Computers, Materials & Continua
DOI:10.32604/cmc.2022.018211
Article

Tech Science Press

Artificial Intelligence Based Reliable Load Balancing Framework in Software-Defined Networks

Mohammad Riyaz Belgaum[1], Fuead Ali[1], Zainab Alansari[2], Shahrulniza Musa[1,*], Muhammad Mansoor Alam[1,3] and M. S. Mazliham[4]

[1]Malaysian Institute of Information Technology, Universiti Kuala Lumpur, Kuala Lumpur, 50250, Malaysia
[2]Majan University College, Muscat, 710, Oman
[3]Institute of Business Management, Karachi, Pakistan
[4]Malaysia France Institute, Universiti Kuala Lumpur, Kuala Lumpur, 43650, Malaysia
[*]Corresponding Author: Shahrulniza Musa. Email: shahrulniza@unikl.edu.my
Received: 01 March 2021; Accepted: 02 April 2021

Abstract: Software-defined networking (SDN) plays a critical role in transforming networking from traditional to intelligent networking. The increasing demand for services from cloud users has increased the load on the network. An efficient system must handle various loads and increasing needs representing the relationships and dependence of businesses on automated measurement systems and guarantee the quality of service (QoS). The multiple paths from source to destination give a scope to select an optimal path by maintaining an equilibrium of load using some best algorithms. Moreover, the requests need to be transferred to reliable network elements. To address SDN's current and future challenges, there is a need to know how artificial intelligence (AI) optimization techniques can efficiently balance the load. This study aims to explore two artificial intelligence optimization techniques, namely Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO), used for load balancing in SDN. Further, we identified that a modification to the existing optimization technique could improve the performance by using a reliable link and node to form the path to reach the target node and improve load balancing. Finally, we propose a conceptual framework for SDN futurology by evaluating node and link reliability, which can balance the load efficiently and improve QoS in SDN.

Keywords: Ant colony optimization; load balancing; particle swarm optimization; quality of service; reliability; software-defined networking

1 Introduction

Big data, cloud computing, and IoT have contributed to massively increased traffic on the traditional networks as these networks are generally non-programmable [1]. The traditional networks have a closely connected control plane and data plane and are complex to manage, incurs high operating expenses for maintenance. Software-Defined Networking (SDN) emerged as a modern and innovative networking model, providing programmability with ease of network

control and management for the future internet [2]. SDN virtualizes network and access to the basic bearers [3]. SDN's virtualization feature empowers multiple routes and better approaches to share the bearers to fulfil client needs at a lower cost. And the clients ensure that the service level agreements guarantee to provide the subscribed applications by the service providers. To ensure QoS, better approaches to evaluate their prerequisites and to quantify the conveyed administration are needed. Fig. 1 illustrates the SDN architecture, which is adapted from [4].

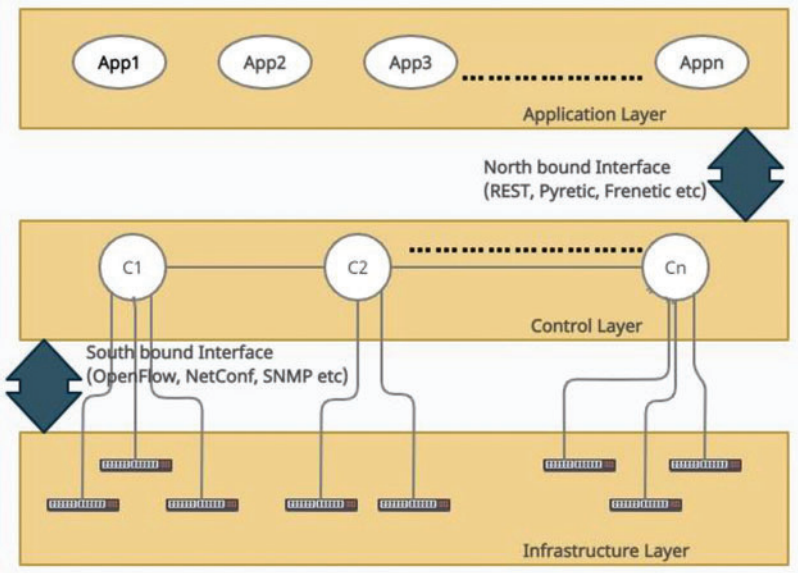

Figure 1: SDN architecture

One of the main features of SDN is that it offers a more open forum for applications to develop, keeping the data plane to receive data from the clients and forward the data to the control plane to meet the dynamic user requirements [5]. In SDN, the centralized control plane handles the network dynamically and summaries the underlying network infrastructure to the applications. The northbound interface provides a means of communication between the SDN controller at the control layer and the application layer components to provide and implement application-level specifications. The southbound interface allows communication between the SDN controller and the switches at the infrastructure layer by the generation of flow tables. The flow tables generated and delegated in the OpenFlow switches direct them for effective and efficient service. The SDN controller has a centralized control and view over the network topology and is responsible for the efficient delivery of traffic in SDN and incorporates a versatile network setup and network security features.

The need for the services offered by the cloud has risen significantly due to the recent innovations in the digital environment [6]. But the design and implementation of intelligent systems for future networking (using 5G and IoT) supporting communication between heterogeneous devices are thus highly limited [7]. This includes high traffic volume, accessibility, and trustworthiness. By the year 2030, the volume of traffic is projected to be 20,000 times higher than that in the year 2010 [8]. This incredibly high volume of traffic cannot be controlled by the existing networking and, therefore, requires new technologies and paradigms. SDN and other technologies, such as Mobile Edge Computing (MEC) and Network Function Virtualization (NFV), are used by 5G

and next-generation networks to meet the user requirements and manage the sizeable amount of data traffic. Load balancing is very challenging and one of the top concerns in SDN, as the incoming requests keep fluctuating with the continuously changing user requirements. Load balancing is the process of handling the requests efficiently by using an optimized path and satisfying the QoS requirements. Load balancing aims to handle the incoming requests efficiently and improve QoS [9].

The benefits of effective load balancing in SDN are as follows:

- The expenditure to manage the network is minimized.
- Increased connectivity of heterogeneous vendor-free devices.
- Increased reliability
- Ease of configuration with customized programming
- Dynamic and flexible network topology
- Ability to automate
- Improved QoS.

The requests are generated from the hosts which are connected to the switches, and the packets can be transmitted between any pair of source and destination hosts. The switches in the infrastructure layer of SDN verifies whether there is a path in the flow table to forward the request to the destination. If the flow table entry is missing in the flow table, then the request is forwarded to the controller in the control plane as a packet_in message. Because several requests are coming from different hosts, the controller aims to reduce the latency in transmitting the packets. The controller uses a load balancing algorithm and decides which path is optimal for the request. The nature of the request assignment problem is NP-Hard as the number of requests dynamically changes, and the length of request processing also varies. Efficient handling of the requests improves network performance. When the latency incurred on transmitting the packets is minimized, it reduces the load on the controller, and thus, the load is balanced. Therefore, an efficient load balancing algorithm can handle the requests by improving the QoS in the network. Many solutions involving heuristic and meta-heuristic approaches can handle such NP-Hard problems. Currently, swarm intelligence techniques are providing better solutions [10]. In this paper, two of the well-known swarm intelligence algorithms, namely, Particle Swarm Optimization (PSO) [11] and Ant Colony Optimization (ACO) [12], have been considered for the study, along with the round-robin technique used for comparison. Many modified versions of various AI algorithms are proposed to balance the load in different deployments, yet the reliability as a constraint to select the node on the optimal path is missing.

To summarize, the elements discussed in this paper are as follows:

- We compared and analyzed three load balancing techniques in SDN.
- We proposed a framework to ensure reliability and enhance QoS in SDN.
- To choose the reliable switch, the proposed reliability verification process is done based on direct information and indirect information considering node reliability and link reliability for better load balancing.

The remaining part of the paper is organized as follows: A related review that discusses similar works on performing load balancing in SDN is done in Section 2. The load balancing in SDN is discussed in Section 3. Section 4 presents the method used for comparative analysis. Section 5 presents the results and discussion, along with the problem of reliability. Section 6 puts

forward a proposed framework and builds a roadmap for future research with a reliable load balancing in SDN. Finally, Section 7 outlines the conclusion and future work.

2 Related Review

The similar works of other researchers that contribute to this paper are described as follows:

Load balancing with controller deployment in the network is very crucial in SDN. The authors in [13] used the affinity propagation algorithm based on PSO for balancing the load in SDN by considering the controller deployment in the multi-domain SDN. The subdomains in the network reassign the switches following the breadth-first search technique for better controller load balancing. The proposed algorithm on comparison out rules the affinity propagation and genetic algorithms in terms of the number of switch migrations, controller load balancing rate, processing time, response time, and control traffic rate. However, the other QoS metrics were not considered in the comparative analysis. Most of the algorithms consider the static load balancing in the network, while the authors in [14] used a dynamic load balancing strategy. The drawback of the PSO algorithm having a poor global search was improved by adding dynamic parameters to have an efficient controller deployment showing a reduced delay between the controller-switch pair and improving the load balancing rate. The reliability of the controller-switch pair, however, remains an area to be explored. A modified PSO used in [15] integrated SDN with fog computing to address the load balancing issue. The constraints of PSO such as the velocity, position, and inertia weight were modified, and the latency was reduced. The reduced latency, in turn, increased the capacity to efficiently handle more load.

Traffic in the network adds more load. The authors in [16] classified the traffic as multimedia and non-multimedia traffic based on bandwidth requirements. The node followed the path with minimum bandwidth consumption in the previous transmission. The proposed routing protocol based on ACO showed a better QoS in comparison with that of PSO. Efficient handling of load remains a concern, since in real-time, there is always a mix of different types of loads from different sources. The resources in the network must be utilized efficiently and ensure that they are not overloaded. The ACO enables us to find the optimal path, as well as the genetic algorithm, which helps in the fast convergence of the global search. A combination of these two techniques was used in [17] to reduce the convergence latency and improve the search criteria. The packet loss rate and the round-trip time were reduced, but the dependency on these parameters was ignored. The evaluation of dependency on the QoS parameters helps to get a reliable path and a reliable node to use in communication. The controller placement problem has been an issue related to load balancing in SDN. If the number of switches managed by each controller exceeds the limit, then the controller gets overloaded. And when the controller is overloaded then, it affects the QoS metrics. The authors in [18] have added external memory to the ant colony system to solve the controller placement problem. The results obtained were compared with those of PSO to verify the effectiveness. The device-to-device communications were evaluated by placing the SDN controller in an appropriate location, and the number of controllers needed to manage the load. Similarly, in [19], the authors have proposed a modification to the ant colony system algorithm by using the knowledge from the performance of SDN controllers to assign the paths for the flow requests. The efficient assignment of the paths considering the resources improves the QoS.

3 Load Balancing in Software-Defined Networking

This section explains how load balancing is done in SDN. The hosts are connected to the switches, and the switches are connected to the controller. The concept of SDN has moved the

decision-making part to the controller, and the switches are only forwarding devices based on the flow rules injected by the controller. The controller decides, based on the technique deployed in it to which switch the load must be transmitted. Load balancing in SDN can be done in two scenarios. The first one is uni-controller deployment, as shown in Fig. 2, and the other one is multi-controller deployment, as shown in Fig. 3.

3.1 Load Balancing in Uni-Controller Deployment

Consider the uni-controller deployment shown in Fig. 2, a single controller manages a group of switches within a domain. Initially, when a source host sends a packet to a switch (S1), the switch explores its flow table for any matching rule to find a path to reach the destination host. If available, then it uses that path to reach the destination. If no matching flow rule is found, then a request as packet_in message is sent to the controller ($C1$). The interface used for communication between the switch and controller is the southbound interface. There are three paths to reach the destination as $S1, S2, S3, S5$. The second one is $S1, S2, S4, S5$ and the third path is $S1, S2, S5$. Now the controller utilizes the technique programmed in it and forwards a message as packet_out with a flow rule to S1 such that every path is equally balanced. A problem arises when the controller itself is overloaded with a greater number of switches to handle. This can be observed when there is an increase in latency because of the increased requests generated at a switch, which sends a packet_in message to the controller. Therefore, a simple solution is to reduce the latency and avoid overload on the controller by using an efficient and reliable optimized path. Another major problem with uni-controller deployment is having a single point of failure (SPF). And to handle this problem, the solution is to have a multi-controller deployment where another controller takes over the load of the failed controller.

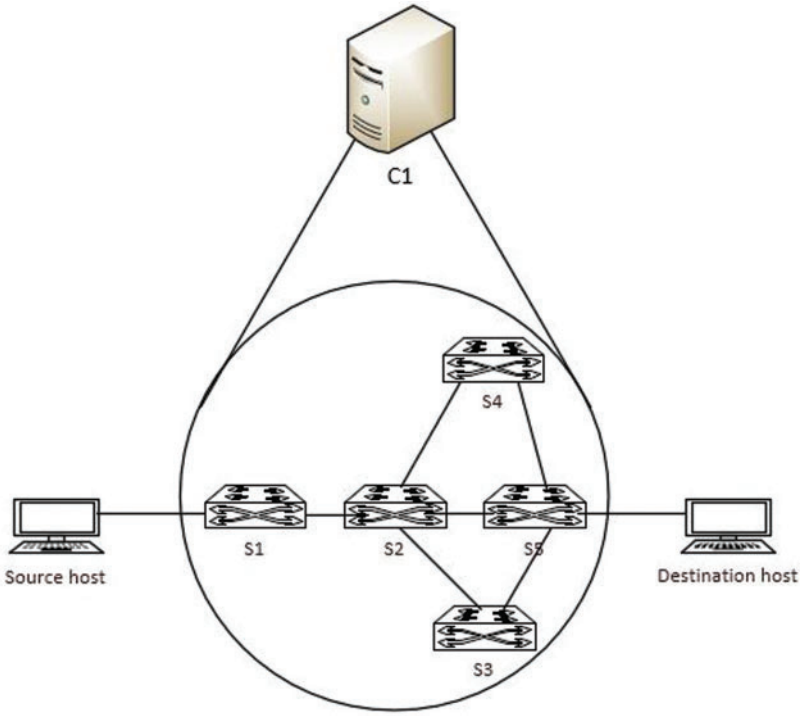

Figure 2: SDN uni-controller deployment

3.2 Load Balancing in Multi-Controller Deployment

The multi-controller deployment shown in Fig. 3, has switches in each domain controlled by a slave controller. These slave controllers of all the domains are in turn controlled by a master controller to avoid the problem of SPF of the uni-controller deployment. When any of the slave controllers is overloaded, then the master controller can shift the load to any other idle slave controller to balance the load. This is done by migrating the switch from an overloaded controller to an idle or low-loaded controller as discussed in [20]. However, identifying the idle or low-loaded controller is an issue as the traffic cannot be predicted. In that case, a lot of time will be spent only on either identifying the low-loaded controller or migrating the load from one device to another.

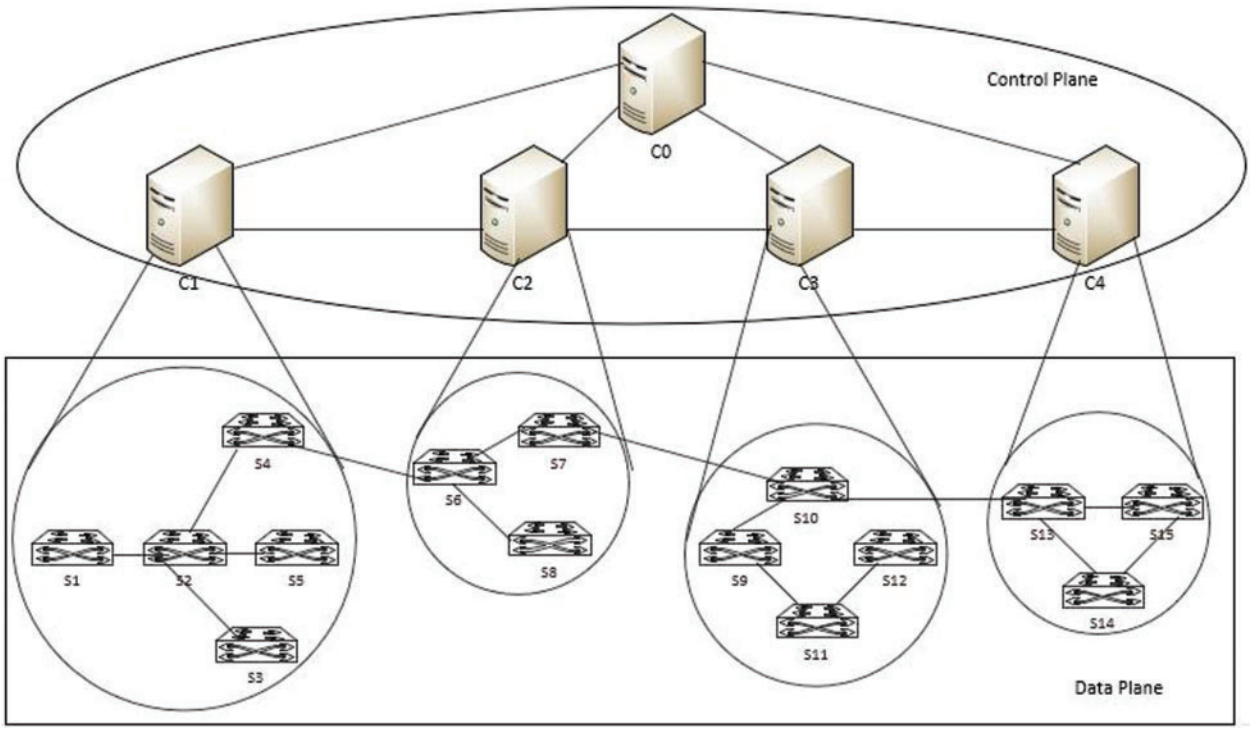

Figure 3: SDN multi-controller deployment

In both the above scenarios, priority has been given to migrating the load to the idle device. But the issue of reliability [21] was not considered in any case.

4 Method

The swarm intelligence algorithms have been applied in many fields, including possible solutions to optimization problems using the shortest path. Apart from the round-robin technique, the two optimization algorithms considered in this study are PSO and ACO, which are discussed below, following a step-by-step process.

4.1 Particle Swarm Optimization

PSO is one of the nature-inspired meta-heuristic algorithms proposed by Kennedy and Eberhart and is used to solve optimization problems. Considering the candidate solutions, which are also known as particles, the PSO algorithm improves the candidate solution and derives the optimal solution. The particle's movement is influenced by its local best-known position within the search space. The following are the steps of the PSO algorithm:

(1) Generation of the population: The first step in the PSO technique is to generate a population of individuals. These individuals are nothing, but the packets transmitted from each switch.
(2) Evaluation of fitness function: The fitness function is determined to get the best optimal solution. For each particle, the fitness function is evaluated.
(3) Find pbest and gbest: The fitness function results of the particle are searched to get the pbest value and, to get the gbest value the fitness function results of the swarm are searched.
(4) At first iteration: Initially gbest = pbest at the first iteration $t = 1$ on the calculation of fitness function. Then the fitness value of the particle is compared with that of the next iteration at $t+1$ to get the best solution.
(5) Updating velocity and position: At every next iteration $t+1$, the particle's velocity and position are updated.

$$\mathrm{V}^{\mathrm{t}+1} = (\omega \mathrm{V}^{\mathrm{t}} + \mathrm{c1.r1}\left(\mathrm{pbest} - \mathrm{pos}^{\mathrm{t}}\right) + \mathrm{c2.r2}\left(\mathrm{gbest} - \mathrm{pos}^{\mathrm{t}}\right)) \tag{1}$$

where $r1, r2$ are random numbers between $[0, 1]$

c1, c2 are acceleration constants.

In Eq. (1), the three terms combinedly designate a purpose to update the velocity of the particle. The first term is the velocity at t^{th} iteration using Eq. (2), the second term represents the cognition model and the third as the social model.

$$\omega = \omega_{\max} - \left(\frac{\omega_{\max} - \omega_{\min}}{\mathrm{k}_{\max}}\right) * \mathrm{k} \tag{2}$$

where k_{max} is the maximum number of iterations and k is the current number of iteration,

ω is the inertia weight factor used to have a balance between global and local explorations,

ω diminishes from $0.9 to 0.4$ during the iterations.

$$pos^{t+1} = pos^{t} + V^{t+1} \tag{3}$$

(a) To ensure the gbest value is reached, a comparison of fitness of the particle and swarm is made after every iteration. The minimum fitness value among both is stored in the particle, and the swarm and all the particles update their velocity for the next iteration using Eq. (1) and position value using Eq. (3).
(b) To stop the search, the termination condition is verified. If not met, then repeat step 2.

Stopping criteria: Step 2 to Step 5. a, is repeated until the termination criteria are satisfied. And the criteria can be either the maximum number of iterations or the convergence set.

4.2 Ant Colony Optimization

ACO was proposed by Marco Dorigo in the early 1990s. ACO is a heuristic optimization approach focused on biological systems to solve complex problems combinatory. The initial version of the ACO was called Ant systems. It is inspired by stigmergy, in which the trace of an action done by an organism stimulates subsequent action by the same or other organisms. In ACO, based on the probabilities, the ants are more likely to choose a path with stronger pheromone and accordingly make decisions. The path with a higher pheromone level is chosen. In the ACO, the goal of an ant is to find the shortest path from its nest to the food source.

Let us assume that an ant k is moving from position x to position y. The probability of movement of the ant k from x to y is denoted as P_{xy}^{k}, as shown in Eq. (4). The probability of movement of the ant depends on two different factors: the attractive coefficient $(\eta_{xy}^{(t)})$ and the pheromone $(\tau_{xy}^{(t)})$. Therefore,

$$P_{xy}^{k} = \frac{[\eta_{xy}^{(t)}]^{\alpha}.[\tau_{xy}^{(t)}]^{\beta}}{\sum_{j \in allowed_x}[\eta_{xy}^{(t)}]^{\alpha}.[\tau_{xy}^{(t)}]^{\beta}} \tag{4}$$

where α is a coefficient used to control the factor of $(\tau_{xy}^{(t)})$ and $\alpha \geq 0$

and β is a controlling factor for the attractive coefficient and $\beta \geq 1$.

The pheromone value is calculated, as shown in Eqs. (5) and (6).

$$\tau_{xy} \leftarrow (1-\rho).\tau_{xy} + \Delta\tau_{xy}^{(k)} \tag{5}$$

where ρ is the coefficient of vaporization of pheromone.

$$\Delta\tau_{xy}^{(k)} = \begin{cases} \frac{Q}{L_k} & \textit{where } Q \textit{ is constant and } L_k \textit{ is distance travelled by } k^{th} \textit{ ant from } x \textit{ to } y \\ 0, & \textit{otherwise} \end{cases} \tag{6}$$

4.3 Network Topology and Experimental Setup

For our analysis, we have considered a uni-controller deployment in our experiment. The reason for considering the study of uni-controller deployment for load balancing is that if the load is efficiently balanced in uni-controller, the requests will not be forwarded to the master controller and reduces the overhead. Moreover, the latency to handle the number of requests reduces, thereby increasing the QoS. Let us consider the following network topology, as shown in Fig. 4, having six switches and eight hosts connected randomly. All the hosts and switches reside in the infrastructure layer, and the controller resides in the control layer of the SDN architecture. The packets can be sent between any pair of source and destination hosts through the switches connecting them. When there is no entry in the flow table to reach a destination host from any source host, the request is sent to the controller as a packet_in message. The controller runs the appropriate algorithm to forward the request to a least loaded optimized path by inserting a flow entry in the flow table of the switches in that path. Let us assume that the source host is $H1$ and the destination host is $H8$, and the packets are sent from source to destination. There are three paths to reach the destination as $S1, S2, S6$. The second path being $S1, S4, S3, S6$ and the third path is $S1, S4, S5, S6$. The parameters used in this experimental environment for simulation are shown in Tab. 1.

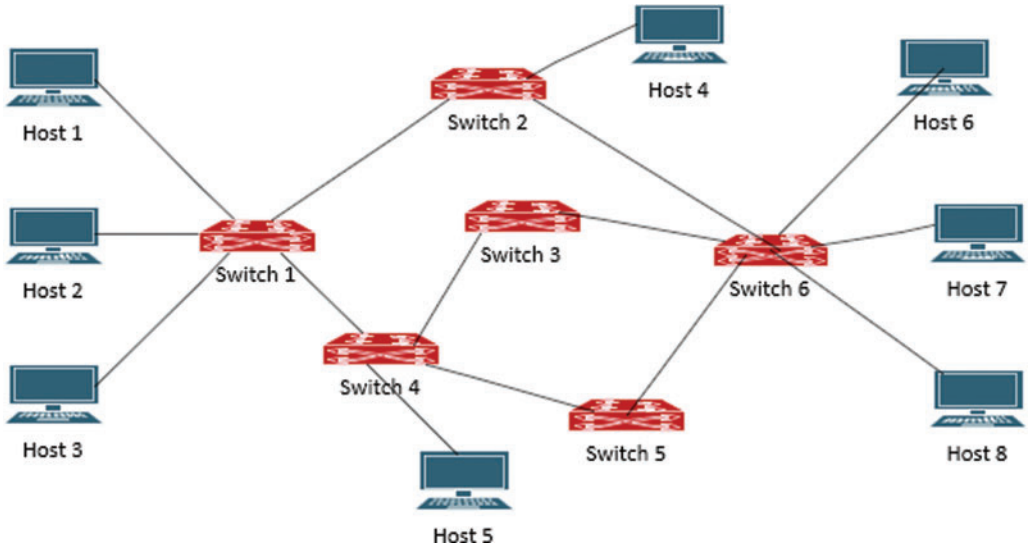

Figure 4: SDN network topology

Table 1: Parameters used in the experimental setup

Parameter	Description
Operating system	Ubuntu 18.04.3
CPU	i7 – 6500
RAM	12 GB
Simulator	Mininet 2.2.1
Controller	Floodlight
Switch	OVS (open virtual switch)
Protocol	OpenFlow V1.3
Traffic generator	D-ITG
Type of load	UDP
Packet size	1024 bytes
Metrics used	Packet loss ratio, round trip time, and latency.

4.4 Flow Sequence Used for Comparative Analysis

Load balancing techniques used in this study for comparison are round-robin, ACO, and PSO. The following sequence is used to extract the results of each of the load balancing techniques.

(1) Create custom topology in Mininet
(2) Configure all hosts, links, and switches
(3) Initialize the network
(4) Set the load balancing technique to be used at the controller
(5) Send the requests from various hosts into the network
(6) Measure the performance of each load balancing technique using metrics
(7) Record all the data for comparative analysis

5 Results and Discussion

To generate the load in the network from different hosts, D-ITG (Distributed Internet Traffic Generator) tool has been used. The packets can be flooded between any source and destination pairs. In our study, though the UDP packets are sent from different hosts, the packets that are flooded from source $H1$ to the destination host is $H8$ are considered. The metrics like latency, packet loss ratio, and round-trip time are used to measure the performance of each load balancing technique in this study.

The latency incurred while transmitting a varying number of packets using different techniques has been shown in Fig. 5. PSO technique has less latency when compared with ACO and round-robin technique. The reason for reduced latency in PSO is due to the quick convergence of the global solution.

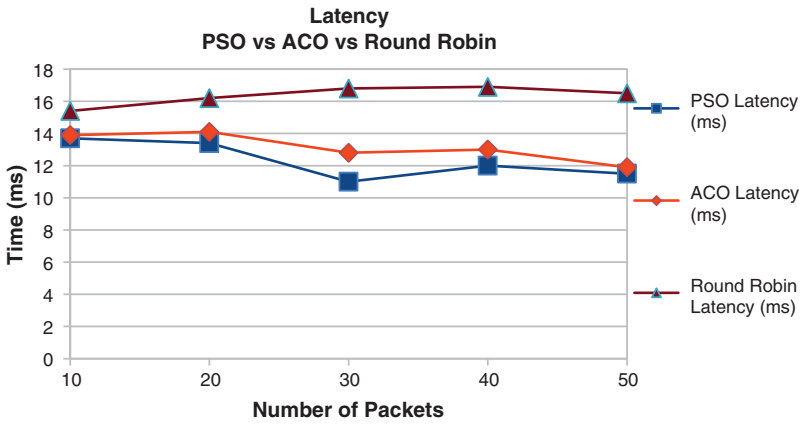

Figure 5: Performance comparison of different techniques in terms of latency

Fig. 6 shows the packet loss ratio of different techniques. The packet loss ratio in the round-robin is higher than ACO and PSO for the varying load.

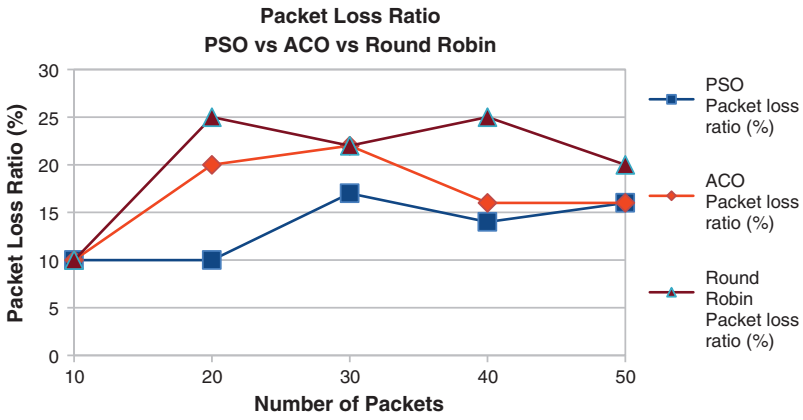

Figure 6: Performance comparison of different techniques in terms of the packet loss ratio

The round-trip time of varying loads using different techniques is depicted in Fig. 7. PSO technique has less round-trip time when compared with ACO and round-robin technique.

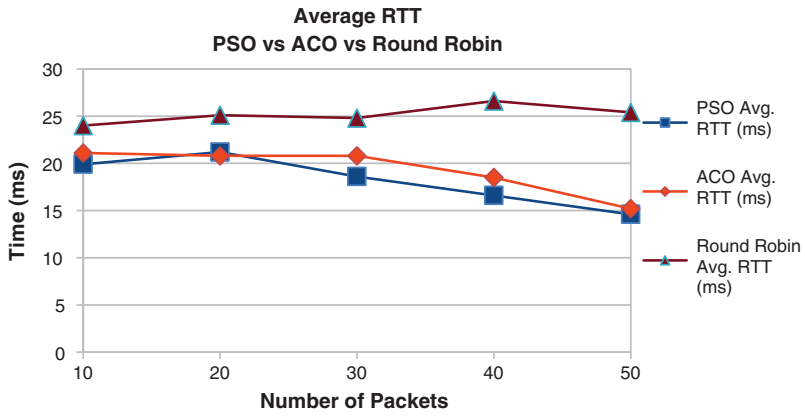

Figure 7: Performance comparison of different techniques in terms of average round- trip time

The PSO algorithm uses path 2 to transmit the packets as it is identified as the optimal path from source $H1$ to the destination host $H8$. And, using the ACO algorithm, the optimal path to reach the destination host $H8$ from source $H1$ is path 2. In both cases, the problem of identification of a reliable node has not been considered. The round-robin technique has been ruled out here, as; it is a conventional load balancing technique, which considers static load. Moreover, the results of the metrics of round-robin, in comparison with PSO and ACO, do not demonstrate better performance. For dynamic load, artificial intelligence load balancing techniques react by considering the individual and social behavior of the neighboring nodes.

The problem of reliability in SDN between the controller and the switches is explained below. The controller (source) is the central source of sending flow entries into the switches (terminals). The connections between the source and terminals are the links, which are denoted by E and are non-correlated. The two states for the links (E) and the nodes (V) are reliable and failure. Following this configuration, let us assume that in a network, there are M controllers and N switches in SDN, such that let $M \leq N$. Let there be a graph $G(V, E)$ such that $N + M = |V|$. We may also use the concept of the graph to formulate the problem of positioning the controller. Assuming that the nodes are randomly connected, there are multiple paths to reach from node u to node v. And the shortest path between them be $p_{u,v}$. The index function: $X_{l,p_{u,v}} = \{0, 1\}$, $X_{n,p_{u,v}} = \{0, 1\}$ is used to verify whether the link l and node n, is on the shortest path or not. If $X_{l,p_{u,v}} = 1$, then the link l is a part of the shortest path, $p_{u,v}$ to node n. Otherwise, $X_{l,p_{u,v}} = 0$, $X_{n,p_{u,v}} = 0$, and the failure is independent. The reliability of link l is denoted by r_l and the reliability of node n is denoted by r_n. Here we have got, $0 < r_l < 1$ and $0 < r_n < 1$. The nodes and links on the optimal path while balancing the load may not be reliable. There is a possibility that a node n and link l, which are on the shortest path to the destination, have less bandwidth utilization ratio or performing with high delay or high packet loss rate. Therefore, while the packets are to be forwarded to the next node, the reliability of the node and link connecting to that node must be evaluated to have a better performance. When a node is reliable to forward the packets, the latency on the controller is minimized. The reduction in latency on the controller enables it to serve more requests, thereby increasing the load balancing capacity. The notations used in this section have been summarized in Tab. 2.

Table 2: Notations

Notation	Description
V	Network node
E	Network link
M	Number of controllers
N	Number of switches
$p_{u,v}$	Shortest path between node u to node v
r_l	Reliability of link l
r_n	Reliability of node n
S_{mj}	j^{th} switch controlled by controller m
S_{mj+1}	$(j+1)^{th}$ switch controlled by controller m
$S_{m(j+1)'}$	$(j+1)'$ are all neighbor switches of $(j+1)^{th}$ switch

Various optimization techniques used to find a path between source and destination consider the only single source of information, i.e., from the node itself, which sometimes is not reliable. Therefore, considering the information from various sources enables one to decide with increased reliability. Hence, the proposed framework ensures better reliability and improves the performance of QoS.

6 Proposed Reliable Load Balancing Framework

The framework proposed here has been done, taking into consideration both the uni-controller and the multi-controller deployments. Fig. 8 shows the proposed framework. The goal of our proposed load balancing framework is to forward the request to a reliable node following the reliable path and reduce the latency. The reduced latency allows the controller to handle more load. Let the controller m be a part of a set such that, $m \in \{1, 2, \ldots, M\}$, and the switches controlled by controller m be $S_m = \{s_{m1}, s_{m2}, \ldots, \}$. The reliability between the source and destination nodes $R_{s,\mathrm{d}}$ can, therefore, be expressed, as shown in Eq. (7), for the optimal path.

$$R_{s,\mathrm{d}} = \prod_{l \varepsilon E} r_l X_{l,p_{u,v}} * \prod_{n \varepsilon V} r_n X_{n,p_{u,v}} \tag{7}$$

A node n, and link l can be parts of the selected path between $p_{u,v}$ if their reliability is high in comparison with the other nodes and links. The node's reliability (r_n) can be evaluated by using the metrics like packet loss rate (PLR), delay (d), throughput (tp). The link's reliability can be evaluated by using the bandwidth utilization ratio (BUR). The values of these metrics are collected from the prior traffic flow results. The reliability of all the links and the nodes forming the optimal path is evaluated using Eq. (7), from the source node to the destination node, with the first expression being the product of all reliable links on the optimal path and the second expression is the product of all reliable nodes on the optimal path. If a packet is transmitted from S_{mj} to S_{mj+1}, then switch S_{mj} collects two types of information from the prior traffic flow results. One is the direct information (D), and the other is indirect information (ID). An aggregate of

direct and indirect information of S_{mj+1} gives the reliability value to S_{mj} as the reliability of the node as r_n, as shown in Eq. (8).

$$r_n = \frac{1}{2}\sum(D, ID) \tag{8}$$

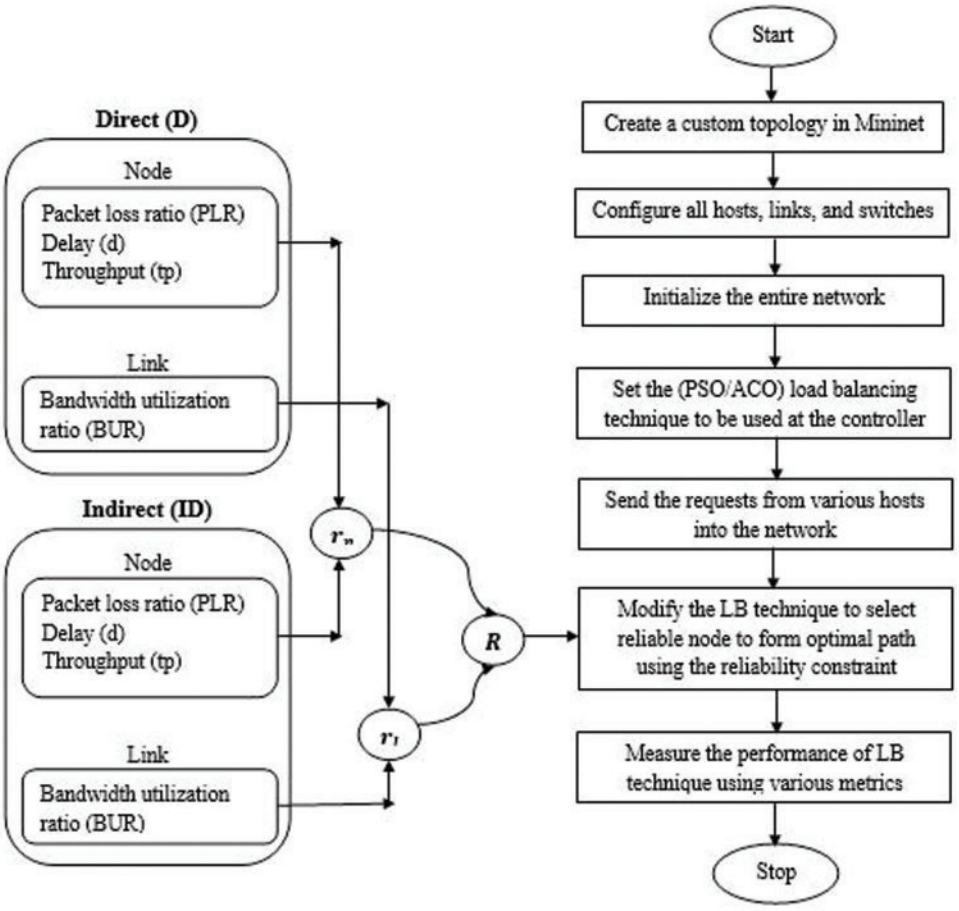

Figure 8: Proposed reliable load balancing framework

Under direct information, the switch S_{mj} gets information from S_{mj+1} about its packet loss rate, delay, and throughput using Eq. (9). And under indirect information, S_{mj} gets information about S_{mj+1} from $S_{m(j+1)'}$ in terms of packet loss rate, delay, and throughput using Eq. (10).

$$D = S_{m(j+1)_{PLR}} * S_{m(j+1)_d} * S_{m(j+1)_{tp}} \tag{9}$$

$$ID = \frac{1}{n}\sum_{j=1}^{n}(S_{m(j+1)'_{PLR}} * S_{m(j+1)'_d} * S_{m(j+1)'_{tp}} \tag{10}$$

In the same way, the link's reliability (r_l) can be calculated as an aggregate of using direct and indirect information of S_{mj+1} in terms of bandwidth utilization using Eq. (11).

$$r_l = \frac{1}{2}\sum(D, ID) \tag{11}$$

Under direct information, the switch S_{mj} gets information from S_{mj+1} about its bandwidth utilization using Eq. (12).

$$D = S_{m(j+1)_{BUR}} \tag{12}$$

And under indirect information, S_{mj} gets information about S_{mj+1} from $S_{m(j+1)'}$ in terms of bandwidth utilization using Eq. (13).

$$ID = S_{m(j+1)'_{BUR}} \tag{13}$$

An aggregate of direct and indirect information of S_{mj+1} gives the reliability value to S_{mj} as the reliability of the link as r_l, as shown in Eq. (11). The index function: $X_{l,p_{u,v}} = \{0, 1\}$, $X_{n,p_{u,v}} = \{0, 1\}$ is used to verify whether the link l and node n, are part of the path or not. Hence, to optimize the network reliability with the shortest path, it can be expressed as follows.

$$\min\left\{1 - \frac{\sum_{m=1}^{M}\sum_{S_{mj\in S_m}} R_{m,S_{i,j}}}{M}\right\} \tag{14}$$

Eq. (14), states the problem of optimization as to consider the location of the controller to maximize the average network reliability between controllers and switches.

Therefore, the function of evaluating reliability can be incorporated in each of the artificial intelligence load balancing algorithms. So, once the reliable node is identified based on the direct and the indirect information as discussed above, the packets can be forwarded to it. On forwarding the packets to a reliable node, a greater number of packets can be transmitted to the destination at full tilt, which in turn reduces the latency. Reduced latency increases the load balancing capabilities of the controller, and it increases the performance in terms of QoS metrics.

7 Conclusion and Future Work

In this paper, we studied three load balancing techniques used in SDN, namely round-robin, PSO, and ACO for UDP load. The performance of each of these techniques has been evaluated and a comparison has been made in terms of latency, packet loss ratio, and average RTT. The results demonstrated that the round-robin technique works on static load, and the artificial intelligence techniques can dynamically mold based on the changes. And the performance of ACO and PSO is better than round-robin in terms of QoS metrics. The latency of ACO and PSO is minimized with the increased load when compared with round-robin. In the same way, the packet loss ratio and the average round trip time also are less in comparison with the round-robin technique. However, the problem of reliability was not addressed in any of the techniques. Though ACO and PSO used path 2 to transmit the packets, there can be other paths too, with greater reliability and optimal path. Therefore, the authors proposed a framework to consider in ACO and PSO algorithms using direct and indirect information from the switches to forward the packets to the next node with improved reliability and enhance the performance in terms of QoS.

In our future work, the proposed framework will be implemented by modifying the ACO and PSO techniques appropriately and measure the load balancing with TCP and UDP load considering a real-time network.

Acknowledgement: The authors are thankful for the support and facilities provided by the Universiti Kuala Lumpur, Malaysia.

Funding Statement: The authors received Excellent Graduate Assistant funding from Universiti Kuala Lumpur for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] W. Rafique, L. Qi, I. Yaqoob, M. Imran, R. ur Rasool *et al.,* "Complementing iot services through software defined networking and edge computing: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1761–1804, 2020.

[2] K. Nisar, E. R. Jimson, M. H. A. Hijazi, I. Welch, R. Hassan *et al.,* "A Survey on the architecture, application, and security of software defined networking," *Internet of Things*, vol. 12, no. 5, pp. 100289, 2020.

[3] K. B. Narayanan, "SDN journey: How SDN brings versatility to 5G networks?," *CSI Transactions on ICT*, vol. 8, no. 1, pp. 57–60, 2020.

[4] L. Zhu, M. M. Karim, K. Sharif, C. Xu, F. Li *et al.,* "SDN controllers: A comprehensive analysis and performance evaluation study," *ACM Computing Surveys*, vol. 53, no. 6, pp. 1–40, 2020.

[5] O. N. Foundation, "SDN architecture ONF TR-502," 2013. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2013/02/TR_SDN_ARCH_1.0_06062014.pdf.

[6] R. El-Haddadeh, "Digital innovation dynamics influence on organisational adoption: The case of cloud computing services," *Information Systems Frontiers*, vol. 22, no. 4, pp. 985–999, 2020.

[7] T. Qiu, N. Chen, K. Li, M. Atiquzzaman and W. Zhao, "How can heterogeneous internet of things build our future: A survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2011–2027, 2018.

[8] D. Jiang and G. Liu, "An overview of 5G requirements," in *5G Mobile Communications*. Cham: Springer, pp. 3–26, 2017.

[9] M. R. Belgaum, S. Musa, M. M. Alam and M. M. Su'ud, "A systematic review of load balancing techniques in software-defined networking," *IEEE Access*, vol. 8, no. 1, pp. 98612–98636, 2020.

[10] M. Elbes, S. Alzubi, T. Kanan, A. Al-Fuqaha and B. Hawashin, "A survey on particle swarm optimization with emphasis on engineering and network applications," *Evolutionary Intelligence*, vol. 12, no. 2, pp. 113–129, 2019.

[11] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. of ICNN'95-Int. Conf. on Neural Networks*, Perth, WA, Australia, IEEE, vol. 4, pp. 1942–1948, 1995.

[12] M. Dorigo and G. Di Caro, "Ant colony optimization: A new meta-heuristic," in *Proc. of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, Washington, DC, USA, IEEE, vol. 2, pp. 1470–1477, 1999.

[13] G. Li, X. Wang and Z. Zhang, "SDN-based load balancing scheme for multi-controller deployment," *IEEE Access*, vol. 7, pp. 39612–39622, 2019.

[14] Y. Li, W. Sun and S. Guan, "A Multi-controller deployment method based on PSO algorithm in SDN environment," in *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conf.*, Chongqing, China, IEEE, vol. 1, pp. 351–355, 2020.

[15] X. He, Z. Ren, C. Shi and J. Fang, "A novel load balancing strategy of software-defined cloud/fog networking in the Internet of Vehicles," *China Communications*, vol. 13, no. Supplement2, pp. 140–149, 2016.

[16] R. Kulkarni and K. Sharma, "QoS-based routing algorithm for Software-defined network using Ant Colony Optimization," in *Int. Conf. on Advances in Computational Intelligence and Informatics*, Hyderabad, India, Springer, pp. 37–45, 2019.

[17] H. Xue, K. T. Kim and H. Y. Youn, "Dynamic load balancing of software-defined networking based on genetic-ant colony optimization," *Sensors*, vol. 19, no. 2, pp. 311, 2019.

[18] Y. P. Llerena and P. R. Gondim, "SDN-controller placement for D2D communications," *IEEE Access*, vol. 7, pp. 169745–169761, 2019.

[19] A. Farshin and S. Sharifian, "A modified knowledge-based ant colony algorithm for virtual machine placement and simultaneous routing of NFV in distributed cloud architecture," *The Journal of Supercomputing*, vol. 75, no. 8, pp. 5520–5550, 2019.

[20] H. Babbar, S. Rani, M. Masud, S. Verma, D. Anand *et al.,* "Load balancing algorithm for migrating switches in software-defined vehicular networks," *Computers, Materials & Continua*, vol. 67, no. 1, pp. 1301–1316, 2021.

[21] Y. R. Chen, A. Rezapour, W. G. Tzeng and S. C. Tsai, "RL-routing: An SDN routing algorithm based on deep reinforcement learning," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 3185–3199, 2020.