Journal of Electronics and Informatics (ISSN: 2582-3825)
www.irojournals.com/iroei/

Shortest Path Routing Performance Evaluation over SDN Environment

Roshani Ghimire[1], Ram Kumar Basnet[2]

[1]Advance College of Engineering and Management, Tribhuvan University, Kathmandu, Nepal

[2]Department of Graduate Studies, Nepal College of Information Technology, Balkumari, Lalitpur, Nepal

E-mail: [1]roshanigd@gmail.com, [2]basnetramkumar@gmail.com

Abstract

Static routing has a manual configuration setup system, and the scope of static routing in an SDN network is just for small networks. The solution to this problem rises up with the new technology defined as software-defined networking (SDN) based on shortest path first dynamic routing. SDN has the facility of a centralized controller that smooth the controls and routes computation over a data packet. The performance analysis of SDN networks that have SDN switches connected to the network based on the shortest path first protocol are simulated on Mininet. The POX controller with Mininet programming feature for creating smart topologies was chosen. In this research, the SDN network using Dijkstra's algorithm, Bellman-Ford algorithm, extended Dijkstra's algorithm and Floyd Warshall Algorithm were implemented. The quality factors of SDN created by using four algorithms are measured in terms of delay, jitter, latency, packet loss, transmit, received, throughput, and bandwidth based on experimental results and European Telecommunications Standards Institute (ETSI) data. The performance parameters of SDN network topology created using Dijkstra's, bellman ford, extended Dijkstra's, and Floyd Warshall algorithms were compared and the experimental results showed that Bellman-Ford algorithm is better in terms of performance parameters than the other three algorithms.

Keywords: SDN, POX Controller, Routing Algorithms

DOI: https://doi.org/10.36548/jei.2023.4.004
Received: 02.12.2023, received in revised form: 21.12.2023, accepted: 08.01.2024, published: 19.01.2024

1. Introduction

Software-Defined Networking (SDN) is a software-based controllers used in networking that are used to communicate with hardware infrastructure and direct traffic on the network. SDN provides an application programming interface (API) that creates a virtual network or controls traditional hardware through the software. SDN is defined as the smart networks where forwarding plane is separated from the network control plane with features of controlling many routers and switches. SDN has dynamic programmability in forwarding packets. It logically centralizes the network view globally. On the control plane, many applications can be defined. SDN has OpenFlow protocols, which helps to manage new networks. The best solutions to the difficulties at static routing used for smaller networks are given by dynamic routing on SDN.

The system cannot recover from link connection and node failure in static routing as compared with dynamic routing. Static is rigid and inflexible when network speed increases with topology [1]. The configuration of the router and management of switches are too difficult and time consuming tasks for static routing. Moreover, it becomes difficult to implement a static network for large organizations where routers are manually configured statically with static IP at earlier networks, due to which link failure causes the link rerouting [2]. To analyze and monitor the overall performance of the network, each routing and switching device must be analyzed separately and individually [3]. The control plane of each switching device must be secured individually to prevent the easy network access by the intruders. The implementation of static routing in small and large networks is a major problem that limits the network and system administrator to allocate resources like bandwidth and server access effectively. The new technology software defined networks (SDN) is based on OpenFlow system which overcomes the problem of static routing by dynamic routing. The SDN networks have a centralized controller [4] which controls the overall network system devices like SDN switches.

The mininet network emulator is used for the experimental analysis using the POX controller [5] for SDN routing. The SDN topology includes five SDN switches connected to a single POX controller. The network switches are connected to each other by edge links, as per the model diagram. Three different cases are considered. In the first case, edges with a weight of1 megabit/sec are assigned for link connection at the SDN topology. In the second case, edges with a weight of10 megabits/sec are assigned for link connection at the SDN topology. In the

third case, different edge weights say (6, 8, and 10) megabits/sec are assigned for link connection at SDN topology. In each three cases, first all links are up and parameters are found and analyzed. The parameters selected for analysis are delay, jitter, packet loss, latency, packet transmit and receive, TCP bandwidth, and finally throughput.

The original Dijkstra's algorithm extends Dijkstra's shortest path algorithm to consider not only the edge weights but also the node weights for a SDN topology where a POX controller is used. POX controller is implemented in all algorithms i.e. Dijkstra's, Bellman Ford, Extended Dijkstra's and Floyd Warshall algorithms and comparison is done between them by taking different performance parameters under SDN network topologies.

In this research, the Dijkstra's, Bellman Ford, Extended Dijkstra's, and Floyd Warshall algorithms are used for coding applications to communicate with controllers using OpenFlow protocols to route the packets in SDN. The performance parameters (such as delay, jitter, packet loss, packet transmitted and received, TCP bandwidth, latency, and throughput) are analyzed for all four algorithms, and the best algorithm is chosen. The coding has been done using Python programming for data visualization and graph plots. A mininet network emulator and SDN controller are used as simulation tool for actual realization of network topology. Different tools like iPerf and ping are used for finding parameter values.

The rest of this study is structured as follows: Section 2 presents relevant works related to SDN and various algorithms used in SDN routing. The proposed method is discussed in Section 3, where the details about the model diagram of SDN are implemented. Section 4 provides results, outputs, observations, and analysis, while Section 5 concludes the paper with future work.

2. Related Work

The author in [4] has proposed the evaluation of performance for multihop routing in an SDN environment. In this study, various scenarios are designed using different network topologies and devices. The cost factor is computed every time while routing the data from source to destination. The routing is not only based upon traditional approaches like shortest path and Dijkstra's routing algorithms. Various topologies like spanning tree, ring tree, and dual tree are evaluated on the basis of factors like bandwidth usage, cost, and link efficiency. The author used the OPNET simulator where various switches are deployed on various

locations across the world map. The authors compare the average cost per link and bandwidth efficiency for spanning and ring topologies. The author cannot deploy some random networks to evaluate the performance of routing in terms of scalability factors.

In this related study [6], the author has discussed the performance analysis of open shortest path first (OSPF) routing protocols in SDN, where the author analyzes how a dynamic routing protocol (OSPF) works in a SDN and the stability of the network with parameters like convergence time, quality of service, and round trip time after sending video streamed between senders and receivers. The comparison is done between traditional networks and SDN for evaluation of performance and stability and concludes that SDN is better in terms of quality of service. This study provides a practical test to evaluate the behavior of OSPF in both SDN and a physical topology [7]. The SDN virtual environment with OSPF has been experimented using Quagga and Mininet and concluded that SDN gives less loss rate, jitter, and better streaming quality than traditional networks. The authors presented OSPF routing algorithms and their behavior in terms of scalability and performance only.

In this study [8], the author has done a survey of the disjoint multipath routing in SDN for multi constraint quality of service (QoS). This study addresses the sustainability of QoS requirements and the efficiency of path optimization for a single physical network like SDN. The proposed related paper did not focus on dynamic regulation of network behaviour, which provides software-oriented control of the network and facilitates centralized path routing. It only balances the load on the core network, reduces congestion, and improves reliability using disjoint paths.

The author in [9] has evaluated the estimation of the bandwidth link in SDN for the performance evaluation of proposed algorithms. This research provided the evaluation of network performance by using Floodlight SDN controller. The greedy method is used to solve network problems by reducing the computational time needed to estimate the available bandwidth on a link [10]. The author proposed a monitoring process to reduce the number of switches to be interrogated to cover all the links within SDN networks.

In this study[11], the authors directly used an evolution of the multipath routing algorithm in SDN. The proposed paper overcomes the limitation and improves TCP performance at a flow level using genetic algorithms. This also resolves the scalability problem. It allows one to choose good solutions that offers the smallest number of flow splits while

maintaining a good compromise in terms of the total network flow and the total cost of the selected paths. It cannot offers the best overall network usage. It improves the performance at the flow level as the packet reordering problem is reduced. We are looking at setting dynamically algorithm parameters, including the network topology, number of flows and the number of paths each flow is allowed to transmit on.

Previous related works illustrated only on the performance of routing protocols with single algorithms and link cost estimation on SDN [9], [12], [13] but does not program routing dynamically with versatility in addition to comparing and analysing the performance parameters of various algorithms used in new dynamic routing technology called SDN routing.

The SDN systems are designed on the basis of shortest path algorithms. It compares performance parameters in order to facilitate the implementation. This study mainly focuses on designing a software-defined networking (SDN) using Dijkstra's, Bellman-Ford, extended Dijkstra's and Floyd Warshall algorithms in SDN routing. The performance parameters like packet loss, received, send, latency, delay, jitter, bandwidth, and throughput are analyzed.

3. Methodology

3.1 Architecture

The general architecture of SDN is presented in Figure 1. The architecture gives the separation of control plane from data plane. Data plane i.e. SDN switches belong to the infrastructure layer and control plane i.e. controller belongs to control layer.

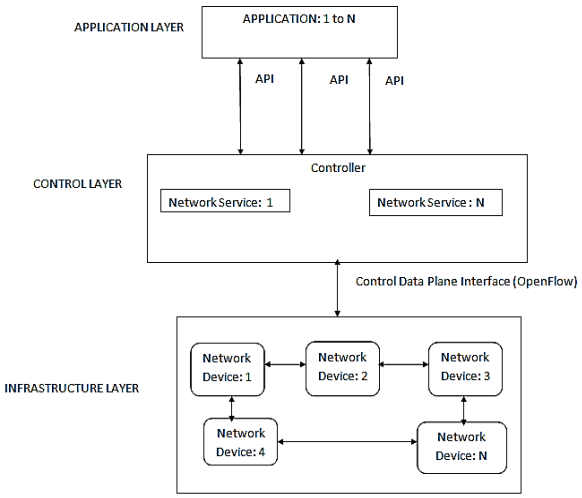

Figure 1. General Architecture of SDN

The Controller and SDN switches are communicated by OpenFlow network protocol. OpenFlow [14], [15] is open protocols defined between the data plane and control plane of the SDN architecture. The switch based on OpenFlow includes more flow tables. An OpenFlow controller can add, delete, and update flow entries in the flow table. Each flow table in SDN switches contains a set of flow entries. After matching entries with flow tables, forwarding the packet to the next switch by port is done. If not matched then packets are forwarded to the controller over OpenFlow channel or dropped.

3.2 Data Collection

The ETSI [14] standards dataset are selected for test and validation. For verification, the data is collected by building the SDN topology virtually after programming four algorithms dynamically. The ETSI standard gives main variables affecting performance of networks. They are mainly packet losses, jitters, and delays. Degradation categories given by ETSI are perfect, very good, good, medium, bad, and poor.

3.3 Model Development

Earlier switches and routers had both data and control planes together. Here, the choice of SDN switches is due to the feature of a separate data plane and control plane for SDN, so that controlling the infrastructure plane can be done easily. In this research, 5 SDN switches and 7 SDN switches are chosen because SDN uses Open vSwitch. It omits the earlier switches and router's tasks by OpenFlow protocol switch, which allows the topology to communicate automatically and reduces the topology's complexities. Less switches help to find the effectiveness of topologies easily. The choice of network switches is based on the energy efficiency of networks. To reduce their usage of energy consumption, fewer switches are chosen. To optimize the routing of the packets for reducing power consumption, reduce the network hardware, i.e., 5 and 7 switches, so that performance analysis can be done efficiently. To reduce the number of active switches, the link is down and packet flows through the less number of switches to find the shortest path.

To gather the network traffic at a subset of network and hardware switches, different choices of weights are necessary [16]. The choice of weights assigned at edges reduces the rate of flow at which the link functions. SDN switches can vary the network at different rates such as 1 Megabit/sec, 10 Megabit/sec, 100 Megabit/sec or different weights like (8, 6, 9, 10) Megabits/sec assigned based on the bandwidth required. It provides alternative paths for

packets flowing through the network. For measuring the variation of different performance parameters like jitter, delay, latency, packet loss, TCP bandwidth, and packet transmit and receive, and finally throughput, the necessary weights can be assigned, and number of switches can vary. The choice of weights give the flow behavior of links and switches. As OpenFlow switches are based on flow tables, performance parameters vary, and analysis can be done effectively if switches vary for different choices of weights. Less switches, less flow entries in the flow table at switches, less routing reduces power consumption, i.e., the link is minimized, which gives a better shortest path. The choice of 5 and 7 switch models shown in Figures 2 and 3 is useful and best suited for medium and small enterprises. The 5 and 7 switches SDN network model diagram having data planes which are controlled by control plane or layer are depicted in Figure 2 and Figure 3 respectively.

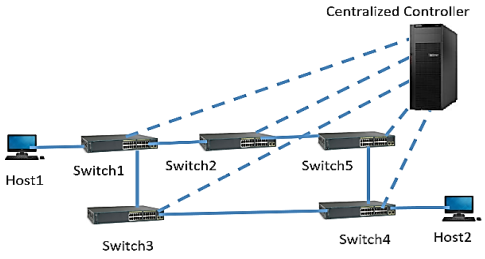

Figure 2. SDN with 5 Switches

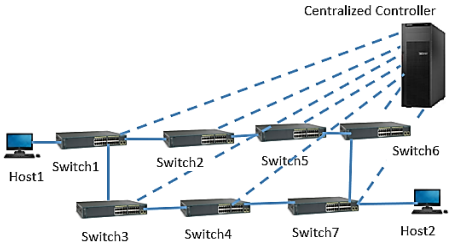

Figure 3. SDN with 7 Switches

Following algorithms are implemented in this research, (a) Dijkstra's algorithm, (b) Bellman Ford algorithm (c) Extended Dijkstra's algorithm, and (d) Floyd Warshall algorithm.

The performances analysis was carried out with the metrics (a) Delay, (b) Jitter, (c) Packet loss, (d) Latency, (e) Bandwidth, and (f) throughput.

4. Results and Analysis

POX controller was selected for experiment, where four algorithms i.e. Dijkstra's, Bellman Ford, extended Dijkstra's and Floyd Warshall algorithms are configured. The virtual machines (VM) created at Oracle VirtualBox. One Mininet VM is used for creating SDN network topology and Ubuntu Linux VM is used for remote pox controller access. The SDN topology was created at Mininet VM by using Mininet programming and some extended Python coding. The POX controllers are connected remotely via Ubuntu Linux VM. All algorithms run via controller programming in SDN topology that was created at Mininet VM.

The various performance parameters are measured by accessing SDN topology with the help of Mininet VM commands. Finally, measurements are compared for all given four algorithms.

4.1 Delay Test

The delay for equal weight of 1Mbps are shown in Figure 4. Delay for equal edge weight (1 Megabit/sec) is assigned for different packet sizes (56, 100, 200, and 400) measured in bytes and found that during comparison between four algorithms, there are more delay time in millisec for extended Dijkstra's algorithm than other algorithms. Similarly, Delay for equal edge weights (10 Megabits/sec) assigned for different packet sizes (bytes) and delay for different edge weights assigned for different packet sizes (bytes) are obtained and analyzed. Single link down case for weight 1 Megabit/sec and 10 Megabits/sec are also analyzed.

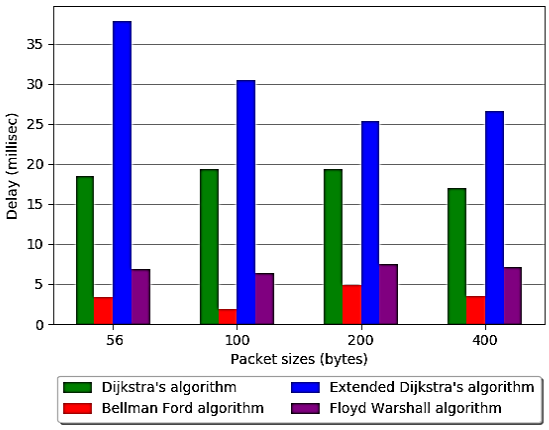

Figure 4. Delay for Equal Weight 1 Mbps (all link up)

Delay Analysis: Both linkup and single linkdown cases are virtually measured, and it was found that extended Dijkstra's algorithm requires more time i.e. more delay than other remaining three algorithms, as shown in graph plot analysis and obtained results data. Ultimately, it was concluded that the Bellman-Ford algorithm is better than Dijkstra's, extended Dijkstra's, and Floyd Warshall algorithms in terms of delay due to least delay. Bellman-Ford relaxes all the edges of the network topology while Dijkstra's and extended Dijkstra's selects the nearest vertex that has not been processed, which is the main reason behind delay. Floyd Warshall algorithm calculates all pairs shortest path. Similar results were obtained for the 7 switches network model.

4.2 Jitter Test

The jitter and packet loss for equal weights are shown in Figures 5 and 6. Equal weights (1 and 10) Megabits/sec are assigned. Jitter parameters are measured for four algorithms in SDN. In all link up cases, both equal edge weights (1 and 10) Megabits/sec assigned separately in the SDN network. It found that extended Dijkstra's algorithm has more jitter than other algorithms at equal weights 10 Megabits/sec. For unity weights assigned with all linkup cases, Dijkstra's has more jitter and extended has least jitter. Similar measurements were done for the link down case for both edge weights 1 and 10 Megabits/sec assigned separately at SDN network. Each network had equal weights and found extended Dijkstra's algorithm has more jitter than other algorithms for edge weight 10 Megabits/sec case. For unity weight assigned for link down case, Bellman-Ford, extended Dijkstra's and Floyd war-shall has similar jitters. During different weights (8, 10, 6) Megabits/sec assigned in SDN topology, there are very less jitter found for Bellman-Ford than other algorithms.

Jitter Analysis: It concluded that bellman ford performance is better than other algorithms in terms of jitter. Less jitter means better smoothness in performance. The reason behind it is that Dijkstra's algorithm is greedy in nature. Bellman-Ford algorithm works in a dynamic programming approach i.e., intermediate data is used for next level data. Floyd Warshall algorithm used for finding shortest path for all pairs. Similarly, as above Bellman-Ford is better for 7 switches model diagram.

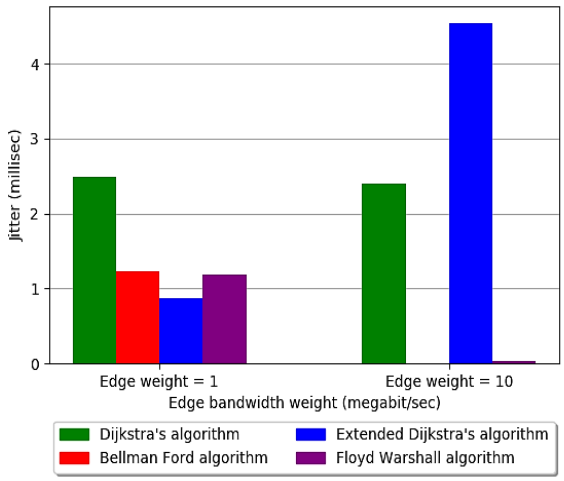

Figure 5. Jitter for All Equal BW (all link up)

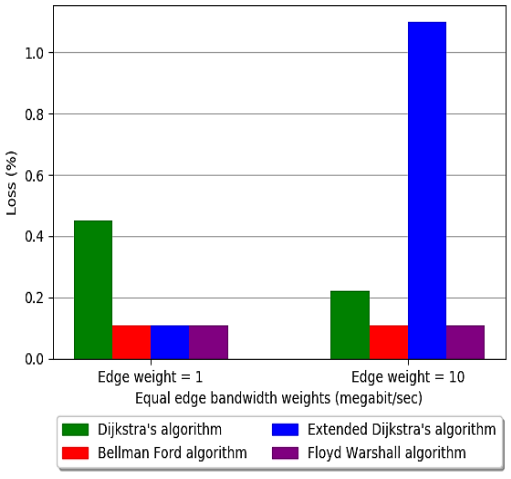

Figure 6. Packet Loss for Equal Bandwidth Edges Weights (all link up)

4.3 Packet Loss Test

Equal weights (1 and 10) Megabits/sec are assigned. Equal edge weights 1 and 10 Megabits/sec are assigned to a separate SDN network for the same topology. In edge weight with 10 Megabits/sec, extended Dijkstra's has more packet losses than other algorithms. In unity edge weight assigned, Bellman-Ford, extended Dijkstra's, and Floyd Warshall algorithms have equal packet losses. For the Linkdown case, Dijkstra's has more packet losses than other algorithms for edge weights of 1 Megabit/sec while for edge weights of 10 megabits/sec, all algorithms have the same packet loss found. Different edge weights are assigned for both all link up and single linkdown cases. Again, it same packet losses were found for Dijkstra's, Bellman-Ford, extended Dijkstra's, and Floyd Warshall algorithms at both link up and link down cases.

Packet Loss Analysis: The result analysis concluded that Bellman-Ford, Floyd Warshall algorithms have less packet loss at condition edge weight of 1 Megabit/sec. while at 10 Megabits/sec and for different edge weights and link down condition, all algorithms have same packet losses. Due to relaxation of all edges at Bellman-Ford, packet loss is less. Since different bandwidth weights are assigned, relaxations vary so that packet losses obtained are similar at different bandwidth cases. For 7 switch models, most of the cases have different same packet loss. Bellman-Ford and Floyd Warshall have less packet loss in this model.

4.4 Packet Transmitted and Received Test

The round trip time (RTT) tests of packet (64 bytes in size) transmission with different algorithms are shown in Figure 7. Equal weight of 1 Megabit/sec assigned and 64 bytes packet sizes are transmitted and RTT are obtained. During ping time for first packet initially, Floyd Warshall has too high round trip time than other algorithms for second, third, fourth and so on up to 10 packets transmitted, extended Dijkstra's algorithm have high Round trip time while Bellman-Ford and Floyd Warshall have very less RTT near about zeroes. All links are up in this case.

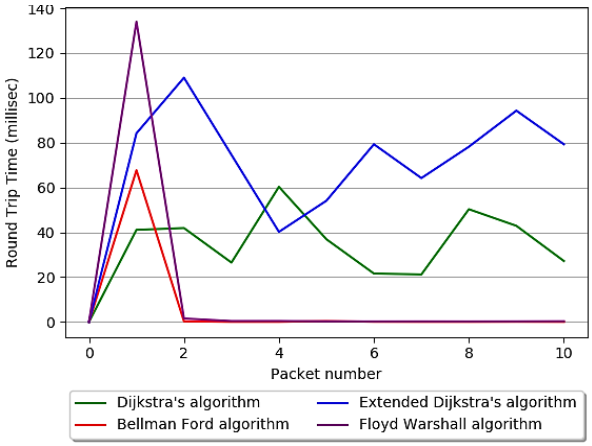

Figure 7. 64 Bytes Packets Transmitted, Weight 1 Megabit/sec (all link up)

Equal edge weights 10 Megabits/sec is assigned and analyzed for all link up cases. For this, 64 byte packet sizes are transmitted and round trip time (RTT) was obtained. Dijkstra's and extended Dijkstra's have a higher round trip time than Bellman-Ford and Floyd Warshall algorithms for most of the packets transmitted but for different edge weights i.e. 10, 8, 6 megabits/sec. Initially, Dijkstra's and extended Dijkstra's have less round trip time than Bellman-Ford and Floyd Warshall algorithms but as the number of transmitted packets increases RTT also increases for extended Dijkstra's and Dijkstra's algorithms.

Equal edge weights 10 Megabits/sec are assigned and analyzed for single link down. Similar way as above 10 packets having 64 bytes each are transmitted and received by the receiver. Initially, Bellman-Ford and Floyd Warshall algorithm had high RTT but quickly, it was near to zero. Similarly for Different weights of (6, 10, 8) Megabits/sec assigned for edges, the Dijkstra's and extended Dijkstra's has more RTT than Bellman-Ford and Floyd Warshall for later number of packets than initial number of packets .

Packet Transmitted and Received Analysis: In all above cases, during ping time for first packet, Bellman-Ford and Floyd Warshall algorithms have too high round trip time than Dijkstra's and extended Dijkstra's algorithm while at other second, third, fourth and so on up to 10 packets transmitted, Dijkstra's and extended algorithm have high Round trip time than the. Bellman-Ford and Floyd Warshall. As compared to each byte between link up and link down cases, RTT at cases of link down is less as compared with link up cases at all conditions. The analysis gives the result that Bellman-Ford is better than other algorithms during all cases above because at almost all times Bellman-Ford has less RTT due to less delay obtained. There

are less RTT for both Bellman-Ford and Floyd Warshall algorithms at 7 switches model when packets are transmitted.

4.5 Latency Test

The latency values for different algorithms for equal weight is shown in Figure 8. Latency for Equal weight of1 Megabit/sec is assigned for edges of different packet sizes (bytes). All edges are linked up. RTT for all packet sizes (64 bytes, 100 bytes, 200 bytes and 400 bytes) are obtained in this case. Packet sizes (bytes) versus round trip time (millisec) are plotted in graphs and found that Dijkstra's and extended Dijkstra's algorithms have very high RTT as compared to Bellman-Ford and Floyd Warshall algorithms. Bellman ford has the least RTT among all algorithms. The significance of latency is to provide better shortest path and analyze the performance of the switches network model in terms of time. Dijkstra's and extended Dijkstra's algorithms have larger latency than other algorithms at 7 switches.

Equal weights 10 Megabit/sec are assigned for edges. All edges are linked up. RTT for all packet sizes (64 bytes, 100 bytes, 200 bytes and 400 bytes) are obtained in this case. Packet sizes (bytes) versus round trip time (millisec) are plotted on graphs and found that Dijkstra's and extended Dijkstra's algorithms have very high RTT as compared to Bellman-Ford and Floyd Warshall algorithms. Similarly, latency for different weights assigned for different packet sizes are analyzed. In each analysis, packet sizes (bytes) versus round trip time (millisec) are plotted in graphs and found that Dijkstra's and extended Dijkstra's algorithms have very high RTT as compared to BellmanFford and Floyd Warshall algorithms. The Floyd Warshall algorithm has less as compared with the Bellman Ford algorithm.

Very high round trip time for Dijkstra's and extended Dijkstra's obtained than Bellman-Ford and Floyd Warshall. Similarly, latency for different weights assigned for different packet sizes are analyzed. In each analysis, packet sizes (bytes) versus round trip time (millisec) are plotted in graphs and found that Dijkstra's and extended Dijkstra's algorithms have very high RTT as compared to Bellman-Ford and Floyd Warshall algorithms. Link was down for a single edge only. Floyd Warshall has more latency than Bellman-Ford algorithms in this case.

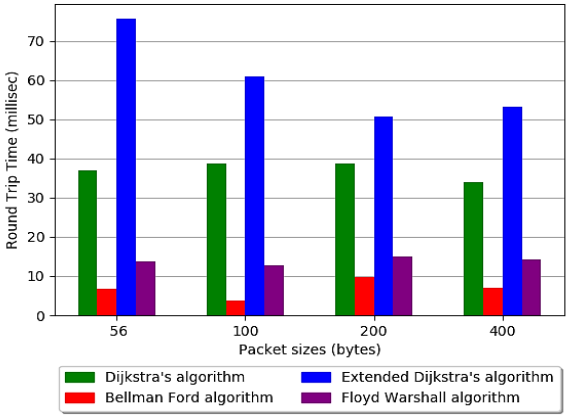

Figure 8. Latency for Equal Weight 1 Megabit/Sec (all link up)

Latency Analysis: The analysis concluded that the performance measurements for latency is better enough for the Bellman-Ford algorithm than other algorithms. Low latency i.e. good SDN network and good speed and vice versa. It justified that due to low delay and jitters in Bellman-Ford algorithm, latency also tends to be low due to edges relaxations.

4.6 TCP Bandwidth Test

Equal edge weight of 1 Megabit/sec is assigned. TCP bandwidths are calculated for host 1(client) and host 2(server) as depicted in Figure 9 during all edges are linked up. Equal edge weights 10 Megabits/sec are assigned. TCP bandwidths are calculated for host 1(client) and host 2(server). All edges are linked up. Similarly, for single link down case, equal weight of 1 Megabit/sec is assigned and analyzed. TCP bandwidths are calculated for host 1(client) and host 2(server). Similarly, for single link down case, equal weights of 10 Megabits/sec are assigned and analyzed. TCP bandwidths are calculated for host 1 and host 2. Here, different weights are assigned for edges. All edge links are up. TCP bandwidth for host h1 (client) and host h2 (server) are calculated and compared for all four algorithms. Here, different weights are assigned for edges. Single edge link down. TCP bandwidths for host h1 (client) and host h2 (server) are calculated and compared for all four algorithms.

TCP Bandwidth Analysis: It concluded from TCP bandwidth at host1 (client) and host2 (server) that Bellman-Ford and Floyd Warshall algorithms transfer high TCP bandwidth rate as compared to Dijkstra's and extended Dijkstra's algorithm for edge weights 10 Megabits/sec assigned and for different weights assigned at edges. But it is opposite to the case, which means Dijikstra's algorithm works well when edge weights equals weight 1 Megabits/sec for both all link up and single link down. It is due to Dijkstra's works well for

hop count as a weight equal to one. For the 7 switches model, Dijkstra's and extended Dijkstra's algorithms have least TCP bandwidth in most of the cases.

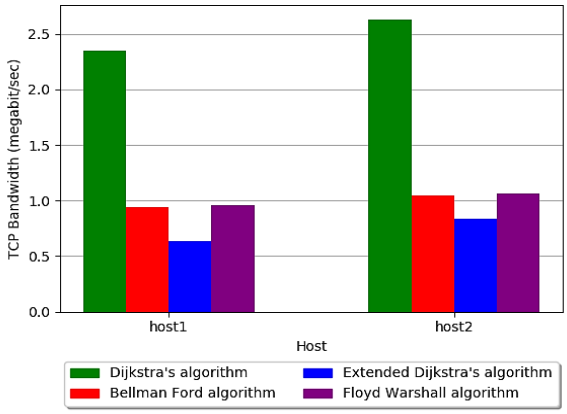

Figure 9. TCP bandwidth for Equal Weight 1 Megabit/Sec (all link up)

4.7 Throughput Test

Equal weight 1 Megabit/sec are assigned and throughput are calculated for host 1(client) and host 2(server) as shown in Figure 10 with the scenario of all edges are linked up. Equal edge weights of 10 Megabits/sec are selected. Host 1(client) and Host 2(server) throughput is calculated. Similarly, for link up cases, equal weights 10 Megabits/sec are assigned and throughput are calculated for host 1 and host 2. Single link was down in this case also. Here, different weights are assigned for edges. Throughput for host h1 (client) and host h2 (server) are calculated and compared for all four algorithms. Single link was down and throughput for host h1 (client) and host h2 (server) are calculated and compared for all four algorithms.

Throughput Analysis: It concluded from throughput that at host1 (client) and host2 (server) Bellman-Ford algorithm has high throughput as compared to all four algorithms for equal edge weights 10 Megabits/sec and for different weights that were assigned at edges. But opposite results were obtained i.e less throughput for Bellman-Ford algorithm as compared to all algorithms for the cases when edge weight equals 1 Megabit/sec for both all link up and single link down. It is because relaxation is easy for weights equal to one. For the 7 switches model, extended Dijkstra's and Dijkstra's algorithms have less throughput which is not a good sign for performance analysis of network routing. The significance is higher throughput is better which occurred for Bellman-Ford and Floyd Warshall algorithms.

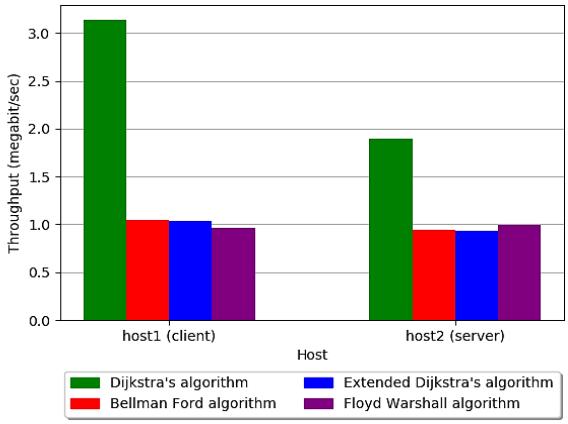

Figure 10. Throughput for Equal Weight 1 Megabit/Sec (all link up)

4.8 Validation Results Analysis

As compared to ETSI standard table, it concluded that the values obtained from experiments are all between 0 to 150 delays (ms). It means all algorithms are in a good category. But Bellman Ford delay in all cases is less than the remaining three algorithms which means Bellman-Ford suits well within the SDN network. Similarly, for the jitter standard on ETSI, all algorithms are in a very good category in between 0 to 74 jitter (ms). But Bellman-Ford has less jitter than other algorithms.

According to ETSI, in standard packet loss there is less packet loss on all algorithms. No packet loss or 0% means that their algorithms are not in a good category. As compared to each other, Bellman-Ford has very less packet loss between all algorithms in the SDN network. For packet transmitted (RTT versus packet sizes measurement), TCP bandwidth and throughput, these are directly related and proportional to delay, jitter, and packet loss standard hence concluded that in average for both 5 and 7 switches SDN model, Bellman-Ford algorithm is better in performance analysis than Dijkstra's, extended Dijkstra's and Floyd Warshall algorithms. In the above analysis of graphs, figures are for 5 switches model topology only. Similarly for the 7 switches model, Bellman-Ford algorithm has less packet loss, delay, latency, and jitter. Similarly, throughput and TCP bandwidth is much better for Bellman-Ford algorithms than other remaining algorithms.

5. Conclusion

The software-defined networking model having 5 and 7 switches network models have been implemented in this research. Three different network topologies have been defined and built separately. Equal edge weights equal to 1 Megabit/sec, 10 Megabits/sec and different edge (6, 8, 10 and 100) Megabits/sec weights are assigned at separate network topologies. All the links of topologies are up at equal and different weights cases and the single link are down in each case i.e. at equal weights and different cases individually. In this research, the Dijkstra's, Bellman-Ford, Extended Dijkstra's and Floyd Warshall algorithms are implemented in a software defined networking environment using Mininet and POX controller to demonstrate the dynamic programmability feature using controller. All four algorithms are compared and analyzed with each other by taking various parameters such as delay, jitter, latency, packet loss, and packet transmit and receive, TCP bandwidth and throughput. As shown by the comparisons and analysis of the results of discussions of all algorithms, the Bellman-Ford algorithm performs better than the other remaining three algorithms. Hence, no need to configure every SDN switch. The only requirement is to configure and program the controller for SDN routing by using algorithms.

References

[1] M. S. Shahriar, J. H. Sabit, A. Ahmed, and others, "Comparative performance analysis of SDN vs traditional networks using SDN controller," PhD diss.Brac University, 2023.

[2] M. Chiesa, A. Kamisiński, J. Rak, G. Rétvári, and S. Schmid, "A survey of fast-recovery mechanisms in packet-switched networks," IEEE Commun. Surv. \& Tutorials, vol. 23, no. 2, pp. 1253–1301, 2021.

[3] L. Zhu et al., "SDN controllers: A comprehensive analysis and performance evaluation study," ACM Comput. Surv., vol. 53, no. 6, pp. 1–40, 2020.

[4] S. Jain and others, "Performance Evaluation of Multi Hop Routing Using Software Defined Network," in 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018, pp. 1–5.

[5] H. M. Noman and M. N. Jasim, "Pox controller and open flow performance evaluation in software defined networks (sdn) using mininet emulator," in IOP conference series: materials science and engineering, 2020, p. 12102.

[6] A. Rego, S. Sendra, J. M. Jimenez, and J. Lloret, "OSPF routing protocol performance in Software Defined Networks," in 2017 Fourth International Conference on Software Defined Systems (SDS), 2017, pp. 131–136.

[7] M. C. AMMOUR and F. DEBBAKH, "PERFORMANCE EVALUATION OF SOFTWARE DEFINED--NETWOEK (SDN) CONTROLLER," PhD diss.UNIVERSITY OF OUARGLA.

[8] M. Doshi and A. Kamdar, "Multi-constraint QoS disjoint multipath routing in SDN," in 2018 Moscow Workshop on Electronic and Networking Technologies (MWENT), 2018, pp. 1–5.

[9] E.-F. Bonfoh, S. Medjiah, and C. Chassot, "A parsimonious monitoring approach for link bandwidth estimation within SDN-based networks," in 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), 2018, pp. 512–516.

[10] A. Casado, S. Bermudo, A. D. López-Sánchez, and J. Sánchez-Oro, "An iterated greedy algorithm for finding the minimum dominating set in graphs," Math. Comput. Simul., vol. 207, pp. 41–58, 2023.

[11] N. Farrugia, V. Buttigieg, and J. A. Briffa, "A globally optimised multipath routing algorithm using SDN," in 2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), 2018, pp. 1–8.

[12] R. Amin, E. Rojas, A. Aqdus, S. Ramzan, D. Casillas-Perez, and J. M. Arco, "A survey on machine learning techniques for routing optimization in SDN," IEEE Access, vol. 9, pp. 104582–104611, 2021.

[13] L. El-Garoui, S. Pierre, and S. Chamberland, "A new SDN-based routing protocol for improving delay in smart city environments," Smart Cities, vol. 3, no. 3, pp. 1004–1021, 2020.

[14] A. S. Nugroho, Y. D. Safitri, and T. A. Setyawan, "Comparison analysis of software defined network and OSPF protocol using virtual media," in 2017 IEEE International Conference on Communication, Networks and Satellite (Comnetsat), 2017, pp. 106–111.

[15] Liatifis, P. Sarigiannidis, V. Argyriou, and T. Lagkas, “Advancing sdn from openflow to p4: A survey,” ACM Comput. Surv., vol. 55, no. 9, pp. 1–37, 2023.

[16] Isyaku, M. S. Mohd Zahid, M. Bte Kamat, K. Abu Bakar, and F. A. Ghaleb, “Software defined networking flow table management of openflow switches performance and security challenges: A survey,” Futur. Internet, vol. 12, no. 9, p. 147, 2020.

Author's biography

Roshani Ghimire is currently an Asst. Professor at Advance College of Engineering and Management, Institute of Engineering, Tribhuvan University, Kathmandu, Nepal. She completed her bachelor of computer engineering in 2010 and MSc Information System Engineering in 2014. Her area of research is networking and security.

Ram Kumar Basnet has completed his bachelor’s degree in Computer Engineering in 2006 AD from Kantipur Engineering College affiliated with Tribhuvan University, Nepal and Master’s Degree in Computer Engineering with Distinction in 2020 AD from Nepal College of Information Technology (NCIT) affiliated with Pokhara University. Additionally, He has also completed a Master's Degree in Business Studies in 2012 AD from Shanker Dev Campus affiliated with Tribhuvan University, Nepal.