Wireless Personal Communications (2021) 121:3451–3472
https://doi.org/10.1007/s11277-021-08886-2

An Enhanced Framework for Traffic Load Balancing and QoS Provisioning in SDN

C. Bharanidharan[1] · S. Indira Gandhi[1] · R. D. Devapriya[1]

Accepted: 8 August 2021 / Published online: 26 August 2021

Abstract
With the recent emergence of advanced network control and flexibility demanded by 5G networks, software defined networks (SDN) are seen as a viable solution. They provide the separation of control and data planes. It has a huge potential for a wide variety of end user applications. Load balancing and QoS provisioning are some of the applications that reap the benefits of the centralized network control that SDN provides. In our work the framework is divided into modules for congestion detection, load balancing and QoS provisioning. The major contributions to the existing literature are the integration of several network quality determining parameters like bandwidth, delay and reliability, efficient node and link disjoint routing and handling a mixture of QoS demands while not ignoring the best effort user traffic. All the results are plotted and suitable comparisons are done. Our framework performs better in elephant flow conditions and handles heavy load well. It also satisfies QoS requirements without starving the non-QoS user traffic.

Keywords Software defined networking · QoS provisioning · Load balancing · Group table · SDN controller

1 Introduction

Software defined networks were initially studied in order to overcome the challenges inherent to traditional networks in terms of configurability and manageability. Traditional means of networking are limited by the number of control points whereas software defined networks are characterized by a single point of control. Hence scalability and reliability are improved. In modern day data centers, the volume of traffic is huge and the networking devices are many. In such a case, centralized control reaps huge benefits to the network administrator. Software defined networking has been a widely

✉ C. Bharanidharan
bharani88.c@gmail.com

S. Indira Gandhi
indira@mitindia.edu

R. D. Devapriya
dahliarichard@gmail.com

[1] Department of Electronics, Madras Institute of Technology, Anna University, Chennai, India

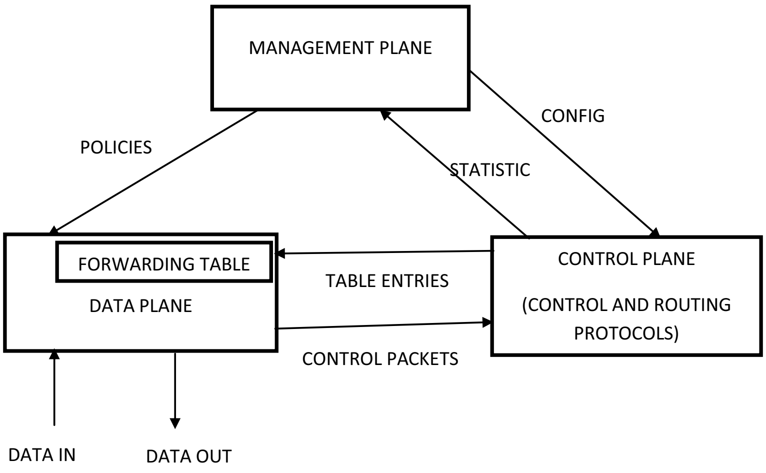


Fig. 1 The planes of networking

researched topic since the progress of information technology. The control and data planes are clearly separated in a software defined network. As detailed by Kreutz et al. [1], the major planes of networking are data, control and management planes. These three are related to each other as depicted in the Fig. 1. The planes of networking and their functions can be described as follows. The data plane is the actual forwarding plane of the switch. It maintains a forwarding table which maintains a list of entries with the source address, destination address and the next hop address. If an incoming packet matches any one of these entries in the table it will be forwarded to the next hop. If the packet doesn't match any of the entries in the table, it is sent to the SDN controller. In addition, control packets like route information packets are also sent to the SDN controller. The actions performed by the data plane include buffering of packets, scheduling, modification of packet headers and forwarding packets. The control plane of networking configures the forwarding table in the data plane. Hence, essentially the control plane acts as the 'brains' for the network finding the best path to a destination and sending route entries to the data plane. In a software defined network, the control plane is centralized in the SDN controller.

The controller receives control packets and packets which don't match any of the entries in the forwarding table from the data plane and sends entries to handle those packets the next time they arrive. Its actions include path finding, route maintenance, periodic statistics report to the management plane and control of forwarding table. The management plane of networking configures global rules to the control plane like the time interval before route updates. It is also configured from a central point since the control plane is centralized. In some implementations, the control and the management planes are integrated. Its functions include setting of global network policies and configuring the same on the control and data planes. The components of a software defined network consist of the networking devices, controller and the protocol or APIs used. In the Fig. 2, the controllers are present in the control plane and the networking devices in the data plane. The terminologies in Fig. 2 were clearly detailed by Kreutz et al. [1]. The roles and functions of each of the components are presented below.

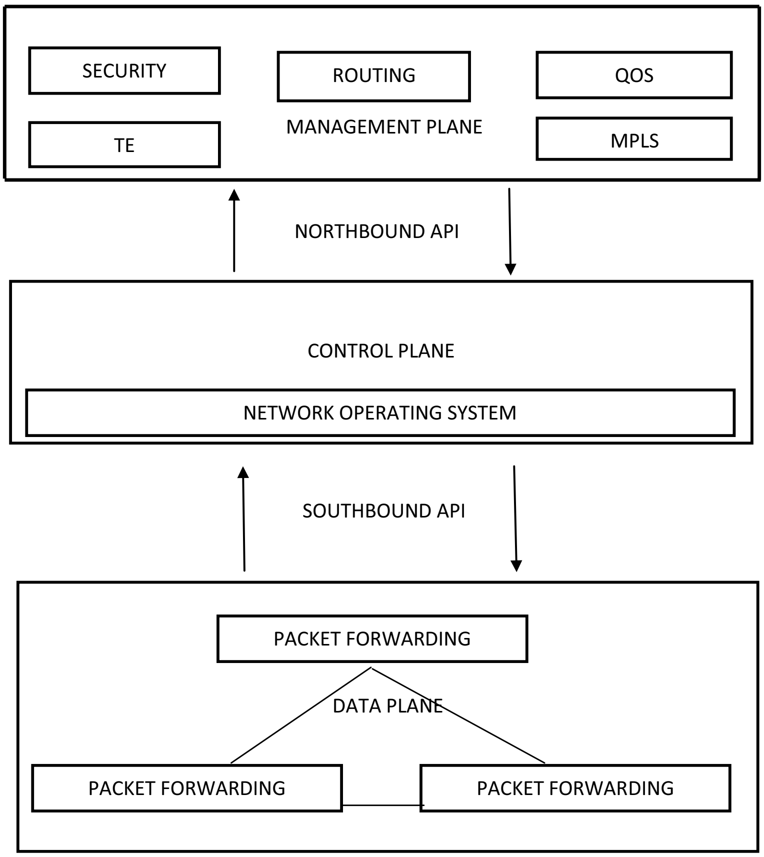


Fig. 2 The components of SDN

- The networking devices include switches and routers which perform the actual task of receiving incoming packets and forwarding them through interfaces or outgoing ports. The role played by the networking devices in a software defined network is packet inspection and deciding whether to drop, forward or modify the packet based on entries in the switch flow table or group table.
- The SDN controller consists of the northbound interfaces, the controller framework and the southbound interfaces. The core controller consists of a topology manager for learning the network, security module for packet filtering, statistics manager for querying switch statistics on a periodic basis and other modules useful to the controller. There are many open source controller frameworks available. A few examples are such as Open Daylight, ONIX, ONOS, POX, NOX, Ryu, etc. Northbound interfaces serve as APIs between the applications like Traffic engineering, security applications, etc. and the core controller framework. The most commonly used northbound interface is the REST API (Fig. 3).

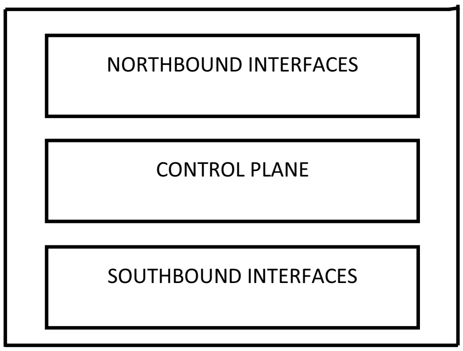


Fig. 3 Components of the controller

Table 1 A flow table

Match criteria			Actions	Priority	Hard timeout (s)	Idle timeout (s)
IP source address	IP destination address	Protocol				
*	10.0.0.1	*	OUT:1	2	20	5
*	10.0.0.2	TCP	OUT:2	1	20	5

Referring the Open flow standard document by the Open Networking Foundation,[1] SDN terminology and its various components can be described as below:

- *Flow*: A flow in SDN comprises of the set of packets with multiple matching criteria like source IP address, destination IP address and the protocol used. Thus each packet is classified to a particular flow and a match is found in the flow table of the switch when the packet arrives at the switch. If no matching entry is found, the packet is sent to the controller.
- *Flow table*: The table that a switch uses to forward the incoming packets belonging to a particular flow is called the flow table. A typical flow table is depicted in Table 1
- *Group table*: An entry in the group table can be used by 2 or more flows in the flow table.

The fields in the flow table can be explained as below.

- *Match*: The match criterion is the set of fields in the packets that are to be compared against the entries in the flow table. In the first entry in Table 1, the '*' denotes that any source IP address is permitted while the destination address must be "10.0.0.1" with any protocol used.
- *Action*: Actions field denotes through which port the packet must be forwarded to reach the destination. For example, from the Table 1, the packets matching the first entry must be forwarded out the port 1.

[1] https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.2.pdf.

Table 2 A group table

Group ID	Action	Out port	Bucket weight
12	SELECT	3, 4	40, 60
34	ALL	3	–

- *Priority*: If a packet matches more than one entry in the flow table, the priority field comes into play. In such a case the flow entry with the highest priority value will be chosen and its corresponding action will be executed.
- *Hard timeout*: The hard timeout value is a timer which specifies the duration for which the corresponding flow entry is valid. For example, a hard timeout value of 20 s indicates that after 20 s, that entry will be flushed. The purpose of this field is that a change in the topology of the network or a link down event must reflect in the flow table as quickly as possible making the switch more adaptive to change and dynamic.
- *Idle timeout*: The idle timeout value indicates that after a specified duration for which that particular flow entry is not used by the switch, it must be flushed by the switch. For example, if an entry is not used by the switch to forward any incoming packet for 20 s and that entry has an idle timeout of 20 s, that entry will be flushed out by the switch.

The fields in a group table are explained as follows.

- *Group ID*: The group ID is an integer value with range up to 65,535. It is used to identify the group entry by the flow table
- *Action*: If “ALL” is present in this field, all the packets belonging to this group will be forwarded along the same output port. If “SELECT” is present, the packets will be divided along 2 or more output ports.
- *Bucket weight*: This denotes that fraction of the packets that must be sent through each output port. For example in Table 2, the first entry forwards 40% of the packets through port 3 and 50% of the packets through port 4.

Overall, the main contributions of the current research work are:

- Utilizing the switch features of flow and group tables to provide load balancing as well as congestion detection
- Enhanced backup path finding technique in the case of congestion detection using disjoint nodes and links
- Admission control based on available queues in addition to redundant link capacity and utilization
- A multiple queue model to provide traffic separation among different flows with different QoS demands
- Priority is given to the low priority user only in the case of no traffic from high priority user
- In the case of delay sensitive traffic, bandwidth pre-emption is done where the bandwidth demanding user flows are a bit delayed
- Only one delay sensitive user traffic is routed along one route while the others take alternate paths to reach the destination
- Comparison against existing mechanism to load balance and provide QoS

The rest of the paper is arranged as follows: Sect. 2 presents a literature survey of existing work relating to SDN load balancing, QoS and Security. Section 3 presents a novel load balancing strategy for the SDN. Section 4 presents an efficient QoS provisioning algorithm. Section 5 presents the implementation and simulation results obtained.

2 Related Work

Traffic load balancing forms a vital part of any network handling huge volume of data. In a data center there are a huge volume of high performance servers. Each server is capable of processing up to n requests at any given time. In order to prevent over-burdening of any one server in the data center, the incoming requests must be shared in real time. The methodologies that can be explored in order to load share have been elucidated by Lee et al. [2] where round robin, least connection and shortest delay approaches are explored and studied. Round robin approach periodically selects a server to accept the incoming user request. It does not consider those requests already being served by the server.

The other two mechanisms choose a server based on the number of requests already served by it and also based on the placement of the server. A much more sophisticated approach was studied by Zhang et al. [3] who studied the effect of clustering a pool of servers. Load balancing was then carried out within the cluster. Another interesting approach where latency constrained and latency tolerant flows were distinguished within the network was studied by Chen et al. [4]. Various performance criteria were found to prioritize the latency constrained flows. Bai et al. [5] studied congestion control in server based networks and studied the use of Explicit Congestion Notification (ECN) alerts in order to balance the load. It has delivered good performance in the case of high volume traffic. In the method outlined by Ghorbani et al. [6] the network consisting of n switches undergoes n separate load balancing mechanisms, one at each switch. The load is micro managed and each switch is provided dynamic intelligence on the bytes through each of its own ports. This in turn rejects to an extent the centralized control feature of software defined networks. A technique for handling 'elephant flows', which are high volume flows sent at a relatively short time at once, was studied by Liu et al. [7]. Multiple path allocation is followed by obtaining slices of the heavy traffic, but there is no traffic sensitive allocation and it may fail in cases where every user sends elephant flows. A dynamic traffic monitoring strategy was adopted by Lang et al. [8], wherein the most congested link was computed in each interval of time previously determined and the traffic split through other paths. An oversight will be the non-adoption of the switch group tables in the traffic re-routing strategy. Another similar strategy studied by Lan et al. [9] addressed the issue of short time high traffic flows that may conjure up a congestion scenario wrongfully by adjusting the times during which traffic re-routing is performed. In an improved method studied by Wang et al. [10], the traffic re-routing times are controlled by adopting a moving average method, in addition to group table based re-routing of traffic flows. In [11] Guo et al. have developed a hybrid OSPF and Open flow approach to tackling the heavy load at the central SDN controller. It identifies crucial flows which occupy more bandwidth in the shortest path leading to congestion and reroutes them. However, the disjointness of the links is not considered as criteria while choosing the alternate path. Link disjointness ensures a greater degree of congestion mitigation. In [12] Binh et al. have developed an algorithm for load balancing and ensuring the quality of transmission in wireless mesh networks. The links are chosen based on the blocking probability and the signal to noise ratio which will be offered using a heuristic algorithm. It doesn't however consider classes of traffic to be treated separately but focuses on overall improvement of quality of routing links. Among the load balancing studies done, there is also a deficiency in the efficient study of QoS traffic coexistence with non-QoS traffic in a typical high volume data center.

The literature on QoS provisioning in software defined networks can be summarized as below. Krishna et al. [13] provides guaranteed bandwidth using Open flow queues and meters. It gives more priority to best effort if excess QoS traffic detected. The drawback of this approach is that it does not consider the presence of delay sensitive flows. Kumar et al. [14] aim to reduce the network delay by increasing the bandwidth allocated to the flow and using separate queues. It does not consider the presence of other bandwidth demanding or delay sensitive flows. Oh et al. [15] aim to proactively change paths in the event of congestion or link failure by pre-assigning multiple paths of differing priorities. The drawback is that temporary congestion can be miss-interpreted as link failure. In the failure recovery approach followed by Zheng et al. [16] during link failure switches automatically route packets through pre-installed links in the flow table. The drawback is that priority is on choosing the shortest path rather than least utilized path for primary path. In the approach outlined by Lee et al. [17] high priority flows always given available bandwidth based on requirement. The drawback is that low priority flows experience greater loss if number of high priority users higher. There is no provision to handle them. In the approach by Devapriya et al. [18] a comprehensive strategy for distinguishing normal and QoS traffic hosts is outlined. The need of a comprehensive algorithm that combines load balancing together with bandwidth, delay and reliability constraints demanded by the user is the motivation for our work. Abbou et al. [19] have developed a software defined queuing framework in order to prioritize low, medium and high priority packets. It uses a fixed means of determining what fraction of the link bandwidth is to be used by a high priority flow when a lower priority flow is in need of the same link. This method of determining bandwidth can starve a non-QoS demanding user when it requires the same path. In our proposed approach this situation has been taken into account by directing the non-QoS flow towards multiple paths by following a weighted approach. Our proposed algorithm differs from the existing works in,

1. The allocation of links in the case of congestion where disjointness is considered as the primary factor in choosing an alternate link
2. Traffic classes are grouped as bandwidth intensive, delay sensitive and fault tolerant
3. QoS bandwidth intensive traffic is assumed to coexist with best effort non-QoS traffic. When it does, the best effort flows are routed along multiple paths by making use of switch group tables in order to avoid starvation of these flows.
4. More than one delay sensitive flow is not allowed to coexist in the same link as shortest delay cannot be guaranteed
5. For flows that demand fault tolerance, a proactive approach is taken wherein the path with the minimum link utilization of its entire links is chosen as the primary path and the rest of all the available paths are chosen as the backup paths in the order of their disjointness with the primary path. These paths are stored in the flow table as the fast failover group. When the primary link fails the alternate link will be chosen.

3 Load Balancing Algorithm

The topology under test is shown in Fig. 5. Congestion detection is performed by following the procedure detailed in [18]. Once the network congestion is detected, load balancing is invoked. All the routing algorithms at present favor the shortest path for selection. This causes these links comprising of the shortest path to become over-burdened very quickly thus degrading network performance. All the previous strategies to load balance the traffic do not select disjoint nodes for the computation of backup paths. If the primary and the backup path selected have one or more nodes in common, the load balanced in such a case will not attain the full benefits of the load distribution if not making the congestion worse [18]. Thus a disjoint path selection for load balancing can significantly improve the network performance and overcome congestion. Also, in the case of elephant flows, the over-burdening of a single path is prevented by following this sharing of traffic among multiple paths based on the existing traffic on each path.

1. Compute all the available paths between the traffic source and sink hosts
2. Take the shortest path among them as the primary path Pi and let Ni be the set of nodes or switches in the primary path. Let the other paths be considered as the secondary or alternate paths. Let N denote the set of switches in each alternate path.
3. For each alternate path, calculate the disjoint ratio between the primary path and itself according to the formula given in Eq. (1).

$$Disjoint\ ratio = \frac{\left|N_A - N_P\right|}{\left|N_A\right| - 2} \tag{1}$$

   where N_A is the set of nodes in the alternate path, N_P is the set of nodes in the primary path. $|N_A - N_P|$ gives the number of different nodes between the primary and the alternate path. $|N_A|$ gives the number of nodes in the alternate path.

   A factor of 2 is subtracted from the numerator to exclude the source and the destination nodes which will also be common nodes among the two paths.

   The disjoint ratio has a range of (0, 1) where, the value of 1 indicates highest level of disjointness.
4. Select the alternate path with the highest value of disjoint ratio with the primary path among all the available paths as the backup path.

The procedure is explained through a flowchart in Fig. 4

4 QoS Provisioning Algorithm

Users in a service provider network demand different classes of service for their different needs. Hence routing must be QoS sensitive. Generally the user preference is indicated in the Type of Service (TOS) bits of the IPV4 header. A contract is made between the users in a network and the service provider regarding the bandwidth or delay constraint to be met for each class of service. Depending on the request the flow is routed. The algorithm is presented below. The overall flowchart is presented in the Fig. 5. In addition to bandwidth and delay constraints, some users may also demand reliability in the case of link failures.

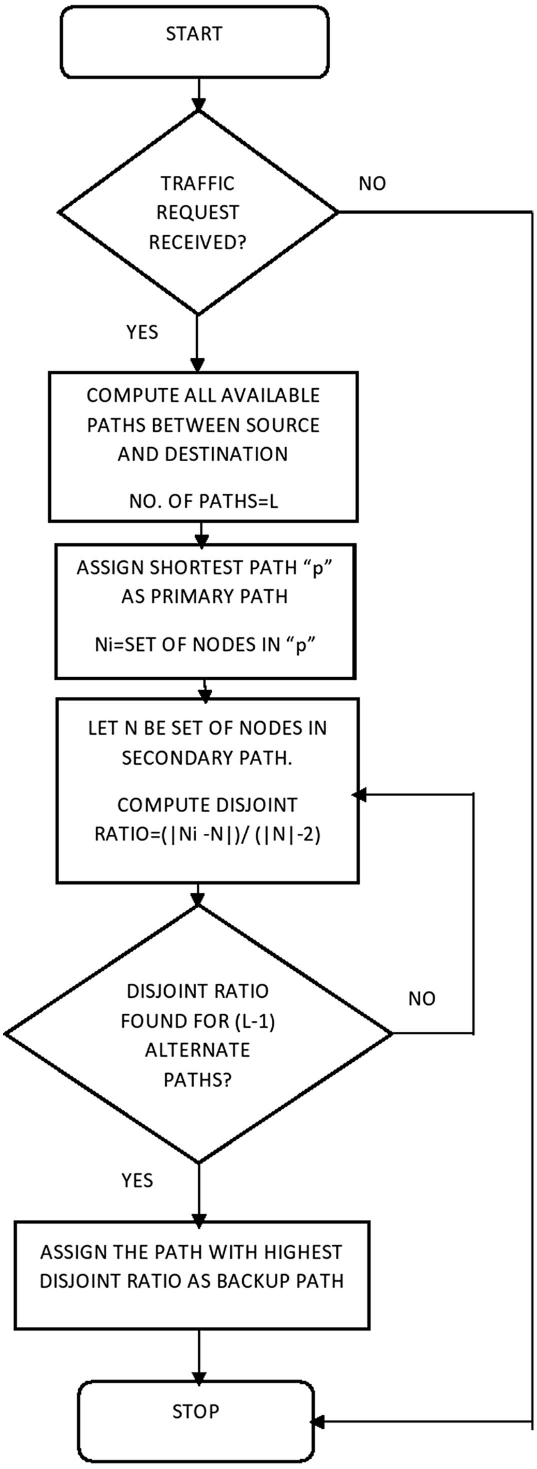


Fig. 4 Flowchart for load balancing algorithm

START
TRAFFIC REQUEST RECEIVED?
NO
YES
COMPUTE ALL AVAILABLE PATHS BETWEEN SOURCE AND DESTINATION
NO. OF PATHS=L
ASSIGN SHORTEST PATH “p” AS PRIMARY PATH
Ni=SET OF NODES IN “p”
LET N BE SET OF NODES IN SECONDARY PATH.
COMPUTE DISJOINT RATIO=(|Ni -N|)/ (|N|-2)
DISJOINT RATIO FOUND FOR (L-1) ALTERNATE PATHS?
NO
YES
ASSIGN THE PATH WITH HIGHEST DISJOINT RATIO AS BACKUP PATH
STOP

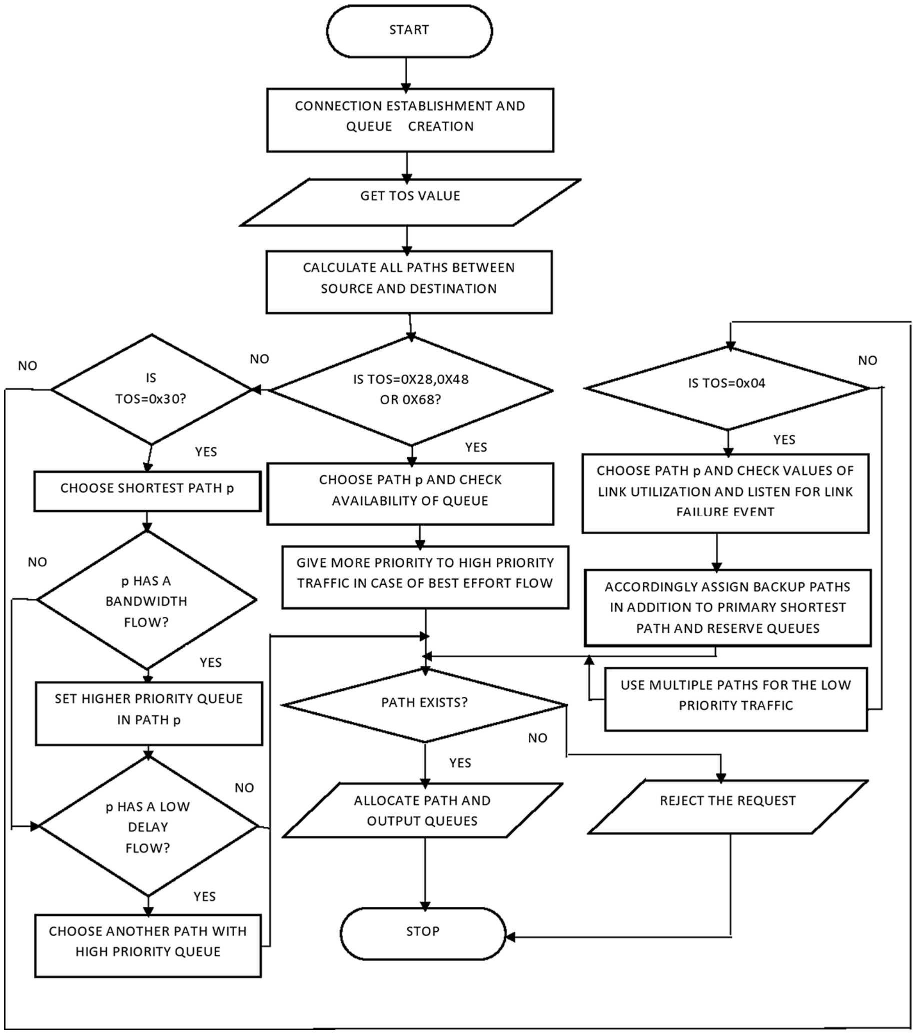

START
CONNECTION ESTABLISHMENT AND QUEUE CREATION
GET TOS VALUE
CALCULATE ALL PATHS BETWEEN SOURCE AND DESTINATION
IS TOS=0x30?
IS TOS=0X28,0X48 OR 0X68?
IS TOS=0x04
NO
YES
CHOOSE SHORTEST PATH p
CHOOSE PATH p AND CHECK AVAILABILITY OF QUEUE
CHOOSE PATH p AND CHECK VALUES OF LINK UTILIZATION AND LISTEN FOR LINK FAILURE EVENT
p HAS A BANDWIDTH FLOW?
GIVE MORE PRIORITY TO HIGH PRIORITY TRAFFIC IN CASE OF BEST EFFORT FLOW
ACCORDINGLY ASSIGN BACKUP PATHS IN ADDITION TO PRIMARY SHORTEST PATH AND RESERVE QUEUES
SET HIGHER PRIORITY QUEUE IN PATH p
PATH EXISTS?
USE MULTIPLE PATHS FOR THE LOW PRIORITY TRAFFIC
p HAS A LOW DELAY FLOW?
ALLOCATE PATH AND OUTPUT QUEUES
REJECT THE REQUEST
CHOOSE ANOTHER PATH WITH HIGH PRIORITY QUEUE
STOP

Fig. 5 Overall flowchart of QoS provisioning

The overall QoS provisioning algorithm is outlined in this section. It involves the following steps.

1. Connection is established between the hosts and the SDN controller. Each host is associated with one or more switches in the network which are connected to the central SDN controller through wired links. Connection between hosts and controller is established by LLDP packets exchanged between the hosts and controller through the switches.
2. Queues are created for each interface of each switch in the network (4 queues each for AF11, AF21, AF31, and AF12). AF represents the Assured Forwarding class of service.
3. The TOS field of incoming IPV4 packet is examined and the QoS request is processed by the controller i.e., the controller identifies the exact service to be provided by decoding the packet.

4. Utilizing depth first search algorithm, all the paths between the traffic source and sink hosts are computed. If bandwidth or delay sensitive service is required for the packet, go to step 5. If reliability or fault tolerance is required, go to step 6.
5. For each path ‘p’:

If TOS = 0×28 OR 0×48 or 0×68 (Bandwidth demanding service):

Choose path p if all links have an empty queue corresponding to AF11, AF21 or AF31 respectively (TOS value of 0×28 indicates AF11, 0×48 indicates AF21, 0×68 indicates AF31 class of service respectively). If Best effort flow coexists, give more priority to QoS traffic by setting the priority parameter of the Queue settings.

If TOS = 0×30 (Delay sensitive service corresponding to AF12 class of service):

(i) If best effort flow coexists in path, choose path p if all links have an empty queue corresponding to AF12.
(ii) If bandwidth sensitive flow coexists in path, assign higher priority to the delay Sensitive flow for pre-emption over the bandwidth sensitive flow.
(iii) If another delay sensitive flow coexists, choose alternate path.

6. For each path ‘p’:

Listen for link failure messages and compute link utilizations:

$$\text{Link utilization of link, } L_U = \frac{A_{i,j}}{C_{i,j}} \tag{2}$$

where $A_{i,j}$ is the number of bytes sent through the link U; $C_{i,j}$ is the link capacity of link U; i and j are the edge nodes of the link.

Add the link utilization parameters of each link of the path and continue this process for all paths and assign path with minimum link utilization as the primary path.

For each path ‘p’:

Assign up to ‘n’ disjoint alternate or backup paths for the incoming request from flow in addition to the primary path already chosen.

7. After assigning path, the remaining number of queues in each interface is updated and the path-finding process for next request is continued.
8. If no path is found which satisfies the constraints, the QoS request is rejected.

The above algorithm is depicted in Fig. 5 as a flowchart. In the subsequent sections, the bandwidth, delay and fault-tolerant service is explained further.

4.1 Bandwidth Guarantees

A multiple queue approach is used in this algorithm where queues are created for each interface of every switch in the network. The minimum rate parameter and the priority settings of the queues are varied in accordance with the minimum bandwidth guaranteed to the user. Such a guaranteed rate flow is also to be given higher priority compared to other flows. The queues are created using ovs-vsctl commands.

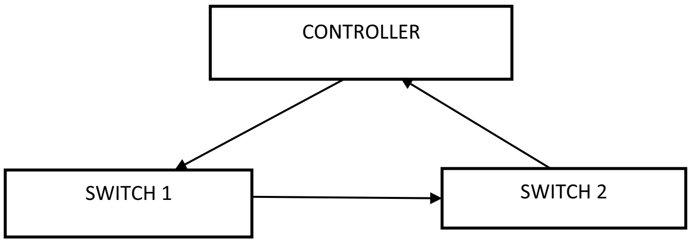


Fig. 6 Propagation delay measurement

In the case of low-priority or best effort class traffic, when there are guaranteed rate QoS flows traversing the same path, they experience starvation and packet loss. In order to minimize this effect a new mechanism is used to handle best effort flows.

The procedure is outlined as follows.

1. All the available paths between traffic source and sink are computed.
2. Switch port statistics from the Ryu controller to the switches are collected and the number of bytes through each port is updated in tables in the SDN controller.
3. The total traffic is sent on multiple paths instead of by a single shortest path by updating the switch group tables and bucket weight according to the Eq. (3).

$$\text{Bucket weight, } B_W(p) = \frac{A_b}{A_i + A_b} * 100 \tag{3}$$

A_b = traffic sent from backup path ports; A_i = traffic sent from primary path port p.

4.2 Delay Guarantees

In order to provide guarantees for delay, a delay measurement module is used. The two main components contributing to total delay are the propagation delay and the queuing delay.

For the measurement of the propagation delay, a probe packet is originated from the SDN controller to the host which requested the delay sensitive connection from which the packet originates and is routed to its destination and back to the controller as seen from Fig. 6.

For the measurement of the queuing delay, a M/M/1/b queue is assumed. Based on previous measurements, the service rate is found to be 10^6 packets per second. The inter-arrival time is Poisson distributed and varied according to the application.

The procedure for finding a suitable delay constrained path is outlined as follows.

1. All the available paths between the source and sink hosts are computed.
2. For each path found, it is checked if there are other bandwidth demanding flows through the same path. If it is so, give higher priority to delay constrained flow in the queues assigned to it so as to achieve pre-emption of traffic.
3. For each path found, assess if there are other delay constrained flows through the same path. If it is so, assign an alternate path to the flow. More than one delay sensitive flow cannot be handled through the same link without compromise.

4.3 Reliability Constraint

In order to ensure reliable service, failure recovery mechanisms must be in place. There are two possible recovery approaches which are reactive failure recovery and proactive failure recovery. In this work, proactive approach is used by exploiting the fast failover group of the mininet switches. The procedure by which multiple paths are selected as primary and backup is as follows.

1. Link failure messages are scanned and link utilization is computed according to the Eq. (2).
2. All the available paths between the traffic source and sink hosts are computed.
3. The path with overall minimum link utilization is assigned as the primary path.
4. Up to n backup paths are assigned.

The proactive approach consumes additional entries in the limited size flow table of the switches. This can be overcome by installation of additional flow rules only when requested by a QoS user and then deleting them afterwards based on idle timeout value.

5 Implementation and Results

All the modules are simulated in Mininet network emulator in Ubuntu 14.04 LTS. The controller framework used is Ryu controller. All the applications are integrated and run on top of the Ryu controller.

5.1 Load balancing

The results pertaining to congestion and load balancing are presented below. The topology under test is shown in Fig. 7.

After congestion detection and load balancing according to the existing algorithm i.e., without the disjoint nodes condition applied, the selected paths are, [1–4] and [1, 3, 4, 8]. It is seen that the paths chosen are disjoint.

From the flow table in Fig. 8, for all packets matching IP source address of 10.0.0.2 and IP destination address of 10.0.0.1, the action will be to follow the instructions in Group ID 2985690309.

From the group table in Fig. 9, for group ID matching 2985690309 the action will be to forward 48% of the packets through port 1 and 52% of the packets through port 2.

In this manner load balancing operation is accomplished.

The simulation parameters are shown in Table 3. The threshold and congestion window parameters are set based on the reasoning followed in [18]. The packets lost are shown in Fig. 10, it is shown for the case when the network is congested while routing on the shortest path, after load balancing using the traditional approach and after load balancing based on disjoints of paths. It can be clearly seen that after load balancing using the existing approach i.e., with no disjoint nodes condition, the amount of packet lost is only reduced by 19.8%. However, after load balancing using the improved algorithm with disjoint nodes condition, the amount of packets lost is reduced by 98.8%.

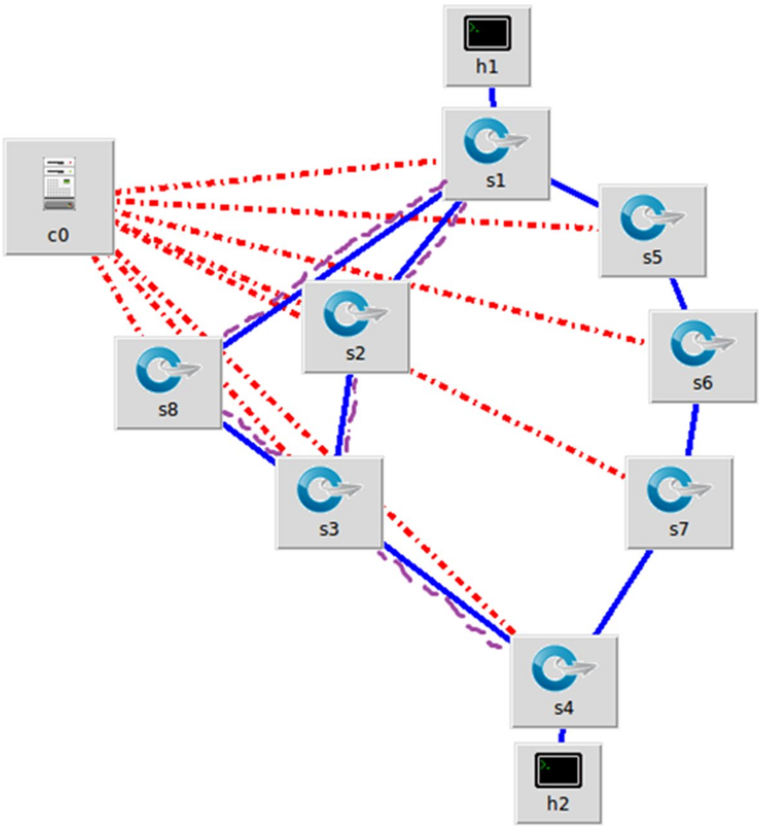
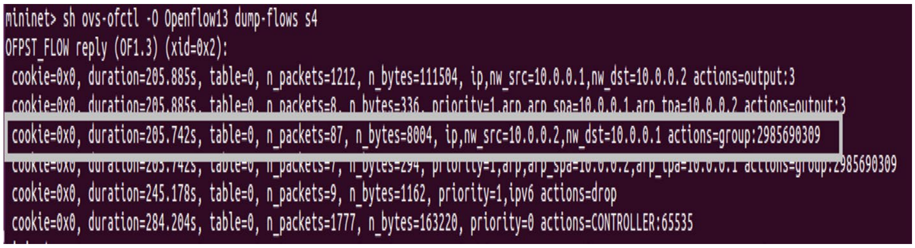
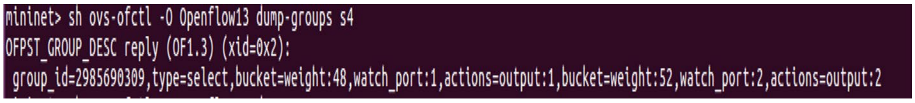


Fig. 7 Mininet topology



```
mininet> sh ovs-ofctl -O Openflow13 dump-flows s4
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x0, duration=205.885s, table=0, n_packets=1212, n_bytes=111504, ip,nw_src=10.0.0.1,nw_dst=10.0.0.2 actions=output:3
 cookie=0x0, duration=205.885s, table=0, n_packets=8, n_bytes=336, priority=1,arp,arp_spa=10.0.0.1,arp_tpa=10.0.0.2 actions=output:3
 cookie=0x0, duration=205.742s, table=0, n_packets=87, n_bytes=8004, ip,nw_src=10.0.0.2,nw_dst=10.0.0.1 actions=group:2985690309
 cookie=0x0, duration=205.742s, table=0, n_packets=7, n_bytes=294, priority=1,arp,arp_spa=10.0.0.2,arp_tpa=10.0.0.1 actions=group:2985690309
 cookie=0x0, duration=245.178s, table=0, n_packets=9, n_bytes=1162, priority=1,ipv6 actions=drop
 cookie=0x0, duration=284.204s, table=0, n_packets=1777, n_bytes=163220, priority=0 actions=CONTROLLER:65535
```

Fig. 8 Flow table after load balancing



```
mininet> sh ovs-ofctl -O Openflow13 dump-groups s4
OFPST_GROUP_DESC reply (OF1.3) (xid=0x2):
 group_id=2985690309,type=select,bucket=weight:48,watch_port:1,actions=output:1,bucket=weight:52,watch_port:2,actions=output:2
```

Fig. 9 Group table after load balancing

Table 3 Simulation parameters

Parameter	Value
Controller framework	Ryu
Number of switches	8
Number of hosts	2
Threshold	1.3125
Mininet SDN controller	Remote
Openflow version	V1.3
Congestion window	3
Number of packets sent	15,000
Protocol used	TCP
Source IP address	S4(10.0.0.2)
Destination IP address	S1(10.0.0.1)

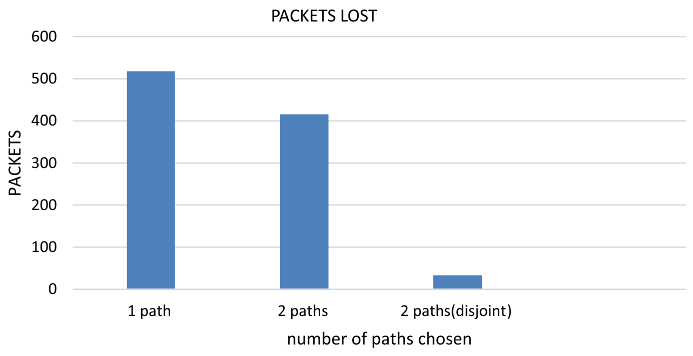


Fig. 10 Packets lost

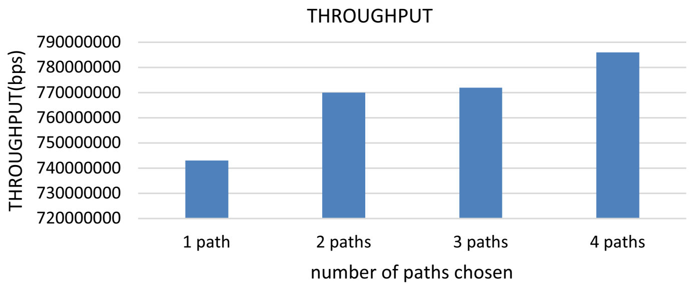


Fig. 11 Throughput

Figure 11 shows the precedent increase in the number of bytes successfully received at the sink host against the increase in the number of alternate routes in addition to the primary path. The maximum capacity of each link is 1 Gbps out of which the throughput of 787 Mbps is achieved for the data plane traffic. The throughput increases with increasing number of disjoint backup paths. The reason for this increase is the traffic is

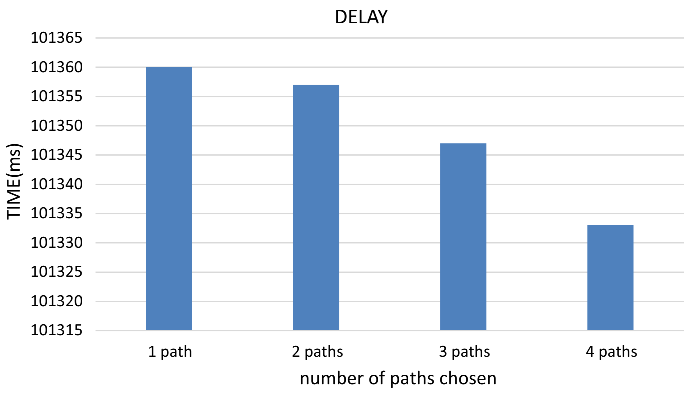


Fig. 12 Delay

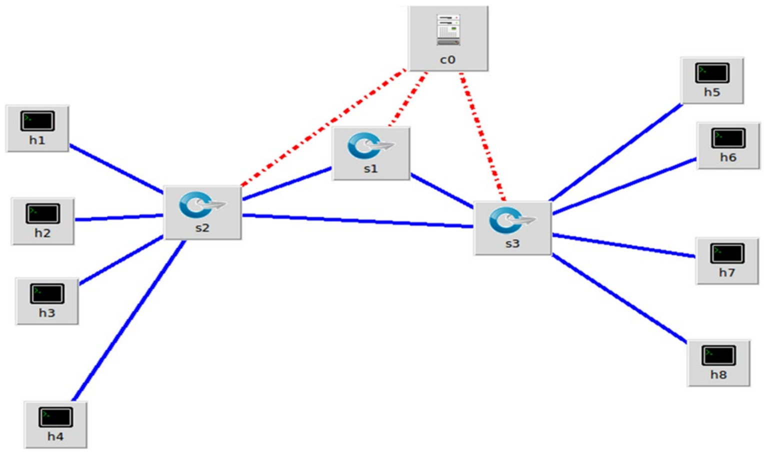


Fig. 13 Topology for bandwidth guarantees

split among several links of capacity 1 Gbps each instead of a single link. This accounts for the improvement observed with respect to throughput. Elephant flows which require huge throughput can be handled well by this procedure. For example, if a flow demanding 780 Mbps is to be served we can utilize 4 alternate paths to serve the request and hence prevent overburdening of a single path which is also serving other flow requests.

The delay performance with increasing number of disjoint alternate paths is seen in Fig. 12. The effect of the disjoint nodes causes significant improvement. The simultaneous transmission along multiple paths decreases the transmission and propagation delay and the effects are seen in the overall end to end delay.

Table 4 Simulation parameters for QoS provisioning

Controller framework	Ryu
Number of switches	3
Number of hosts	8
Size of UDP packet	1470 bytes
Bandwidth for AF11	100 Kbps
Bandwidth for AF21	200 Kbps
Bandwidth for AF31	500 Kbps
Capacity of each link	1 Mbps

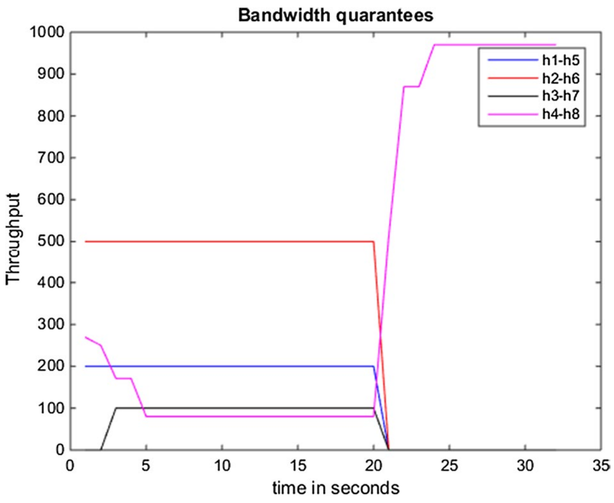


Fig. 14 Bandwidth guarantees

5.2 QoS Provisioning

5.2.1 Bandwidth Guarantees

The bandwidth guarantees are offered to the user. The topology under test is shown in Fig. 13. The simulation parameters have been tabulated in Table 4.

The host mapping is as follows: h1–h5, h2–h6, h3–h7, h4–h8 i.e., the host h1 sends traffic directed to h5, h2–h6, h3–h7 and h4–h8. The class of service demanded by the users are as follows:

- Host h1–h5: AF21
- Host h2–h6: AF31
- Host h3–h7: AF 11
- Host h4–h8: best effort

The packets are generated at the source host by using Distributed Internet Traffic Generator (D-ITG) [20] and the instantaneous throughput is measured using Wireshark at interfaces of the sink host. The QoS source hosts h1, h2 and h3 send traffic at 200 Kbps, 500 Kbps and 100 Kbps respectively based on their bandwidth demands. The non-QoS

Table 5 Packets received

Hosts	Packets sent	Packets received
h1–h5	342	342
h2–h6	852	852
h3–h7	172	172
h4–h8	1702	1364 (20% loss)

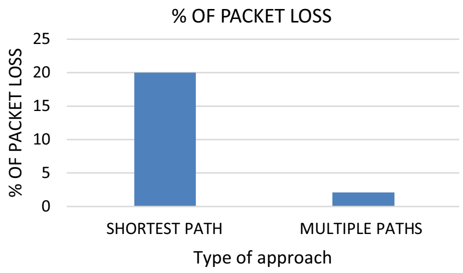


Fig. 15 Packet loss comparison

source host h4 sends its traffic at 1 Mbps. The instantaneous throughput observed at all the receiver hosts are plotted and shown in Fig. 0.14. The source hosts send their packets for the time period of 20 s. Upto 20 s, it can be observed from Fig. 14 that throughput received between hosts h1–h5, h2–h6 and h3–h7 are 200 Kbps, 500 Kbps and 100 Kbps respectively. For these hosts, the guaranteed rate is offered according to their requirement. Between hosts h4–h8 which are not QoS users, it can be observed that the throughput offered between 0 to 20 s is lesser. After 20 s, the full throughput of 1 Mbps is offered to it. The throughput offered to the Best effort user is less when the QoS users transmit their flow, while after the QoS users finish their flows; this Best effort flow is granted the full available bandwidth. This behavior is in accordance with the requirements. When traffic from a QoS user is being served, the low priority user traffic is starved. When the QoS users finish sending their traffic, the low priority user can send its traffic. The account of number of packets received is shown in Table 5.

Under the same simulation conditions, when low priority flows are sent on multiple paths the % of packet loss decreases to almost 0% as seen from the Fig. 15. This is due to the utilization of links which are less congested than the primary path and thereby increasing the bandwidth offered to the low priority Best effort user without compromising the QoS user demands.

5.2.2 Delay Guarantees

In order to provide delay guarantees between source and destination simulations were conducted to predict the upper bound. The delay with respect to variations in the inter-arrival time is tabulated in Table 6.

In order to test the pre-emption of delay sensitive traffic over bandwidth constrained traffic, bandwidth guaranteed traffic of 500 Kbps was sent over the same link as a delay sensitive flow. The results are tabulated in Table 7. The delay taken for a bandwidth constraint flow is higher compared to a delay sensitive flow. This is because higher

Table 6 Delay predicted and practical

λ (packets/s)	Average delay (practical) (ms)	Delay (predicted) (ms)
10	0.172	0.1851
20	0.171	0.1852
30	0.163	0.1853
40	0.172	0.1854
50	0.164	0.1855
60	0.170	0.1856
70	0.157	0.1857
80	0.161	0.1858
90	0.166	0.1859

Table 7 Delay pre-emption

Hosts	Rate (Kbps)	Packets sent	Packet loss	Request type	Total delay (ms)
h1–h5	500	427	0%	AF31 (bandwidth)	4.305
h2–h6	500	427	0%	AF12 (delay)	0.687

Table 8 Failure recovery time

Approach taken	Failure recovery time (ms)
Proactive	0.4125
Reactive	567

priority queues are assigned to the delay sensitive flow forcing delay sensitive traffic first out of the switch interface.

5.2.3 Reliability Guarantees

The proactive and reactive failure recovery mechanisms are compared. The failure recovery time is much lesser for the proactive approach as shown in Table 8.

The packet delivery ratio is compared between the two approaches as the numbers of path changes are increased. The result is shown in Fig. 16. In the reactive approach, the packet delivery ratio decreases steadily with increasing route changes whereas in the proactive approach, the number of path changes has no effect on the packet delivery ratio. This is because the backup links are pre-determined and queues created before the start of the flow. Hence the proactive approach is a more robust one for a constantly changing network.

This shows that the proactive approach much be the preferred one in an unpredictable network which is prone to link failures.

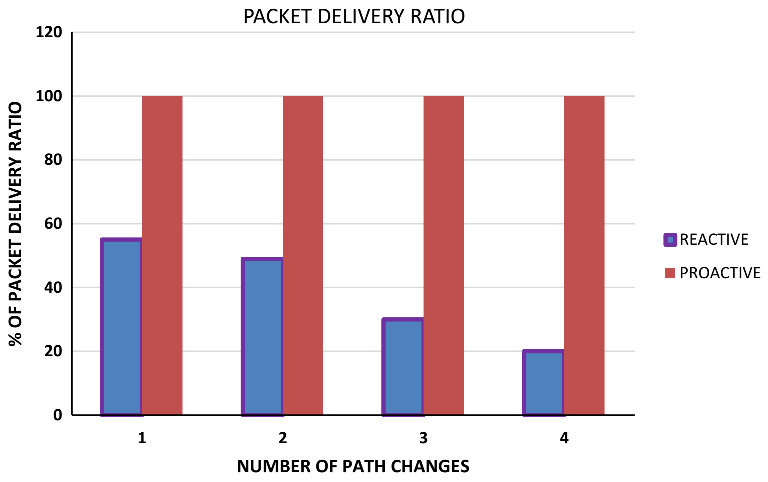


Fig. 16 Packet delivery ratio

6 Conclusion and Future Work

Thus this work gives a new enhanced load balancing and QoS provisioning algorithm with suitable comparisons where required. The major contributions or novel ideas in this work include the following:

- In the event of elephant flows, the switch detects them and follows a multiple path approach based on bucket weights.
- Grouping the QoS traffic classes based on throughput, delay and fault tolerance and creating queues uniquely for each class at each switch interface so that it can be used for incoming QoS traffic.
- In the event of non-QoS traffic taking the same path as the QoS traffic, the non-QoS traffic is sent on multiple paths based on the weighted approach instead of through the same path taken by the QoS traffic in order to avoid its starvation.

An extension of this work could include an effective mechanism for keeping the control packets in check and below threshold levels, adding security enhancements and a distributed SDN approach for load balancing.

Data availability The data generated and analysed during the current study are available from the corresponding author on reasonable request. The code base for SDN controller and traffic generators were obtained from: https://github.com/faucetsdn/ryu; https://github.com/jbucar/ditg.

References

1. Kreutz, D., Ramos, F. M. V., Veríssimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE, 103*(1), 14–76.
2. Lee, R. & Jeng, B. (2011). Load-balancing tactics in cloud. In *International conference on cyber-enabled distributed computing and knowledge discovery* (pp. 447–454).
3. Zhang, H. & Guo, X. (2014). SDN-based load balancing strategy for server cluster. In *IEEE international conference on cloud computing and intelligence systems* (pp. 662–667).

4. Chen, L., Chen, K., Bai, W., & Alizadeh, M. (2016). Scheduling mix-flows in commodity datacenters with Karuna. In *ACM SIGCOMM conference* (pp. 174–187).
5. Bai, W., Chen, L., Chen, K., & Wu, H. (2016). Enabling ECN in multi-service multi-queue data centers. In *USENIX symposium on networked systems design and implementation* (pp. 537–550).
6. Ghorbani, S., Yang, Z., Godfrey, P. B., Ganjali, Y., & Firoozshahian, A. (2017). DRILL: Micro load balancing for low-latency data center networks. In *ACM SIGCOMM conference* (pp. 225–238).
7. Liu, J., Li, J., Shou, G., Hu, Y., Guo, Z., & Dai, W. (2014). SDN based load balancing mechanism for elephant flow in data center networks. In *International symposium on wireless personal multimedia communications* (pp. 486–490).
8. Long, H., Shen, Y., Guo, M., & Tang, F. (2013). LABERIO: Dynamic load balanced routing in open flow-enabled networks. In *IEEE international conference on advanced information networking and applications* (pp. 290–297).
9. Lan, Y. L., Wang, K., & Hsu, Y. H. (2016). Dynamic load-balanced path optimization in SDN-based data center networks. In *International symposium on communication systems, networks and digital signal processing* (pp. 1–6).
10. Wang, Y., & You, S. (2018). An efficient route management framework for load balance and overhead reduction in SDN-based data center networks. *IEEE Transactions on Network and Service Management, 15*(4), 1422–1434.
11. Guo, Z., Dou, S., Wang, Y., Liu, S., Feng, W., & Xu, Y. (2021). Hybrid flow: Achieving load balancing in software-defined WANs with scalable routing. *IEEE Transactions on Communications*. https://doi.org/10.1109/TCOMM.2021.3074500
12. Binh, L. H., & Duong, T.-V.T. (2021). Load balancing routing under constraints of quality of transmission in mesh wireless network based on software defined networking. *Journal of Communications and Networks, 23*(1), 12–22.
13. Krishna, H., Adrichem, N., & Kuipers, F. (2016). Providing bandwidth guarantees with open flow. In *Symposium on communications and vehicular technologies (SCVT)* (pp. 1–6).
14. Kumar, R., Hasan, M., Padhy, S., Evchenko, K., Piramanayagam, L., Mohan, S., & Bobba, R. (2017). End-to-end network delay guarantees for real-time systems using SDN. In *IEEE real-time systems symposium (RTSS)* (pp. 231–242).
15. Oh, B., Vural, S., Wang, N., & Tafazolli, R. (2018). Priority-based flow control for dynamic and reliable flow management in SDN. *IEEE Transactions on Network and Service Management, 15*(4), 1720–1732.
16. Zheng, J., Xu, H., Zhu, X., Chen, G., & Geng, Y. (2019). Sentinel: Failure recovery in centralized traffic engineering. *IEEE/ACM Transactions on Networking, 27*(5), 1859–1872.
17. Lee, S. W., & Chan, K. (2019). A traffic meter based on a multicolor marker for bandwidth guarantee and priority differentiation in SDN virtual networks. *IEEE Transactions on Network and Service Management, 16*(3), 1046–1058.
18. Devapriya, R. D. & Gandhi, S. I. (2020). Enhanced load balancing and QoS provisioning algorithm for a software defined network. In *International conference on emerging trends in information technology and engineering (IC-ETITE)* (pp. 1–5).
19. Abbou, A. N., Taleb, T., & Song, J. (2021). A software-defined queuing framework for QoS provisioning in 5G and beyond mobile systems. *IEEE Network, 35*(2), 168–173.
20. Avallone, S., Guadagno, S., Emma, D., Pescape, A., & Ventre, G. (2004) D-ITG distributed internet traffic generator. In *Proceedings of first international conference on the quantitative evaluation of systems. QEST 2004* (pp. 316–317).

C. Bharanidharan did my BE computer science and engineering in adhiparasakthi engineering college and degree awarded by Anna university Chennai, ME computer science in annamalai university at 2013 and pursuing (PhD) at Madras Institute of Technology campus Anna University. And my research area includes wireless networks, Software Defined Networking, 5G.

Dr.S. Indira Gandhi did her BE Electronics and Communication in AC Tech, Karaikudi, and ME Laser & Electro Optic Engineering at College of Engineering, Anna University, and PhD from Madras Institute of Technology, Anna University. She is currently working as an Associate Professor in Madras Institute Of Technology campus Anna University. She has around 30 national, international journals, And conference proceedings. Her research interest includes communication, optical networks,image processing.

R. D. Devapriya received her Electronics and Communication Engineering BE degree from College of Engineering, Guindy in 2017 and ME degree specializing in Communication and Networking from Madras Institute of Technology in 2020.She is currently pursuing research at National Institute of Technology, Tiruchirappalli as a Junior Research Fellow. Her research interests include Software Defined Networking, 5G and Intelligent Reflecting Surfaces.