www.ijecs.in
International Journal Of Engineering And Computer Science
Volume 14 Issue 09 September 2025, Page No. 27735-27741
ISSN: 2319-7242 DOI: 10.18535/ijecs/v14i09.5235

Designing WebRTC Video Communication Based SDN Approach for Low latency

Sura Saad Basher
Northern Technical University

Abstract
To offer Quality of Service (QoS) for real-time collaborative video applications like video conferencing and distance learning, the network must select the optimal path among several choices. There might be other network paths that link the source and the endpoint, but it is difficult to avoid low latency by taking a different path due to the network's tight coupling and complex architecture. As long as network topologies such as Integrated Services (ISs) install the path selected by the routing protocol, it may not offer optimal performance. The main aim of this paper is to build and implement interactive video in real-time and find out the QoS using Python language and Web Real-Time Communication (WebRTC) technology with Software-Defined Networking (SDN) for selecting network architecture to obtain the best route based on a network-wide perspective. Besides, it has demonstrated the outcomes of the best route and assesses the performance. Not only that but also, this work presents a new technique using WebRTC with SND to obtain better paths with low latency.

Keywords: Web Real-Time Communication (WebRTC); Software-Defined Networking (SDN); Metrics (Network delay, and Low latency.

1. Introduction

In May 2011, the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C) developed Web Real-Time Communication (WebRTC) as a contemporary technology. Google at the time released a set of open-source, JavaScript APIs and standards for creating peer-to-peer real-time browser interactions [1]. As a result, JavaScript API is directly used in WebRTC to provide interactive interactions via a variety of data formats, provided that it provides several benefits, including the ability to use the API without the need for extensions, licensing, registration, installation, or payment. Furthermore, it includes noteworthy options for video and audio calling features. Additionally, WebRTC technology makes use of HTML5-based web browsers managed by Firefox, Chrome, etc. This standard gives developers an advantage over competing technologies by enabling them to see the browser as a full video chat terminal. On the other hand, researchers have suggested ways to address the drawbacks of conventional networks, such as Software-Defined Networking (SDN), which usually consists of two sets of controllers: SDN controllers for a data centre's Network Function Virtualization (NFV) and controllers for a network's programmable switches [2]. Therefore, both SDN and NFV offer the ability for the network design and infrastructure to be abstracted in software and then implemented across different hardware and devices. This abstraction in software enables the two technologies to cope with the failings of traditional networks. While, SDN allows the configuration and running of the network to be defined and modified via programming, thus adding programmability to the network. An important part of designing and managing a network includes the process of effectively distributing traffic among network

devices, i.e., load balancing [3]. Consequently, an effective load balancer optimizes network parameters such as latency, resource utilization, throughput, and fault tolerance with minimal power consumption. For example, in traditional networks, load balancing is usually carried out by a dedicated server to cope with the ever-changing network demands, and dynamical load-balancing servers have been implemented [4]. Furthermore, for load balancing in VFNs, it has been referred the reader to and references within.

As a result, SDN provides the option to abstract network architecture and infrastructure in software, which can subsequently be applied to various hardware and devices. However, SDN gives the network more programmability by enabling programming to define and alter the way it operates and is configured. Load balancing, or the process of efficiently dividing flow among network devices, is a crucial component of network design and management. For the networks to manage the extra traffic and keep their high speeds, load balancing and multipathing are required[17].

The networks have to load balance and multipath to handle the increased traffic while sustaining high speeds. This research has concentrated on the evaluation and examination of various load-balancing approaches to SDNs. Therefore, classic Ethernet switches have several drawbacks because of how difficult it is to install and maintain them. Manufacturers of conventional network devices typically use standardization protocols created by organizations such as WebRTC on the control plane of their devices. This work is structured as follows: part II contains the methodology, implementation, and evaluation. The result and subsequent research III.

2. Methodology

The fact that integrated services require a lot of communications and states to maintain a single flow is one of their main issues [5]. This restricts the maximum size of the reservation control protocol-using router. The inability of integrated services to select a different path across the network compared to the one selected by the protocol for routing is another issue [6]. End-to-end QoS is limited because the network cannot try out different paths because the routing protocol along with the path selection method are so tightly linked [7]. To explain the process, it has been discussed the network topology consisting of 8 nodes (switches) and 2 Hosts (sender and receiver).

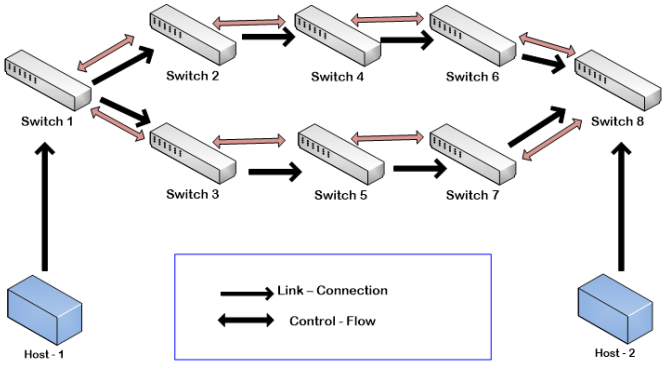
Since increasing network throughput and increasing fault tolerance are the primary goals of data networks, the Network topology is utilized in the SDN OpenFlow protocol [8]. Core, edge and aggregation switches, make up the three-tiered switch architecture that results from this particular topology, which has all of the switches connected to and communicating to one another. The graphical representation of network topology is shown in Figure (1).

Figure (1), demonstrates a 13-node topology using the node's routing protocol. Additionally, the data cannot be mounted from the sender to the recipient by the routing protocol in the most effective manner. Let's assume that the path (switch1–switch2–switch4–switch6–switch8) is the most efficient way to transmit video data from the sender of the message (Host 1) to the recipient (Host 2). The shortest path, switch1–switch3–switch6–switch7–switch8, is the preferred path if the routers employ Open Shortest Path First (OSPF). This indicates that the path taken by video data consists of five hops. To set up QoS on the path (switch1–switch3– switch6- switch7 – switch8), the source sends a path message to switch 1, which then sends it to switch 8. When switching 8 the path message, it sent it on to the receiver.

After verifying that the traffic specifications in the PATH message are accurate, the receiver adjusts its soft state to match the traffic specification and replies with the reservation request message. After receiving the reservation request message through the receiver, switch8 sends it to switch1 in a soft state by the limitation request traffic standard.

Switch1 transmits the reservation demand message back to the sender after setting its soft indication to reservation request activity, as shown in Figure (1). Once the reservation demand traffic standard is set, the sender completes the installation of the QoS path (switch 1–switch 3–switch 6–switch 7–switch 8).

Figure1. Network Topology



As described previously, the best route is (switch1– switch2– switch4- switch6 – switch8). Integrated service's reserve control protocol cannot choose the best path, which is (switch1– switch2– switch4- switch6 – switch8) because it uses the same path chosen by the routing protocols to set up QoS, which may have the worst performance. Also, if the path (switch1– switch2– switch4- switch6 – switch8) fails, as shown in Figure (1), the routing protocol determines a new shortest path. The routing algorithm notices that the link between (switch1– switch3– switch6- switch7 – switch8), it will recalculate the shortest path, then update the path databases on the routers to reflect the broken link between (switch1– switch2– switch4- switch6 – switch8). The video traffic goes through (switch1– switch2– switch4- switch6 – switch8). The network hasn't found the best path for the video streams to send. If the network depends on the integrated services design for QoS, the worst-case path may be chosen. To choose the best way, there is a need to consider two aspects:

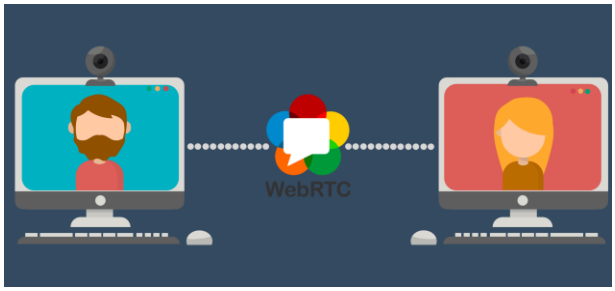
(A) Create a network architecture with the ability to select the optimal path. Selecting the best approach is made simpler by using a network-wide state. Figure (1) illustrates how hop-by-hop selection may cause real-time interactive video applications to select the worst-case scenario. The best path must be chosen by the network architecture based on the video program's QoS requirements.

(B) Create a control system that enables interactive real-time video applications to continuously request QoS from the network. Real-time interactive video apps must have a protocol to request Quality of Service (QoS) from the network. To ensure that the optimal course is selected, the protocol needs to be simple and only require the states that are necessary to be tracked.

2.1 Network Architecture

To solve Issue 1, a key component of the network architecture design must be the network concept, which compiles information regarding every one of the network's network devices. Keeping with the previous example, Issue 1 has been resolved by using SDN in conjunction with the OpenFlow protocol due to the network-wide view design decision. To more accurately depict our recommended network architecture in Figure (1), the SDN controller is now displayed in Figure (2). The SDN controller, also known as the control plane, and the data plane communicate via the OpenFlow protocol. A ray controller has been used in this investigation. Within the SDN controller, a video traffic control system operates on an eight-node topology. From the sender to the recipient, the entire network is visible to the SDN manager. The best route is selected by the SDN manager based on performance [9]. Besides, Figure (2) illustrates the architecture and how the various components of the design interact (only switches 1 and 2 are shown for simplicity). The four most crucial components are the switch, controller, sender, and receiver. The sender and user rely on switch1 and switch2 in terms of QoS. The controller switches over secure channels and exchanges information with the sender and recipient via OpenFlow. The user requests QoS from the network, as does the sender. Nevertheless, switch1 and switch2 are edge switches that form the traffic from the sender and the traffic from the recipient using packet shapers. As a result, network traffic may pass through multiple intermediary switches between switches 1 and 2. Nevertheless, only the edge switches change the way network traffic flows.

Figure2. Proposed Network Architecture



The routing algorithm's objective is to find every network path that exists inside the topology and then determine which path is the best, shortest, and least expensive to successfully route the massive flow. After being pushed, the switches will direct every single flow through them [11]. To decide which path has the lowest overall cost, the algorithm must also determine how many bytes are transmitted and received on the switch ports. Finally, it must compute the computational expense of each path. Finally, three different routing strategies are used by the model; these are called a short test path with delay, a quick test path with hop, and a hybrid for both of these. It is crucial to remember that either the sender or the recipient can initiate a request. The following are some of the strategies that are included in the proposed model protocol: (a) controller strategy, (b) switch strategy, the transmitter strategy, and (c) receiver strategy.

2.2 Architecture Implementation

The controller for the OpenFlow switch in this project. Because the Quality of Service (QoS) module is already in place, it is simple to implement the QoS function via the web page and notify the controller. The suggested model can define some QoS queues in advance and then send requests that correspond to the specific type of data, like "the source address is 10.0.0.1." When the switch's requirements matched those of the users, it would add QoS rules accordingly. Ultimately, this application can consume bandwidth continuously and still deliver a stable and consistent service to users.

The edges, nodes, and hosts from the previous section are combined to create the network topology using a mininet [12]. During the process of developing the network topology, the SDN controller will function as the interface controller, enabling packet routing in that network design [13]. In this project, the network topology is established according to the following specifications: SDN controller (a), eight hosts (b), and two switches (c). The SDN Open flow protocol uses a network structure since the two primary objectives of data networks are to handle issues and speed up the network. There are two layers to the switch architecture in this topology: aggregation, core, switches, and edge switches. All of the switches are connected.

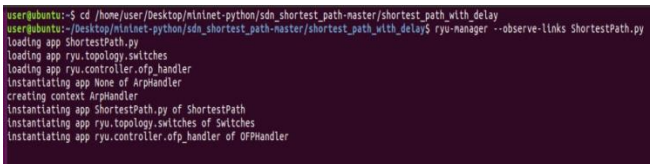
2.3 Topology Generation

In software-defined networking, the controller is in charge of storing all network topology data [14]. Next, the recommended method is applied to find out which of the potential network topologies has the shortest path. When using Mininet, the ping all command checks if all of the hosts are active and reachable from one another as well as the connectivity between them all. This command is used to verify that every host in the system is connected to every other host. The hosts receive all packets and the percentage of dropped packets is 0% when there is activity on the hosts. The ping all command might take up to an hour and a half, or even longer, to connect two hosts successfully. The bandwidth of the links that link the hosts and switches, in terms of data packet transmission, is 0.2 Mbit, 0.1 Mbit, and 0.05 Mbit, respectively. The following procedures are used to create a fat topology for all routing algorithms: Launch the Mininet terminal, type in the algorithm's path, and produce a fat tree model. The Mininet then creates links, activates the controller, and adds 8 hosts and 8 switches.

2.4 Controller and Mininet Communication

The default port 8080 is used by the switches and controller to communicate to create a connection between the controller and the network topology [15]. The topology assigns a unique port to each switch so that the packets that are transmitted can be tracked separately. There are twenty Openflow switches connected to eight hosts as part of this project's scope [16]. The first four switches—S1001 through S1004—among these 20 switches are regarded as the core switches. The aggregation switches are the next eight switches, from S2001 to 2008. The following eight switches are edge switches, and eight hosts in the network are connected to them altogether. The controller script can be executed to calculate the shortest path possible within the network topology. To send and receive packets between the switches, all of the handler events and topology switches have been loaded, as seen in Figure (3).

Figure3. Run Controller for loading files.



2.5 Evaluation

In SDN, OSPF—which talks with routers and the OpenFlow controller—is used to identify the optimal path first. The controllers that rely on the Address Resolution Protocol store the packet data. because there are two modes of operation for the controller: proactive and reactive. While switches forward the necessary Address Resolution Protocol to the controller in the reactive mode and determine the best path, the controller receives information from switches in the proactive setting.

2.6 Stander WebRTC Model

The video submitted for HOST001 and HOST008 is assessed in this section. The hop count for both the "HOST001 to HOST008" and "HOST008 to HOST001" paths is 5, which is the same. Nevertheless, the way that packets travel through the switches is different. It has taken the following actions to run a brief test path with the hop in the Mininet terminal:

Input the short test path with the hop directory.

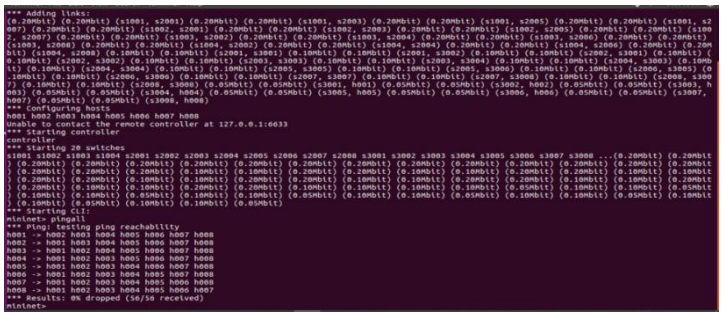
Run a short test path with a slandered WebRTC Model.

The next step is to run ICMP-protocol-ping for all networks.

2.6.1 Architecture Model

In this section, the video transmission for HOST001 and HOST008 has been assessed. Every link between hosts HOST001 and HOST008 has its delay (weight) determined. On the other hand, the path with the lowest wait time is the optimal one for packet transmission. The Mininet terminal executes the algorithm and the suggested model to run a brief test path with the hop. Running ICMP-protocol-ping on every network is the next step. Figure (4) shows the ICMP process in action.

Figure4. Process of running for all networks.



The controller then performs a make-discovery for every network before using ping to determine the shortest path. The algorithm that follows is controller-make-make-host1-client-generate-Tcp-traffic-delay. Lastly, the algorithm that generates the UDPP traffic delay is controller-make-host1-client.

2.6.2 Measurement of TCP and UDP Throughput

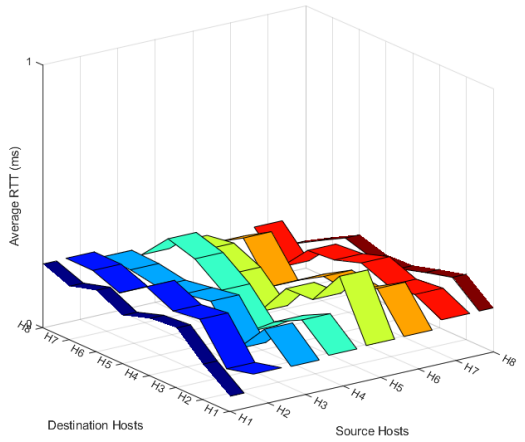
The TCP packet (video transmission) is exchanged between two hosts, identified as HOST001 and HOST (H008). In client mode, TCP packets are sent from HOST001 to HOST008, and the bandwidth and time needed to send the standard 85.3KByte window size are computed. HOST008, on the other hand, runs in server mode, meaning that it is the one that receives TCP packets and applies the same computations for time and bandwidth to the information it receives. As a result, we evaluated the TCP packet transmission for the algorithm and transmission of slandered WebRTC. Furthermore, two hosts that exchange TCP packets with one another are HOST001 and HOST008. HOST001 transmits UDP packets to HOST008 at a rate of 10M while in client mode. It calculates the amount of bandwidth and time needed for the data transmission. Additionally, the sent-message totals are computed. In server mode, HOST008 employs the same receiving bandwidth and delay computations while accepting UDP packets. Consequently, we can state that the average throughput is 144 Kbits/sec, or 333 Kbytes, between 0 and 28.5 seconds. Only 303 datagrams of information were lost after ten tries. The algorithm's and WebRTC's UDP data transfer has been measured.

2.6.3 Round-Trip Time Measurement

The round-trip time (RTT) is the duration,

measured in milliseconds, that a browser needs to send a request to a server and then wait for a response. It is a crucial component in determining network latency and page load time, and it is a crucial performance metric for web applications. The round-trip time results at a typical basis, maximum, lowest, and standard RTT are displayed in Figure (5).

Figure5. Typical RTT for the proposed algorithm and WebRTC.



3. Conclusion

This work has demonstrated that the main objectives of data networks are to increase fault tolerance and network throughput. The SDN OpenFlow protocol makes use of network topology in this regard. The three-tiered switch architecture that arises from this specific topology, in which all of the switches are connected to and communicating with one another, consists of core, edge, and aggregation switches. Additionally, in SDN, OSPF—which communicates with all of the routers via the OpenFlow controller is used to determine the optimal path first. The controllers that rely on the Address Resolution Protocol store the packet data. While switches forward the necessary Address Resolution Protocol to the controller and determine the best path in the reactive mode, the controller receives information from switches in the proactive setting. Consequently, a novel work utilizing Software-Defined Networking (SDN) and Web Real-Time Communication technology has been completed for network architecture selection to determine the optimal path from a network-wide viewpoint.

References

1. A. T. K. Al-Khayyat and S. A. Mahmood, “Peer-to-peer media streaming with HTML5,” *Int. J. Electr. Comput. Eng.*, vol. 13, no. 2, pp. 2356–2362, 2023, doi: 10.11591/ijece.v13i2.pp2356-2362.
2. D. Wobuyaga, S. T. Arzo, H. Kumar, F. Granelli, and M. Devetsikiotis, “Latency and Reliability Aware SDN Controller: A Role Delegation Function as a Service,” in *2023 IEEE 13th Annual Computing and Communication Workshop and Conference, CCWC 2023*, 2023, no. June, pp. 205–211. doi: 10.1109/CCWC57344.2023.10099225.
3. T. P. Raptis and A. Passarella, “A Survey on Networked Data Streaming with Apache Kafka,” *IEEE Access*, vol. 11, no. August, pp. 85333–85350, 2023, doi: 10.1109/ACCESS.2023.3303810.
4. T. Semong *et al.*, “Intelligent load balancing techniques in software defined networks: A survey,” *Electron.*, vol. 9, no. 7, pp. 1–24, 2020, doi: 10.3390/electronics9071091.
5. A. Hazra, P. Rana, M. Adhikari, and T. Amgoth, “Fog computing for next-generation Internet of Things: Fundamental, state-of-the-art and research challenges,” *Comput. Sci. Rev.*, vol. 48, no. May, p. 30, 2023, doi: 10.1016/j.cosrev.2023.100549.
6. N. Mansoor, M. I. Hossain, A. Rozario, M. Zareei, and A. R. Arreola, “A Fresh Look at Routing Protocols in Unmanned Aerial Vehicular Networks: A Survey,” *IEEE Access*, vol. 11, no. June, pp. 66289–66308, 2023, doi: 10.1109/ACCESS.2023.3290871.
7. S. Tumula *et al.*, “An opportunistic energy-efficient dynamic self-configuration clustering algorithm in WSN-based IoT networks,” *Int. J. Commun. Syst.*, no. August, pp. 1–21, 2023, doi: 10.1002/dac.5633.
8. V. H. Kelian, M. N. M. Warip, R. B. Ahmad, P. Ehkan, F. F. Zakaria, and M. Z. Ilyas, “Toward Adaptive and Scalable Topology in Distributed SDN Controller,” *J. Adv. Res. Appl. Sci. Eng. Technol.*, vol. 30, no. 1, pp. 115–131, 2023, doi: 10.37934/araset.30.1.115131.
9. H. Eltaief, A. El Kamel, and H. Youssef,

"MSA-SDMN: multicast source authentication scheme for multi-domain software defined mobile networks," *J. Inf. Telecommun.*, pp. 1–24, 2023, doi: 10.1080/24751839.2023.2250123.
10. Y. Al-Dunainawi, B. R. Al-Kaseem, and H. S. Al-Raweshidy, "Optimized Artificial Intelligence Model for DDoS Detection in SDN Environment," *IEEE Access*, vol. 11, no. August, pp. 106733–106748, 2023, doi: 10.1109/ACCESS.2023.3319214.
11. S. A. Combes, N. Gravish, and S. F. Gagliardi, "Going against the flow: bumblebees prefer to fly upwind and display more variable kinematics when flying downwind," *J. Exp. Biol.*, vol. 226, p. 13, 2023, doi: 10.1242/jeb.245374.
12. J. Hammer, D. Kimovski, N. Mehran, R. Prodan, and H. Hellwagner, "C3-Edge - An Automated Mininet-Compatible SDN Testbed on Raspberry Pis and Nvidia Jetsons," in *Proceedings of IEEE/IFIP Network Operations and Management Symposium 2023, NOMS 2023*, 2023, p. 5. doi: 10.1109/NOMS56928.2023.10154397.
13. S. J. Rashid, A. M. Alkababji, and A. S. M. Khidhir, "Performance evaluation of software-defined networking controllers in wired and wireless networks," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 21, no. 1, pp. 49–59, 2023, doi: 10.12928/TELKOMNIKA.v21i1.23468.
14. Y. Yang, M. Ye, Q. Jiang, and P. Wen, "A Novel Node Selection Method in Wireless Distributed Edge Storage Based on SDN and Multi-attribute Decision Model," 2023.
15. B. M. Rashma and G. Poornima, "Performance Evaluation of Multi Controller Software Defined Network Architecture on Mininet," *Lect. Notes Networks Syst.*, vol. 80, pp. 442–455, 2020, doi: 10.1007/978-3-030-23162-0_40.
16. M. I. Kareem, M. N. Jasim, H. I. Hussein, and K. Ibrahim, "Performance evaluation of RYU controller under distributed denial of service attacks," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 32, no. 1, p. 252, 2023, doi: 10.11591/ijeecs.v32.i1.pp252-259.
17. Manhal Mohamad Basher, "Designing SDN Approach Using WebRTC for Low Bandwidth Over Data Communication", Iraqi Journal of Statistical Sciences, Vol. 21, No. 2, 2024, Pp (38-44), DOI 10.33899/iqjoss.2024.185238.