713	286.05683	283.90255		285.65552	Kbit/s
Avg. packet rate	80.337733	81.027079	80.595918	79.680099		80.467441	pkt/s
Packets dropped	0	0	0	0		0	0.000%
Avg. loss-burst size	0	0	0	0		0	pkt
Error lines	0	0	0	0		0	

In Table 2, two different controllers are used at different places, where one pox controller is used at wired and Wi-Fi hosts and the ryu controller at the Wi-Fi docker host, suddenly at pox-controlled Wi-Fi sta1, shows a drastic change in data and almost sends double packets.

TABLE II. ONE POX CONTROLLER AND TWO POX CONTROLLER WIRELESS MESH TOPOLOGY DATA

Flow 1	Wifi Sta1	Wifi Sta2	Wifi Sta3	Wifi Sta4	Wifi-Docker Sta	Docker host	Unit
flows sent	3	3	3	3	2	3	
Total Time	20.094269	13.584241	19.927497	14.298056	13.605275	19.99405	S
packets	2642	1243	1192	1162	984	1311	
Min. delay	0.018256	0.001057	0.000997	0.001224	0.001097	-0.000011	S.
Max. delay	0.534928	13.874746	0.880092	15.570375	14.445648	0.114413	S.
Avg. delay	0.071555	5.740336	0.013066	10.236923	3.870392	0.001707	S.
Avg. jitter	0.015255	3.130346	0.009952	1.722279	1.693706	0.002247	S.
Delay std. deviation	0.048788	5.731129	0.044844	6.513703	5.448142	0.00508	S.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *12 August 2023* ***Revised:*** *08 October 2023* ***Accepted:*** *24 October 2023*

Bytes received	1205248	563712	541696	521216	341248	586496	
Avg. bitrate	479.83751	331.979976	217.46675	291.629016	200.65629	234.668214	Kbit/s
Avg. packet rate	131.480274	91.503088	59.816845	81.269789	72.324889	65.569507	pkt/s
Packets dropped	0	0	0	208	62	0	0.00%
Avg. loss-burst size	0	0	0	0	0	0	pkt
Error lines	0	0	0	0	0	0	

In Table 3, the two Ryu controllers show approximately constant data flow across all the Wi-Fi hosts. However, when used with a 2 pox controller [Table 1], the docker container host does not send packets.

TABLE 3. TWO RYU CONTROLLER WIRELESS MESH TOPOLOGY DATA

Flow 1	Wifi Sta1	Wifi Sta2	Wifi Sta3	Wifi Sta4	Wifi-Docker Sta	Docker host	Unit
flows sent	3	3	3	3	3	3	
Total Time	19.979401	19.986738	20.021434	19.997228	20.002581	19.998082	S
packets	1591	1589	1643	1597	1430	1594	
Min. delay	0.001104	0.001029	0.000973	0.000946	0.000913	0.000054	S.
Max. delay	0.072423	0.046312	0.027251	0.055663	0.066211	0.012098	S.
Avg. delay	0.004714	0.004854	0.003368	0.004844	0.009659	0.000719	S.
Avg. jitter	0.002876	0.003031	0.002043	0.003235	0.009269	0.000044	S.
Delay std. deviation	0.00416	0.003705	0.002716	0.004052	0.011016	0.000636	S.
Bytes received	707072	706816	725760	708608	271235 2	708352	
Avg. bitrate	283.12 04	282.914	289.993 214	283.4 82491	1084.8 00806	283. 367975	Kbit/s
Avg. packet rate	79.632017	79.502718	82.062054	79.861069	71.490774	79.707644	pkt/s
Packets dropped	0	0	0	0	31	0	0.000%
Avg. loss-burst size	0	0	0	0	0	0	pkt
Error lines	0	0	0	0	0	0	

The following are the comparison graphs of all three scenarios: two Pox controllers, one Pox one Ryu controller, and two Ryu controllers. Fig. 3 is the graph of the total number of Packets sent across topology, In Fig. 4 graph of the average delay shown in seconds of all three situations. Fig. 5 and Fig. 6 graphs show respectively Maximum and minimum delay in Seconds. Fig 7 shows the graph of the total average jitter of each host in topology in all three situations, Fig. 8. shows the graph of delay in standard deviation in seconds, and Fig. 9. graphs show the bytes received at various hosts. Fig. 10. graph shows the average bitrate, Fig. 11. Shows the average packet rate with unit Kilo Bytes per second and Fig. 12. graph shows the total packets dropped in all three situations.

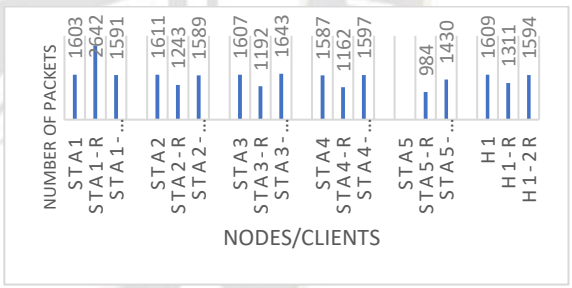

Figure 3. Total Number Of Packets

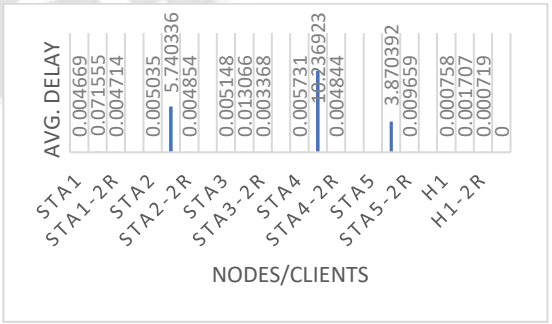

Figure 4. Average Delay in Seconds

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 10
Article Received: *12 August 2023* ***Revised:*** *08 October 2023* ***Accepted:*** *24 October 2023*

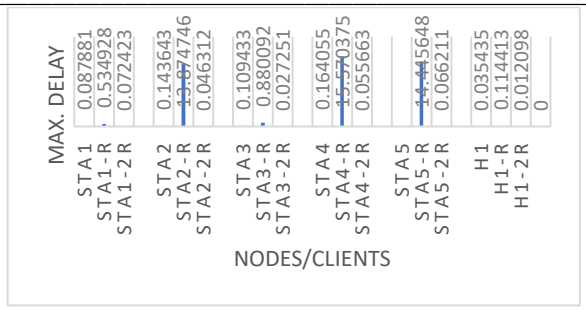

Figure 5. Maximum Delay In Seconds

Figure 6. Minimum Delay In Seconds

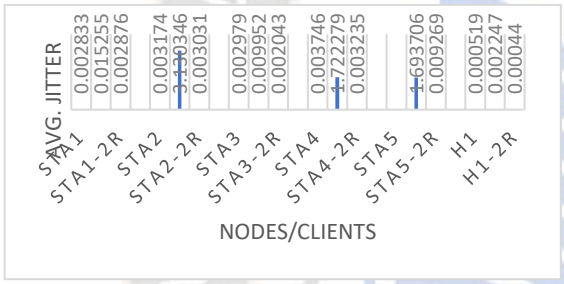

Figure 7. Average Jitter

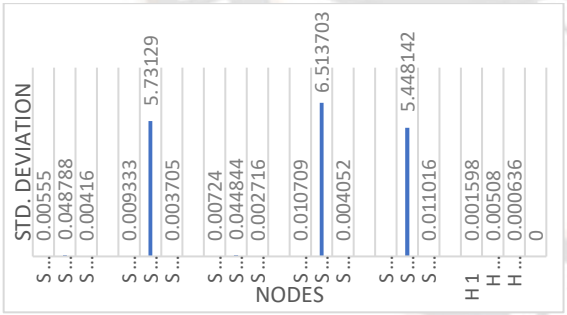

Figure 8. Delay Standard Deviation in Seconds

Figure 9. Bytes Received

Figure 10. Average Bitrate

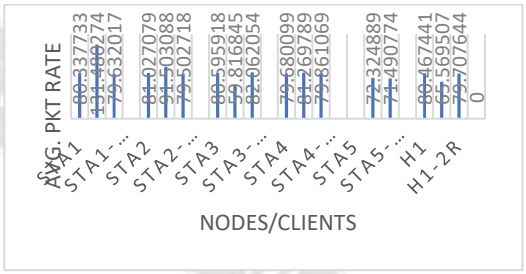

Figure 11. Average Packet Rate In KB/S

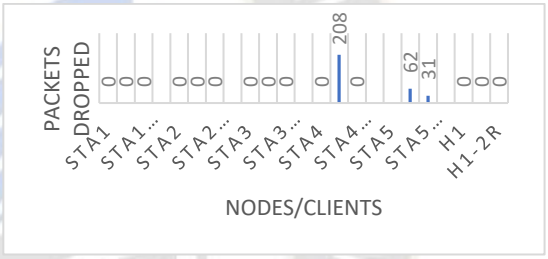

Figure 12. Packets Dropped

Figure 1 shows four Wi-Fi hosts (sta1, sta2, sta3, and sta4), one docker container Wi-Fi host(sta5), and one wired host(h1) with various parameters that are used to analyse the performance of the topology. Here, three OpenFlow packets are sent from the aforementioned host to the docker container host(d1), which has a redistribution topology connected outside the mininet-Wi-Fi network.

In figure 3, the Wi-Fi station (sta1) shows a significantly improved average number of three flow packets when one Pox controller and one Ryu controller are used in a mesh topology. However, at the workstation (sta5), which is a docker container, the Wi-Fi node has a problem when the pox controller is used instead of one pox one Ryu controller that time it shows no packets; however, when there are two Ryu controllers, it sends 1430 packets [Fig 3]. It is less compared to other Wi-Fi hosts and wired host Ryu controller-controlled topology, but almost the same average packets are sent across all hosts, including wired, except for the docker container Wi-Fi node, which has slightly fewer packets.

So, it shows that when there is a docker container host near a Wi-Fi access point and the pox controller manages the access point and switch properly parallel to two Ryu controller which

manages other switches improves the packet transmission drastically also average packet transfer rate is almost 80% more comparing with other two scenarios[Fig.3] which are two pox controller or two Ryu controller. The jitter was 15.15 milliseconds [Fig. 7], which is very high compared to the other two scenarios. However, anything less than 29 ms is acceptable for audio-video transmission. The number of dropped packets was also very low and the number of bytes received was higher. When the 1 pox and 1 Ryu controllers were almost 80% higher.

If we consider a docker-container Wi-Fi host sta5. This shows that when two pox controllers are used in parallel, there are no packets transmitted or received, and 100% of packets drop when they are used in one pox and one Ryu controller parallel. It shows 50% fewer packets than the two Ryu controllers [Fig. 3].

Therefore, it is always better to use the Ryu controller only for the docker-container Wi-Fi host.

In sta2, sta3, and sta4, Wi-Fi hosts to one pox and one Ryu controller result with respect to various parameters such as the number of packets sent average delay in seconds, maximum [Fig. 5] and minimum delay [Fig. 6], jitter average [Fig. 7], standard deviation [Fig. 8], bytes received [Fig. 9], bit rate [Fig. 10], the average rate of packets (Kilobytes / s) [Fig. 11], and the last packet drop [Fig. 12] is shown. Some parameters are approximately equal, and there is a slight difference.

VI. CONCLUSION

From all the above figures it shows that when the access point and open flow switch are connected to the pox controller and if the host is in the range of the access point the performance is improved because the other part of topology is handled by the Ryu controller, so it is always useful in video and audio streaming use all the Wi-Fi host connected to pox controller and other mesh hosts (Wired, docker container Wi-Fi host, etc) to Ryu controller is useful. It also shows that when the docker container Wi-Fi host is connected to the only pox controller, it does not work and all packets drop, but if you connect the topology that uses all docker container Wi-Fi hosts, the time-ryu controller always performs well. Frr is always useful for the redistribution of routing protocols, and if it is used in a docker container, it not only performs very well but also reduces the expenditure of network topology design.

REFERENCES

[1] A. Detti, C. Pisa, S. Salsano and N. Blefari-Melazzi, "Wireless Mesh Software Defined Networks (wmSDN)," 2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2013, pp. 89-95, doi: 10.1109/WiMOB.2013.6673345.

[2] H. Huang, P. Li, S. Guo and W. Zhuang, "Software-defined wireless mesh networks: architecture and traffic orchestration," in IEEE Network, vol. 29, no. 4, pp. 24-30, July-August 2015, doi: 10.1109/MNET.2015.7166187.

[3] I. F. Akyildiz, A. Lee, P. Wang, M. Luo and W. Chou, "Research challenges for traffic engineering in software defined networks," in IEEE Network, vol. 30, no. 3, pp. 52-58, May-June 2016, doi: 10.1109/MNET.2016.7474344.

[4] Dan Levin, Marco Canini, Stefan Schmid, Fabian Schaffert, & Anja Feldmann, "Panopticon: Reaping the Benefits of Incremental SDN Deployment in Enterprise Networks," In 2014 USENIX Annual Technical Conference (USENIX ATC 14). USENIX Association, 2014, pp. 333–345.

[5] Stefano Vissicchio, Laurent Vanbever, and Olivier Bonaventure, "Opportunities and research challenges of hybrid software defined networks," SIGCOMM Comput. Commun. Rev., vol. 44, no. 2, pp. 70–75, April 2014. https://doi.org/10.1145/2602204.2602216

[6] Y. Peng, L. Guo, Q. Deng, Z. Ning and L. Zhang, "A Novel Hybrid Routing Forwarding Algorithm in SDN Enabled Wireless Mesh Networks," 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, 2015, pp. 1806-1811, doi: 10.1109/HPCC-CSS-ICESS.2015.271.

[7] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner, "OpenFlow: enabling innovation in campus networks," SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, April 2008. https://doi.org/10.1145/1355734.1355746

[8] Y. Guo, Z. Wang, X. Yin, X. Shi, J. Wu and H. Zhang, "Incremental deployment for traffic engineering in hybrid SDN network," 2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC), 2015, pp. 1-8, doi: 10.1109/PCCC.2015.7410320.

[9] M. Labraoui, M. M. Boc and A. Fladenmuller, "Software Defined Networking-assisted routing in wireless mesh networks," 2016 International Wireless Communications and Mobile Computing Conference (IWCMC), 2016, pp. 377-382, doi: 10.1109/IWCMC.2016.7577087.

[10] Junfeng Wang, Yiming Miao, Ping Zhou, M. Shamim Hossain and Sk Md Mizanur Rahman, "A Software Defined Network Routing in Wireless Multihop Network," Journal of Network and Computer Applications, http://dx.doi.org/10.1016/j.jnca.2016.12.007

[11] Ajith Prasad, Abhishek, "Cross-Layer Design for optimizing Network using SDN along with NFV and Big Data Application in Docker Container," 2016. 10.13140/RG.2.2.10142.13121.

[12] S. S. A. Gilani, A. Qayyum, R. N. B. Rais and M. Bano, "SDNMesh: An SDN Based Routing Architecture for Wireless Mesh Networks," in IEEE Access, vol. 8, pp. 136769-136781, 2020, doi: 10.1109/ACCESS.2020.3011651.

[13] Michael Rademacher, Karl Jonas, Florian Siebertz, Adam Rzyska, Moritz Schlebusch, Markus Kessel, "Software-Defined Wireless Mesh Networking: Current Status and Challenges," The Computer Journal, vol. 60, no. 10, pp.

1520–1535, October 2017. https://doi.org/10.1093/comjnl/bxx066

[14] Ramon dos Reis Fontes, Mohamed Mahfoudi, Walid Dabbous, Thierry Turletti, Christian Rothenberg, "How far can we go? Towards Realistic Software-Defined Wireless Networking Experiments," The Computer Journal, Oxford University Press (UK), vol. 60, no. 10, pp.13, 2017. hal-01480973

[15] R. R. Fontes, S. Afzal, S. H. B. Brito, M. A. S. Santos and C. E. Rothenberg, "Mininet-WiFi: Emulating software-defined wireless networks," 2015 11th International Conference on Network and Service Management (CNSM), 2015, pp. 384-389, doi: 10.1109/CNSM.2015.7367387.

[16] S. Avallone, S. Guadagno, D. Emma, A. Pescape and G. Ventre, "D-ITG distributed Internet traffic generator," First International Conference on the Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings., 2004, pp. 316-317, doi: 10.1109/QEST.2004.1348045.

[17] M. Peuster, H. Karl, and S. v. Rossem, "MeDICINE: Rapid Prototyping of Production-Ready Network Services in Multi-PoP Environments," IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Palo Alto, CA, USA, pp. 148-153, 2016. doi: 10.1109/NFV-SDN.2016.7919490.

[18] OpenState-SDN. "GitHub - OpenState-SDN/Ryu: OpenState Controller Implementation Based on Osrg/Ryu." GitHub, github.com, 17 Feb. 2016, https://github.com/OpenState-SDN/ryu.

[19] "Docker Hub." Docker Hub, hub.docker.com, https://hub.docker.com/r/frrouting/frr

[20] A. Rastogi and A. Bais, "Comparative analysis of software defined networking (SDN) controllers — In terms of traffic handling capabilities," 2016 19th International Multi-Topic Conference (INMIC), 2016, pp. 1-6, doi: 10.1109/INMIC.2016.7840116.

[21] L. L. Mentz, W. J. Loch and G. P. Koslovski, "Comparative experimental analysis of Docker container networking drivers," 2020 IEEE 9th International Conference on Cloud Networking (CloudNet), 2020, pp. 1-7, doi: 10.1109/CloudNet51028.2020.9335811.

[22] S.S. Kadam, D.R. Ingle, "Literature Review on Redistribution of Routing Protocols in Wireless Networks Using SDN Along with NFV". In: Ranganathan, G., Fernando, X., Shi, F., El Allioui, Y. (eds) Soft Computing for Security Applications. Advances in Intelligent Systems and Computing, vol 1397, 2022. Springer, Singapore. https://doi.org/10.1007/978-981-16-5301-8_41

[23] "Redistributing Routing Protocols - Cisco." Cisco, www.cisco.com, 22 Mar. 2012, https://www.cisco.com/c/en/us/support/docs/ip/enhanced-interior-gateway-routing-protocol-eigrp/8606-redist.html.

[24] "Docker Documentation." Docker Documentation, docs.docker.com, 2 Aug. 2022, https://docs.docker.com/.

[25] Rademacher, Michael & Jonas, Karl & Siebertz, Florian & Rzyska, Adam & Schlebusch, Moritz & Kessel, Markus, "Software-Defined Wireless Mesh Networking: Current Status and Challenges", Comput. J., vol. 60, pp. 1-16, 2017. 10.1093/comjnl/bxx066.