Computers, Materials & Continua
DOI:10.32604/cmc.2022.019766
Article

Tech Science Press

Intelligent Transmission Control for Efficient Operations in SDN

Reem Alkanhel[1], Abid Ali[2,3], Faisal Jamil[4], Muzammil Nawaz[2], Faisal Mehmood[5] and Ammar Muthanna[6,7,*]

[1]Department of Information Technology, College of Computer and Information Sciences, Princess Nourah Bint Abdulrahman University, Riyadh, Saudi Arabia
[2]Department of Computer Science, University of Engineering and Technology, Taxila, 47080, Pakistan
[3]Department of Computer Science, Govt. Akhtar Nawaz Khan(Shaheed) Degree College KTS, Haripur, 22620, Pakistan
[4]Department of Computer Engineering, Jeju National University, Jeju-si, Jeju Special Self-Governing Province, 63243, Korea
[5]School of Information and Communication Engineering, Zhengzhou University, China
[6]Department of Telecommunication Networks and Data Transmission, The Bonch-Bruevich Saint-Petersburg State University of Telecommunications, 193232, Saint Petersburg, Russia
[7]Peoples Friendship University of Russia (RUDN University) 6 Miklukho-Maklaya St, Moscow, 117198, Russian Federation
*Corresponding Author: Ammar Muthanna. Email: ammarexpress@gmail.com
Received: 25 April 2021; Accepted: 28 September 2021

Abstract: Although the Software-Defined Network (SDN) is a well-controlled and efficient network but the complexity of open flow switches in SDN causes multiple issues. Many solutions have been proposed so far for the prevention of errors and mistakes in it but yet, there is still no smooth transmission of pockets from source to destination specifically when irregular movements follow the destination host in SDN, the errors include packet loss, data compromise etc. The accuracy of packets received at their desired destination is possible if networks for pockets and hosts are monitored instead of analysis of network snapshot statistically for the state, as these approaches with open flow switches, discover bugs after their occurrence. This article proposes a design to achieve the said objective by defining the Intelligent Transmission Control Layer (ITCL) layer. It monitors all the connections of end hosts at their specific locations and performs necessary settlements when the connection state changes for one or multiple hosts. The layer informs the controller regarding any state change at one period and controller collects information of end nodes reported via ITCL. Then, updates flow tables accordingly to accommodate a location-change scenario with a route-change policy. ICTL is organized on prototype-based implementation using the popular POX platform. In this paper, it has been discovered that ITCL produces efficient performance in the trafficking of packets and controlling different states of SDN for errors and packet loss.

Keywords: Software-defined network; ITCL; POX; open flow

This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1 Introduction

The development in Information and Communication Technology, the advanced requirements, and enhancements for existing computing systems led the network style towards a new software-based networking system called software-defined networking (SDN) [1] which is a controller-based network with OpenFlow switches (OF-switches). The controller uses the OpenFlow application, which contains programmed modules and code, the communication medium between the controller and OpenFlow switches while SDN provides open-source platforms to run and modify the controller's functionality according to the necessity of the network. It has offered several innovations in networking with more efficiency than the traditional network, such as load balancing, network security, network intelligence, etc. while network measurements are more straightforward because of centralized controller [2].

1.1 Infrastructure of SDN

Being a part of expanding the system of the telecommunication world, most of us are well acquainted with traditional network construction which includes switches, routers, hubs, modems, or network adapters interconnected either wired or wirelessly. The Intercommunication among these network peripherals is attained by implementing several guidelines called Communication Protocols [3]. In this network, the main tasks of the network rely on dedicated devices like routers, switches, and controllers, unlike SDN (Software Defined Network) in which Control aircraft, Management planes, and Data planes are accessible on the same instruments in a standard network. This means that the packet maintained all three information about data, control, and management airplane. Configuration of the kind of network is fixed and merchant-specific [4].

1.2 The New Paradigm Shift SDN

SDN is a new paradigm shift which is capable of overcoming the security issues and other limitations of traditional networking. Unlike the conventional network, two device planes, i.e., control and data planes, have been decoupled in SDN, instead, the control plane is moved to the software layer and behaves like a centralized control layer that manages the network policies [5]. This structure provides more secure and flexible network configurations than manual configurations of network devices in a traditional network [6].

SDN devices, such as switches and routers, have turned into simple forwarding devices, which only implement data-plane logic [7] while the control of the whole network is handed over to a centralized unit called the controller, which makes all decisions regarding the flow of packets from one location to another location and sets the primary behaviors of the network. Moreover, the controller defines the policies which are further used to run the whole network flow [8–10]. By using forwarding paths one or more hosts send packets is also determined by the controller. In addition, a path-defining process controller defines a specified rule for a specific host when the host is about to start a packet flow to another host [11].

1.3 SDN Performance

There are many reasons which cause traffic-related issues, i.e., packet loss during data communication between hosts, in a traditional network, such a scenario may occur due to traffic or link congestion, bad performance of a network device (switch, router, etc.), bugs on the device, or faulty hardware used in the network [12]. For a smooth network process the links should be well active; if somewhere a link's capacity is full, all the packets must wait for its workable state into a queue which cause congestion and maximizes packet loss chances. A low-performance device can only handle the

traffic according to its capacity compared to the performance of a programmed controller [13–15]. Suppose, A significant bits of traffic are to be passed through such a device, in that case, the device can only give the number of bits according to its internal storage, which will cause communication delay and congestion in the network while bugs on devices is another factor causing traffic issues as most of the available devices in the market are not free of bugs and faults [16]. Moreover, damaging of hardware equipment also causes link failure and disconnections between network links.

1.4 Traffic\|Packets Loss Issues in SDN Network

In software-defined networking, all process of traffic flow is controlled by a centralized controller. As in SDN switches are OpenFlow that take instructions from the controller. When hosts start communication, the routing path for the flow of packets between hosts is defined by the controller by installing rules [17]. For example, when 'host A' sends packets to another 'host B', the information packet first travels towards the controller to take direction for its packet's flow and according to the defined SDN policy, the controller then assigns the shortest or suitable forwarding path [18]. The rest of the packages follow the rule and start flow according to the controller's defined control. Though, there exist some traffic-related issues in SDN on its level which may cause a forwarding loop and traffic congestion, packets drop may also occur due to the said reasons or by facing an unexpected link failure in the network [19].

1.5 The Considerable Packets Loss in SDN

SDN is based on a centralized operating controller, the handling of activity and state-change events of the network are also based on how smartly the controller monitors the complexity of network states. In each newly starting communication, the first packet goes to the controller where the controller finds its destination and then its location on the network [20]. Then, the controller sets a switch-by-switch forwarding path to flow all packets of its sequence to the destination host [21]. After taking the rule from the controller, the rest of the packets only follow the controller's defined path to reach the destination host. Meanwhile, if the host moves from its original location and appears to another site, the OpenFlow switch of the last location of the destination host gets confused. This time packets reach their destination switch, but there is no host available on the OF-Switch to receive these packets, which causes silent drop of packets [22].

SND plays a vital role in the operation of OpenFlow traffic enhancement. The complexity of Networks leads towards facing bugs. The proposed solutions, so far, cannot prevent it entirely crowd and errors as the variety of mistakes are also arising with the growing worth of OpenFlow. The SDN network traffic may cause errors and bugs while using the OpenFlow information. The congestion and other such problems may arise, and these problems motivate us to define a new solution through a control transmission layer to deliver the traffic flow efficiently. The primary motivation is absence of effective material that monitors all the connections of end hosts at their specific locations and performs necessary settlements when the connection state changes for one or multiple hosts while the layer informs the controller regarding any state change at one period.

The rest of the paper is generally organized as; in Section 2, we presented the related work based on Load balancing in SDN, Section 3 presented the framework of the proposed solution and model diagram, in Section 4, we simulate our techniques. Finally, in Section 5, we have shown results and graphs, and at last, conclusion and future directions are discussed.

2 Related Work

SDN consists of three main stacks SDN applications, SDN controller, and SDN networking devices. The data plane and control plane are decoupled in SDN architecture and distributed as the application, control, and infrastructure layers into the layered structure. Some of the existing techniques are used to monitor and control SDN networks to manage network performance. A CharryPick [23] technique is used to trace packet trajectories and minimize two data plane resources. One is Reduces Switch flow rules, and the other Minimizes packet header space. Each network link is assigned a unique identifier in this technique, and all switches embed this identifier into each packet header during the forwarding process. This minimizes the switch flow rules, with this process, the packet header goes with a high packet header space, especially the line of packet transverse in the forwarding loop due to the failure of short paths. To reduce packet header space, only minimum number of essential links are picked. For a 48-port switch, CherryPick requires (log (48) = 6 bits) to represent each link. Simultaneously, the path tracer, the previous technique, required log(p) bits for packet header space where P represents the number of absolute paths between the source host and destination host. After evaluation, the cherrypick technique used minimal header space and reduced switch flow rules compared to other approaches (Pathlettracer, Pathquery). On the other hand, many switch-flow rules are required in the Pathlet-tracer technique which is used to reduce the packet headers. Controller floods a packet to learning switch to find about an unknown destination host [24,25].

One of the flooded packets responds to the controller about its host destination's location and learns its location. Here the remaining packets in flood can cause additional, unnecessary rules installed in the network, which might occur a packet storm. In the said situation, such bugs can cause serious consequences. For example, a forwarding loop is considered the first bug that would stay until it is related to the rule expires. Writing of additional rules (excess restrictions may decrease the network performance [26]. 12_multi, a POX platform component, involves maintaining the mapping of hosts at a switch and port for every control in the network. According to the controller's code, it is located for the destination host when a packet is sent to it because the host's destination is unknown [27].

The controller floods the packets to switches to find an unknown destination. When flooded packets reach their adjacent switches, the controller instructs to continue flooding of packets, then it sees reply packets and learns the host's destination. The remaining flood stays there in the network, and by forwarding these packets, the switch to switch reaches them for OpenFlow switches [28]. These duplicate packets are again sent to the controller by OpenFlow switches as the destination's location is unknown. The controller resultantly installs additional rules for duplicate packets, which may cause inefficiency in network performance. Query language has been introduced called path query [29] through regular expressions, which also includes SQL-like group-by construct to take information from the network. The path query monitors network traffic by transmitting packet trajectories at any network level on the data plane. It is also possible to specify different queries at a time that are independent. Tab. 1 depicts more in-depth literature after comparing the related works of flow table entries reduction mechanism.

Table 1: Comparison of the related works of flow table entries reduction mechanism

Related Work	Methodology	Algorithm	Advantages	Disadvantages	Use case
[30]	Hidden markov model (HMM)	Quine-Mcclustkey algorithm.	• Manage multiple flow table. • Reduce flow processing time	• No OpenFlow Management	Flowtable Management
[31]	Binary tree aggregation	Shrink the flow table size	• Reduce flow table size • fast rule updating time	• Not intelligently define control flow information	Flowtable Management
[32]	Binary Tree aggregation	Rule optimization	• Reduce the number of flows entries • full-filled flow tables. • Retaining the original QoS	• Extends the range of transmission	Flowtable Management
[33]	aggregation	Redundant flow entries	• Reduce flow table overflow problem • Less flow aggregation convergence time.	• Not effectively enhance the extraction of flow data entries.	Flowtable Management
[34]	Flow entry compression	Flow entry compression	• Reduce flow table size	• Through comparing the header matching • Throughput may be compromised	Flowtable Management
[35]	One big switch	One big switch	• Distribute rules • Abstracted forwarding element called "one big switch."	• The fault tolerance reduces the network performance.	Distributed ACL and load balancer

(Continued)

Table 1: Continued

Related Work	Methodology	Algorithm	Advantages	Disadvantages	Use case
[36]	Flow entry compression	Flow entry compression	• To maximize the utilization of SDN switches, flow table	• The switch optimization enhances • Less effective elemental techniques.	Traffic engineering
[37]	Partition based	Bit and subset weaving to merge flow entries	• Reduce flow table size • Fast rule updating time	• The enhancement of the rulemaking process.	Flow table Management
[38]	Flow Entries and management Protocol	Split flow table entries management	• Decompose flow entries into smaller part	• All the entries are checked and provide	Distributed ACLs
[39]	Optimization	Linear optimization model	• Modeled rule allocation problem in • Resource-constrained OpenFlow networks	• More bandwidth utilization for effective monitoring.	Traffic engineering
[40]	Divide and conquer	Break tables into several smaller sub-tables.	• Ternary content addressable • memory (TCAM) shortage problem	• Enhance the processing time.	Distributed ACLs

The SDN is convenient, flexible in its structure, and economical by providing programmable controlled strategies as OpenFlow layered architecture [41]. The OpenFlow is programmed for better performance and mature network architecture utilizing error-free communication. But there are several issues and problems faced, when nodes want to communicate with each other. An SDN Switch process *packet_in* and *packet_out* messages and generates a possible reply to the controller. But a completely processed *Packet_out* message does not assure the guarantee for the actual existence of its included packet in the switch, which may be dropped due to switch failure, congestion, or by facing invalid ports. In authors [42] propose a hierarchical SDN control plane approach to guarantee the E2E QoS among multiple domains with various QoS classes on the E2E path. We propose a controller module for selecting the most suitable QoS class for each domain in the E2E path, based on multi-criteria decision-making by using the technique for order of preference by similarity to ideal solution (TOPSIS). A new approach is proposed with a controller with an optimum feature set that must be available for SDN. Furthermore, a cluster of optimum feature set controllers will overcome a SPOF

and improve the QoS in SDN. Herein, leveraging an analytical network process (ANP), we rank SDN controllers regarding their supporting features and create a hierarchical control plane-based cluster (HCPC) of the highly ranked controller computed using the ANP, evaluating their performance for the OS3E topology. In [43], a two-step approach is proposed for SDN controller selection. First, the controllers are ranked with analytical network process (ANP) according to their qualitative features, which influence the performance of these controllers, then, a performance comparison is performed to check for the QoS improvement.

Fig. 1 describes the scenario. The rest of the packets should be forwarded to the new location of Host B, which is accommodated in our proposed solution. In the SDN environment, the packet loss problem always occurred with a simple SDN environment where all the hosts are connected and provide packet flow rates. The SDN network traffic may cause errors and bugs while using the OpenFlow information. The congestion and other such problems may arise, and these problems motivate us to define a new solution through a control transmission layer to deliver the traffic flow efficiently. The Transmission Control Layer provides efficient packet transmission and packet delivery control using OpenFlow switches in VANET.

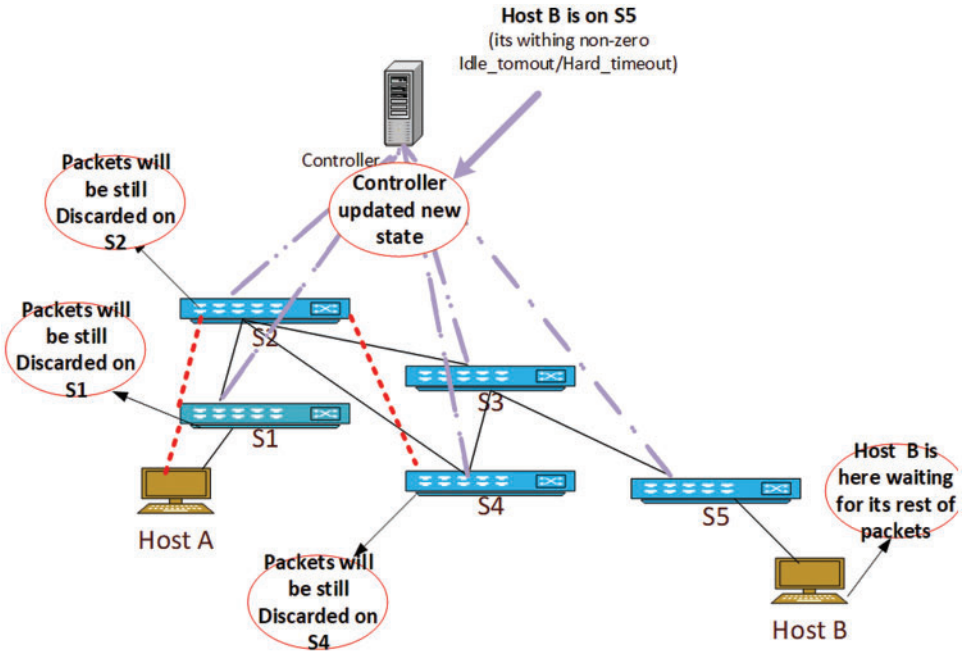

Figure 1: Packets discarded on all switches as per controller's rule

3 Proposed System

To handle the packet loss problems discussed in our problem statement, we proposed an efficient working mechanism capable of monitoring end nodes and informing the controller regarding any state of change within a specific time. It detects a movable host and informs the controller. Then the controller updates the switch to move the rest of the packets towards the new destination of the portable host instead of discarding these packets. Figs. 2a and 2b present that the packets are being forwarded towards the changed location of the destination host. These packets were discarded in the existing solution.

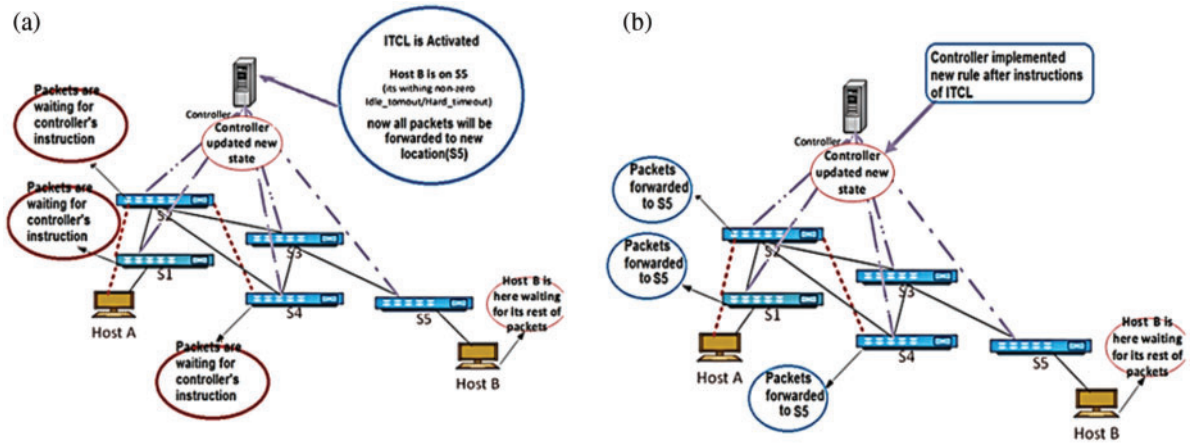

Figure 2: Packet forwarding techniques through destination host in proposed work (a) ITCL Layer instructing controller about movable host (b) all packets forwarded to changed location of destination host

3.1 Preparation and Implementation of Rules

By getting information on topology from the network, we need to install flow tables for switches. First, we maintain I.P. and MAC tables for all active hosts on the network. Second, we implement a MAC-matching policy for end nodes of switches that match the destination host's MAC address while delivering packets to destination end nodes. Finally, we instruct the SDN controller to install flow rules to all available switches on the network.

3.2 Activation of ITCL

We keep active a transmission control layer named "Intelligent Transmission Control Layer," pronounced as ITCL. The said layer intelligently monitors the network, especially the movement of hosts during communication. For example, suppose a host moves from its location to another location during communication within a specific timeout period. In that case, ITCL informs the controller about its new location and instructs to update the flow table of all switches according to the location change state. Updating of flow table forwards the packets of the movable host towards its changed location.

3.3 Workflow of Proposed Solution

When the network starts, ITCL also starts functioning. First, it starts getting information of hosts with I.P. & MAC, then, it instructs the controller to implement an IP-MAC-Matching policy for the packets while delivering them to their end nodes. By implementing the said policy, the packages are tagged with I.P., and the MAC address of the destination host also matches with the I.P. address and MAC address of the destination host before delivering it to the end node.

If I.P. and MAC tagged in packet header matches with the I.P. and MAC of the destination host, the packet delivers to the end node, otherwise, ITCL keeps its I.P. and MAC Address reserve if the host moves to another location until Idle Timeout/Hard Timeout is nonzero. In this state, when I.P. and MAC do not match, the packet also does not get deliver to the end host. Packet waits for new instructions initiated by ITCL to the controller, and controller assigns forwarding path and updates flow table in switches. When the destination host appears on another location within the timeout

period, the ITCL identifies it by its MAC. The controller assigns the reserved I.P., which is retained with the same MAC address instructed by ITCL. As a result, the rest of the packets start travelling towards the changed location of the destination host and gets delivered there. The proposed solution prevents packet loss (packet discard and packet drop) due to the movable host. As the flow diagram in Fig. 7 presents step by step flow of the system.

3.4 Table of Notation

Tab. 2 summarizes the symbols and notations used in this paper and in algorithm one and explains the corresponding meanings.

Table 2: Table of notations

Notation	*Explanation*
$Forward_{path}$	Path forward transmission ratio
P_k	Packet of request
Ns	Number of switches in the network
S_{witch}^{SDN}	SDN Switch
L.B	Load balancer
C_{SDn}	SDN controller
H_{info}	Header info
$\sum_{1}^{n} Li$	Links between nodes in SDN network
F_{info}	Flow information
F_T	Flow table
N_{path}	New forward path
D_{packet}	The packets that needs to forwards for the distributions.
$P_{trans}()$	Transmit the packets from node to node (Nodes are any devices linked with the network)
Pkt_{puched}	Pushed any packet towards the destination.

Fig. 3 contains the flow diagram; it describes how the controller is updated for a host changing its location, and the rest of the packets travel towards the host's changed location.

Algorithm 1: Installation of Flow Rules

Input: Network Packets
Output: $(Forward_{path}) + (\frac{IP}{MAC}$ matching on end node$) + Active_{ITCL}$
Procedure:
Start
$P_{trans}() \leftarrow D_{packet}()$

(Continued)

if ($(pkt.Dist_{IP} \leftarrow EndNode.IP)$&& $(pkt.Dist_{MAC} \leftarrow EndNode.MAC)$
$Pkt_{puched} \rightarrow endNode()$
else while $(T_{out} \rightarrow \sum_{0}^{i} NonZero)$
$Store()_{pkt} \leftarrow ITCL_{route\_change()}$
if $(Route_{Change()} \leftarrow router)$
$pkt \leftarrow newNode()$
End loop
end if
else
$pkt \leftarrow$ send(C, R, pkt)
Packet_discard ()
end

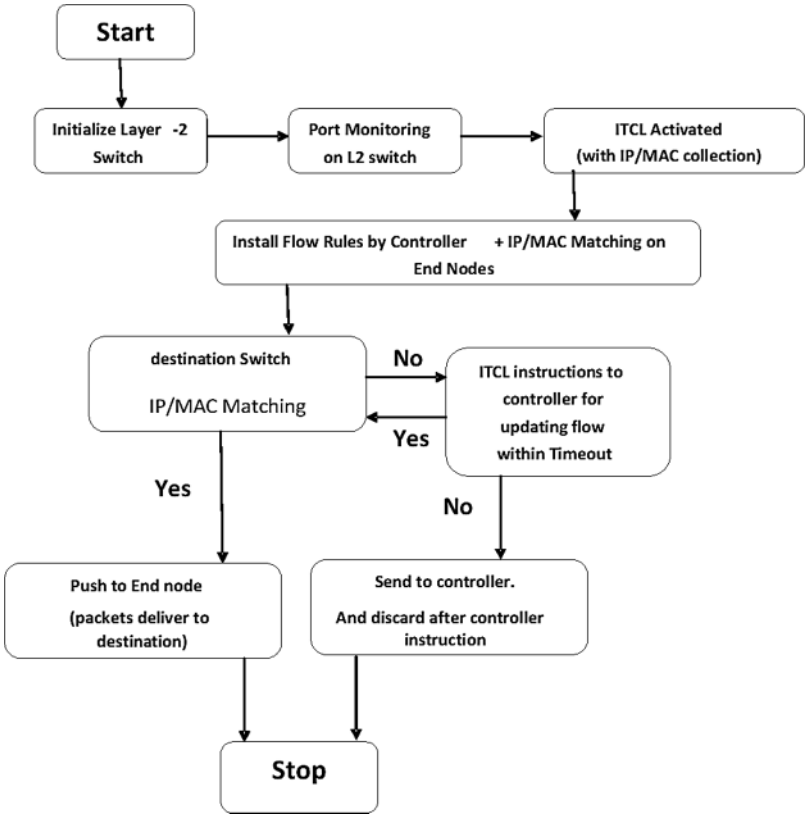

Figure 3: Data flow inside the implemented technique

4 Results and Discussion

We measured various parameters for evaluating our proposed solution as packet loss rate, congestion, measurement of latency, load on CPU, and throughput of proposed algorithms. We also tested multi-state change scenarios and measured the packet loss ratio with other aspects. Mininet is simply a network emulator that virtually provides a complete network environment like switches, links, and controllers. The hosts in Mininet run a standard Linux-based network software and create

a real-like virtual network environment. Mininet is based on a single Linux kernel capable of running Mininet offers learning, research and development, prototyping, testing and debugging, and a whole experimental network on a single machine.

Wireshark is an analyzing tool for analysis of network packets and generates detail as captured as possible. Wireshark can be used by network administrators, network security engineers, or network developers for network troubleshooting to examine network security issues or debug while protocol implementation [44–46]. Furthermore, different parameters are used like bandwidth usage, packet delivery/loss rate, and throughput to evaluate our proposed solution's performance. We passed 1000 packets from the source host to the destination host for each scenario and repeated our scenario ten times. The bandwidth information is saved in Tab. 3.

Table 3: Packets used

No. of packets	Reach time at destination (ms)
100	30
200	60
300	90
400	120
500	150
600	180
700	210
800	240
900	270
1000	300

4.1 Movable Host Detection Time and New Host Registration Time

We evaluate the parameters as in our proposed solution, total time used in detecting a moveable host on the network rather than sending the rest of the packets to it. In the existing solution, the registration time of a host on the network. The time interval for both is measured by delivering N number of packets to a movable host. To understand the scenario's algorithm, we generated a Tab. 5 as follows. There are N numbers of packets to be sent from host to destination. In our discussed network, there are 02 hops between source and destination. We calculated packets reaching a time in a constant state as:

$packets_{reached\,destination}()$ ← N Packets

$hope_{count}(source,\ destination)$ ← H Hope

$perHope_{time}()$ ← T Time

$time_{reached}(destination)$ ← T x H x N

Fig. 4 presents the number of packets that reach the destination with constant v. The said scenario is before moving a destination host. After state change, when a destination host moves to another location, the existing solution re-registers it and sends zero packets. But in the proposed solution, we safe the packets already received by the destination host and the packets in a transit state and discarded

in the existing solution as shown in Tab. 4. Hence, the number of packets to be processed after movable host in the current and proposed solution. So, we saved time as following:

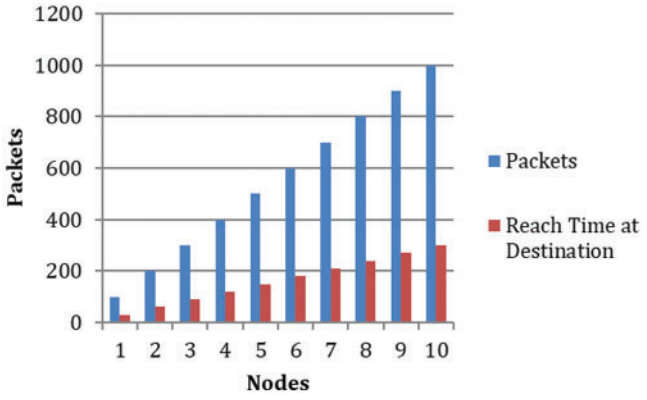

Figure 4: Reach time calculation for packets

Table 4: Retrieved information from Wireshark during states change

Total packets	Reached to destination	Discarded	Dropped	POX controller [46]		Proposed solution	
				Packets to be sent	Packets saved	Packets to be sent	Packets saved
1000	321	53	6	1000	0	626	374
1000	159	13	9	1000	0	828	172
1000	624	57	3	1000	0	319	681
1000	511	62	11	1000	0	427	573
1000	120	8	4	1000	0	872	128
1000	362	56	9	1000	0	582	418
1000	451	43	3	1000	0	506	494
1000	362	22	8	1000	0	616	384
1000	152	32	4	1000	0	816	184
1000	685	28	9	1000	0	287	713

4.2 Existing Solution

$Total_{packet} \leftarrow 1000$

$packets_{reached\,destination}() \leftarrow 321$

$packet_{discard} \leftarrow 53$

$packet_{dropped} \leftarrow 6$

After host movable total packets to be processed $\leftarrow 1000$

Table 5: Bandwidth Usage

POX controller [46]		Proposed solution	
Packets to be processed	Band width/8Mb	Packets to be processed	Bandwidth/8Mb total
1000	0.008	626	0.01278
1000	0.008	828	0.009662
1000	0.008	319	0.025078
1000	0.008	427	0.018735
1000	0.008	872	0.009174
1000	0.008	582	0.013746
1000	0.008	506	0.01581
1000	0.008	616	0.012987
1000	0.008	816	0.009804
1000	0.008	287	0.027875
Total	0.08		0.155651
8Mb-total used	7.92		7.844349

4.3 Proposed Solution

$Total_{packet} \leftarrow 1000$

$packets_{reached\,destination}() \leftarrow 321$

$packet_{discard} \leftarrow 53$

$packet_{dropped} \leftarrow 6$

After host movable total packets to be processed: $= \frac{total_{packet} + packets_{reacherd}(dest) + packet_{discard}}{hosts}$

$= 626$ packets sent

Fig. 5 presents the ratio of the packets to be processed after the host movement. Because processing fewer packets in the proposed solution saves the network's bandwidth, which maximizes network performance. As in Tab. 4, the packets are less processed in the proposed solution; the bandwidth according to the number of packets is measured. Fig. 6 shows the packet ratio to get processed when hosts move. The results demonstrate that the proposed technique has fewer packets to be processed as security after the host move.

$$Bandwidth\ per\ Host = \frac{Total\ bandwidth}{Number\ of\ packets\ to\ be\ processed\ for\ host}$$

4.4 Network Throughput

Another parameter of network performance is throughput. Throughput is defined as 'the maximum performance of the network'. Our described problem network works more after the movable host state because it sends packets from the start after the host changes its location. But in our proposed solution, the packets that have already been received and will be discarded due to movable host are saved. These packets start traveling from their respective switches towards the destination, instead of traveling again from the source host (Fig. 8). Throughput can be measured by dividing "TCP window

size (data packets)" with communication data packets. We can calculate the throughput of the existing solution as following:

$$\text{Throughput} = \frac{TCP\ window\ size\ (Data\ Packets)}{Round\ Trip\ Time\ (RTT)}$$

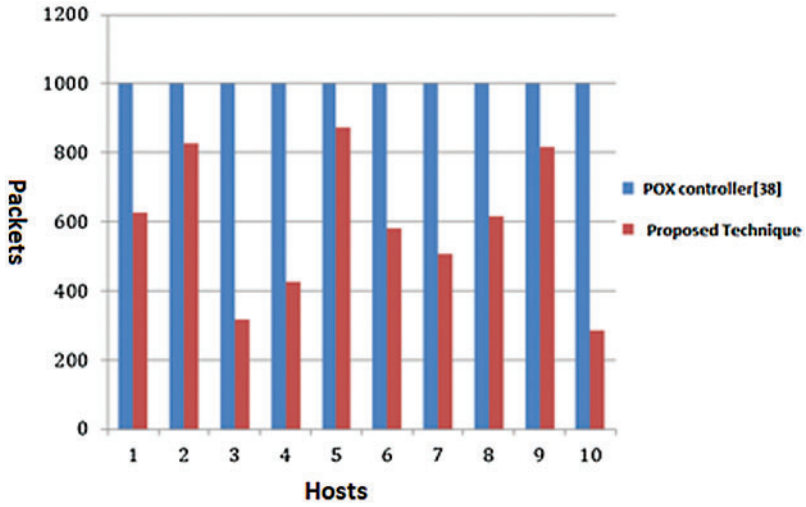

Figure 5: Graph showing ratio of packets to be processed after host movable

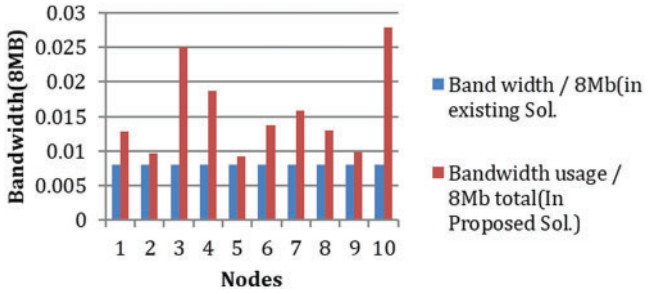

Figure 6: Bandwidth availability in both solutions (8MBs)

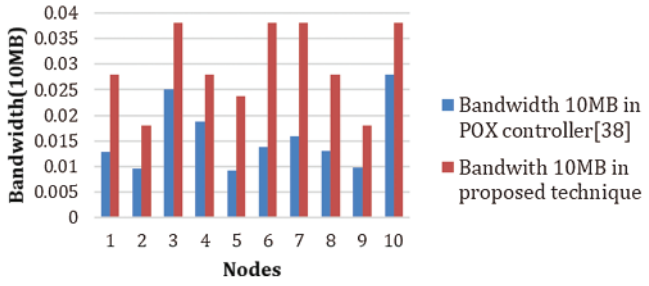

Figure 7: Bandwidth availability in both solutions (10MB)

As the TCP window size is 65535 bytes (or 524280 bits) predefined in IETF RFC 1323, the round value of the average of 10 packets calculated from the host terminal is 0.054 s.

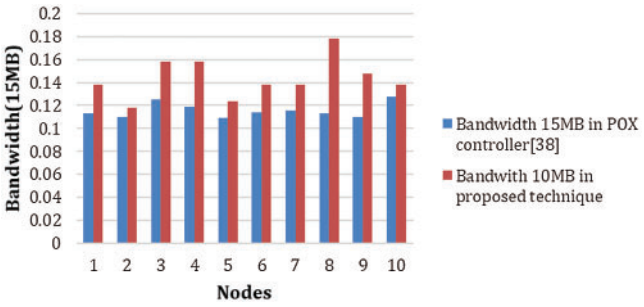

Figure 8: Bandwidth availability in both solutions (15 MB)

So Throughput of existing solution:

$$\text{Throughput} = \frac{524280\ bits}{0.054\ seconds} = 9708888.9 = 9.259\text{Mbps}$$

The saved discarded packets of the existing solution will not take a full round from source host to destination. These packets get forward from their respective switches on which these exist, when the destination host changes its location. So, the number of hops for saved packages is reduced, time to reach the changed destination will also be reduced for these packets. Fig. 9 shows the throughput for 500 s, and Fig. 10 shows the throughput for 1000 s.

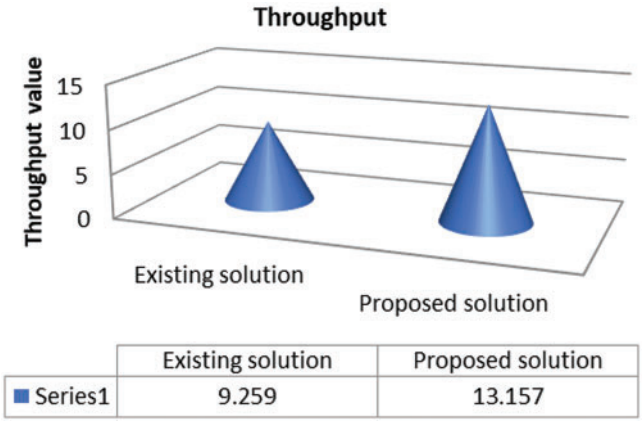

Figure 9: Throughput comparison between existing solution and proposed solution

Here we take the average time of minimum hops of saved packets + time of maximum hops of packets and divide by total hops in the path as:

$$\text{Reducedtime} = \frac{(0.0135 + 0.054)}{4} = 0.016\text{seconds}$$

So through of proposed solution:

$$\text{Throughput} = \frac{TCP\ window\ size\ (Data\ Packets)}{Round\ Trip\ Time\ (RTT) -\ \ Reduced\ time\ of\ saved\ packets}$$

Hence it is:

$$\text{Throughput} = \frac{524280\ bits}{0.054\ seconds - 0.016\ seconds} = 13796842.105 = 13.157\text{Mbps}$$

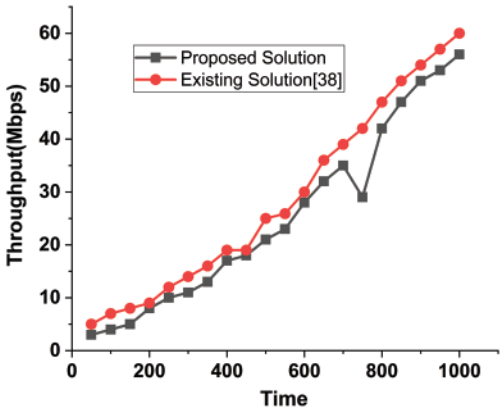

Figure 10: Throughput comparison between existing solution and proposed solution

5 Conclusion

In this paper, we examined the loosing of packets due to moveable hosts in two forms: One is examined as dropped packets when a destination host changes its location while the other is due to discard of the packets in transit state or midway between source and destination. We conclude that our proposed solution has well rectified this issue and produced better evaluation results. We have achieved more network efficiency, saved the bandwidth, protected packets from loss, and our proposed solution has achieved better throughput. In future work, one can utilize the proposed technique for large data centers and wireless SDN network setup as this technique can be utilized in SDN-based cellular networks, as it is cost-effective and a programmed network technique for efficiently monitored networks. The chances of data lost during communication are reduced, and the time-saving approach will lead nicely to the organizational networks through such a programmatically controlled network. Our results show that the bandwidth, packet delivery ratio, and Hop Count are better from the existing technique by 28%. Furthermore, the results show that the packet does not lose and improved 60% from the existing technique, on 8Mb, 10Mb, and 15Mb, the bandwidth is effectively utilized for all values. The network throughput is checked and measured for 500 sec and 1000 sec. The effectiveness of the results show that the proposed technique outperformed the existing approaches in network bandwidth and throughput.

Funding Statement: This research was funded by the Deanship of Scientific Research at Princess Nourah bint Abdulrahman University through the Fast-track Research Funding Program.

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

References

[1] S. Bera, S. Misra and A. V. J. I. I. o. T. J. Vasilakos, "Software-defined networking for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1994–2008, 2017.

[2] M. Karakus and A. J. C. N. Durresi, "A survey: Control plane scalability issues and approaches in software-defined networking (SDN)," *Computer Networks*, vol. 112, pp. 279–293, 2017.

[3] C. Li, Z. Qin, E. Novak and Q. J. I. I. o. T. J. Li, "Securing SDN infrastructure of IoT–Fog networks from MitM attacks," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1156–1164, 2017.

[4] F. R. de Souza, C. C. Miers, A. Fiorese, M. D. de Assunção and G. P. J. J. o. G. C. Koslovski, "Qvia-sdn: Towards QoS-aware virtual infrastructure allocation on sdn-based clouds," *Journal of Grid Computing*, vol. 17, no. 3, pp. 447–472, 2019.

[5] G. Ruggeri, M. Amadeo, C. Campolo, A. Molinaro, A. J. I. T. o. N. Iera and S. Management, "Caching popular transient IoT contents in an SDN-based edge infrastructure," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, 2021.

[6] D. Sinh, L.-V. Le, B.-S. P. Lin and L.-P. Tung, "SDN/NFV—a new approach of deploying network infrastructure for IoT," in *2018 27th Wireless and Optical Communication Conf. (WOCC)*, Hualien, Taiwan, pp. 1–5, 2018.

[7] H. Lu, N. Arora, H. Zhang, C. Lumezanu, J. Rhee and G. Jiang, "Hybnet: network manager for a hybrid network infrastructure," in *Proc. of the Industrial Track of the 13th ACM/IFIP/USENIX Int. Middleware Conf.*, Québec city, QC, Canada, pp. 1–6, 2018.

[8] F. Jamil and D. H. Kim, "Improving accuracy of the alpha–beta filter algorithm using an ANN-based learning mechanism in indoor navigation system," *Sensors*, vol. 19, no. 18, pp. 3946, 2019.

[9] F. Jamil, S. Ahmad, N. Iqbal and D. H. Kim, "Towards a remote monitoring of patient vital signs based on IoT-based blockchain integrity management platforms in smart hospitals," *Sensors*, vol. 8, no. 20, p. 2195, 2020.

[10] F. Jamil, M. A.Iqbal, R. Amin and D. H. Kim, "Adaptive thermal-aware routing protocol for wireless body area network," *Electronics*, vol. 8, no. 1, pp. 47, 2019.

[11] F. Jamil, L. Hang, K. H. Kim and D. H. Kim, "A novel medical blockchain model for drug supply chain integrity management in a smart hospital," *Electronics*, vol. 8, no. 5, pp. 505, 2019.

[12] K. Govindarajan, K. C. Meng, H. Ong, W. M. Tat, S. Sivanand *et al.*, "Realizing the quality of service (QoS) in software-defined networking (SDN) based cloud infrastructure," in *2014 2nd Int. Conf. on Information and Communication Technology (ICoICT)*, Bandung, Indonesia, pp. 505–510, 2014.

[13] A. Gelberger, N. Yemini and R. Giladi, "Performance analysis of software-defined networking (SDN)," in *2013 IEEE 21st Int. Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*, San Francisco, CA, USA, pp. 389–393, 2016.

[14] S. Ahmad, F. Jamil, A.Khudoyberdiev and D. H. Kim, "Accident risk prediction and avoidance in intelligent semi-autonomous vehicles based on road safety data and driver biological behaviours," *J. Intell. Fuzzy Syst*, vol. 38, no. 4, pp. 4591–4601, 2020.

[15] F. Jamil, H. K. Kahng, S. Kim and D. H. Kim. "Towards secure fitness framework based on IoT enabled blockchain network integrated with machine learning algorithms," *Sensors*, vol. 21, no. 5, pp. 1640, 2021.

[16] Y. Zhao, L. Iannone and M. Riguidel, "On the performance of SDN controllers: A reality check," in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, San Francisco, CA, USA, pp. 79–85, 2015.

[17] S. Rowshanrad, V. Abdi and M. J. I. E. J. Keshtgari, "Performance evaluation of SDN controllers: Floodlight and open Daylight," *IIUM Engineering Journal*, vol. 17, no. 2, pp. 47–57, 2016.

[18] M. M. Tajiki, B. Akbari, M. Shojafar and N. J. A. S. Mokari, "Joint QoS and congestion control based on traffic prediction in SDN," *Applied Sciences*, vol. 7, no. 12, pp. 1265, 2017.

[19] H. Yahyaoui, S. Aidi and M. F. Zhani, "On using flow classification to optimize traffic routing in SDN networks," in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–6, 2020.

[20] R. Mohammadi, R. Javidan, M. Keshtgari and N. Rikhtegar, "SMOTE: An intelligent SDN-based multi-objective traffic engineering technique for telesurgery," *IETE Journal of Research*, vol. 139, pp. 1–11, 2021.
[21] A. Tahmasebi, A. Salahi and M. A. J. W. P. C. Pourmina, "A novel feature-based DDoS detection and mitigation scheme in SDN controller using queueing theory," *Wireless Personal Communications*, vol. 117, no. 3, pp. 1–22, 2021.
[22] N. Abdolmaleki, M. Ahmadi, H. T. Malazi, S. Milardo, "Fuzzy topology discovery protocol for SDN-based wireless sensor networks," *Simulation Modelling Practice and Theory*, vol. 79, pp. 54–68, 2017.
[23] Z. Shi, Y. Tian, X. Wang, J. Pan and X. J. C. N. Zhang, "Po-Fi: Facilitating innovations on WiFi networks with an SDN approach," *Computer Networks*, vol. 187, pp. 107781, 2021.
[24] Yazdinejad, A., M. P. Reza, D. Ali, S. Gautam *et al.*, "Cost optimization of secure routing with untrusted devices in software defined networking," *Journal of Parallel and Distributed Computing*, vol. 143, pp. 36–46, 2020.
[25] A. Ali, M. M. Iqbal, H. Jamil, F. Qayyum, S. Jabbar *et al.*, "An efficient dynamic-decision based task scheduler for task offloading optimization and energy management in mobile cloud computing," *Sensors*, vol. 21, no. 13, pp. 4527, 2021.
[26] A. M. El-Shamy, N. A. El-Fishawy, G. Attiya and M. A. J. E. I. J. Mohamed, "Anomaly detection and bottleneck identification of the distributed application in cloud data center using software–Defined networking," *Egyptian Informatics Journal*, 2021. https://www.sciencedirect.com/science/article/pii/S1110866521000013.
[27] L. Bonati, M. Polese, S. D'Oro, S. Basagni and T. J. C. N. Melodia, "Open, programmable, and virtualized 5G networks: State-of-the-art and the road ahead," *Computer Networks*, vol. 182, p. 107516, 2020.
[28] P. Kokkinos, P. Soumplis, A. Kretsis and E. M. Varvarigos, "Network slicing in 5G infrastructures from the edge to the core," in *12th Int. Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, Porto, Portugal, pp. 1–5, 2020.
[29] R. Mohammadi, R. Javidan and M. J. I. J. o. B. -I. C. Keshtgari, "An intelligent traffic engineering method for video surveillance systems over software defined networks using ant colony optimization," *International Journal of Bio-Inspired Computation*, vol. 12, no. 3, pp. 173–185, 2018.
[30] K. Jung, "*Platform-independent Live Process Migration for Edge Computing Applications*," Vancouver, Canada: University of British Columbia, 2021.
[31] M. Alsaeedi, M. M. Mohamad and A. A. J. I. A. Al-Roubaiey, "Toward adaptive and scalable openflow-SDN flow control: A survey," *IEEE Access*, vol. 7, pp. 107346–107379, 2019.
[32] T. Jamal, P. Amaral and K. Abbas, "Flow table congestion in software defined networks," in *ICDS*, Rome, Italy, pp. 57, 2018. https://www.iaria.org/conferences2018/ICDS18.html.
[33] Z. Guo, R. Liu, Y. Xu, A. Gushchin, A. Walid *et al.*, "STAR: Preventing flow-table overflow in software-defined networks," *Computer Networks*, vol. 125, pp. 15–25, 2017.
[34] T.-Y. Chao, K. Wang, L. Wang and C.-W. Lee, "In-switch dynamic flow aggregation in software defined networks," in *2017 IEEE Int. Conf. on Communications (ICC)*, Paris, France, pp. 1–6, 2021.
[35] A. J. I. N. Ghiasian, "Impact of TCAM size on power efficiency in a network of openflow switches," *IET Networks*, vol. 9, no. 6, pp. 367–371, 2020.
[36] J. Yan, X. Liu and D. Jin, "Simulation of a software-defined network as one Big switch," in *Proc. of the 2017 ACM SIGSIM Conf. on Principles of Advanced Discrete Simulation*, Singapore Republic of Singapore, pp. 149–159, 2017.
[37] Rifai, M., H. Nicolas, C. Christelle, G. Frederic *et al.*, "Minnie: An sdn world with few compressed forwarding rules," *Computer Networks*, vol. 121, pp. 185–207, 2017.
[38] J. Park, J. Hwang and K. J. M. I. S. Yeom, "Nsaf: An approach for ensuring application-aware routing based on network qos of applications in sdn," *Mobile Information Systems*, vol. 2019, 2019.
[39] X. Li and W. Xie, "CRAFT: A cache reduction architecture for flow tables in software-defined networks," in *2017 IEEE Symposium on Computers and Communications (ISCC)*, Heraklion, Greece, pp. 967–972, 2017.

[40] Q. Li, N. Huang, Y. Jiang, R. Sinnott and M. Xu, "Scale the data plane of software-defined networks: A lazy rule placement approach," in *2020 IEEE 40th Int. Conf. on Distributed Computing Systems (ICDCS)*, Singapore, Singapore, pp. 366–376, 2020.
[41] J.-P. Sheu, W.-T. Lin, G.-Y. J. I. T. o. N. Chang and S. Management, "Efficient TCAM rules distribution algorithms in software-defined networking," *IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 854–865, 2018.
[42] S. Tamleh, G. Rezaei and J. J. P. L. A. Jalilian, "Stress and strain effects on the electronic structure and optical properties of ScN monolayer," *Physics Letters*, vol. 382, no. 5, pp. 339–345, 2018.
[43] J. Ali and B. H. Roh, "An effective hierarchical control plane for software-defined networks leveraging TOPSIS for end-to-end QoS class-mapping," *IEEE Access*, vol. 8, pp. 88990–89006, 2020.
[44] J. Ali, B. H. Roh and S. Lee, "Qos improvement with an optimum controller selection for software-defined networks," *Plos One*, vol.14, no. 5, pp. 0217631, 2019.
[45] J. M. Schmidt-Engler, L. Blankenburg, B. Błasiak, L. J. Van Wilderen, M. Cho *et al.*, "Vibrational lifetime of the SCN protein label in H2O and D2O reports site-specific solvation and structure changes during PYP's photocycle," *Analytical Chemistry*, vol. 92, no. 1, pp. 1024–1032, 2019.
[46] G. Jain, "Application of SNORT and wireshark in network traffic analysis," *IOP Conference Series: Materials Science and Engineering*, vol. 1119, no. 1, pp. 012007, 2021.