Network Resource Management Optimization for SDN based on Statistical Approach

Mohammed Najm Abdullah
Department of Computer Engineering
University of Technology,
Baghdad, Iraq

Afrah Salman Dawood
Department of Computer Engineering
University of Technology,
Baghdad, Iraq

Ali Kamal Taqi
Department of Computer Engineering
University of Technology,
Baghdad, Iraq

ABSTRACT
Software-Defined Networking (SDN) is a new networking scheme in network technologies in which the data plane and network plane are separated. It can be considered as an umbrella including numerous sorts of network topologies, such as linear, minimal, tree, datacenter topologies and several others, proposed principally to solve problems of diverse nature of applications and different demands for resource management and performance. Where traditional networks would utilize a specific machine, for example, a firewall or load balancer, a Software-Defined Networking conveys an application that uses the controller (like OpenDaylight, Floodlight, OVS, etc.) to oversee information plane conduct. SDN can be implemented on Mininet emulator or NS3 simulator, which are both Linux basis but can also be implemented on other operating systems.

One of important tools in network simulation is Fast Network Simulation Setup (FNSS); which is very useful toolchain that provides the ability of parsing different topologies from datasets or generating them artificially and implement a complete simulation scenario. FNSS supports different programming languages and network simulators using suitable APIs in the target simulator.

In this paper, we proposed a solution to increase the performance of datacenter network topology implemented in python programming language on FNSS toolchain with Mininet emulator. The proposed solution is based on Linear Least Squares method to set different capacities to links based on average delays. This solution has been implemented and tested on two types of SND controllers which are OVS controller and Floodlight controller. Thus, a comparison based on these results has raised between these two controllers. In general, we choose video steaming application played on VLC media player to evaluate our solution. The results show that, as an average, the bandwidth increased from 953Kbps- 1.3Mbps between two hosts to 8.7Mbps- 13.9Mbps according to *iperf* command. The delay has been reduced in 89.6% in OVS controller and in 87.16% in Floodlight controller.

Keywords
Software-Defined Networking (SDN), Fast Network Simulation Setup (FNSS), resource management, OVS Controller, Floodlight Controller, Mininet, Least Square Solution, Datacenter topology.

1. INTRODUCTION
In the last years, Software-Defined Networking attract the attention of both commercial and academic fields since it can be considered as a key improvement in network environments. As mentioned earlier, the main facility of SDN is the separation of data plane and control plane [4]. Software-Defined Networking (SDN) alludes to a method for sorting out computer network implementation. SDN permits the network to be virtualized, giving more noteworthy control and support to traffic engineering [5]. Computer networks today can be seen particularly to accomplish application requirements efficiently: the data plane, the control plane, and the management plane.

The data plane, also known as the forwarding plane, which is related with the hardware devices is responsible for handling incoming packets [5]. The control plane. The control plane and management plane avail the data plane, bears the traffic that the network exists to carry. The management plane is responsible for coordinating the interaction between the control plane and the data plane [5]. All three planes are accomplished in the firmware of routers and switches. SDN decouples these planes by eliminating the control plane from the network hardware and implements it in software on a special controller, which can be thought of as the brain of the network. This implementation enables programmatic access, and accordingly, makes network management much more flexible.

Implementing the control plane in software and eliminating it from the control plane allows and dynamic access and administration and allows efficient management to network resources; since network administrator can change switches' rules without the need to manually configure each switch in the network. Management and admin plane are responsible on setting up the network resources and allows researchers to optimize these resources and improve the efficiency of the network.

In this paper, we propose a procedure to enhance Quality-of-Service (QoS) requirements of Software-Defined Networking (SDN) by parsing implemented solution and topology in python file from FNSS to Mininet. We show the difference between two types of controllers according to this procedure. We also show the difference in quality and delay on chosen video streamed through VLC media player. The rest of the paper is organized as follows: section II describes basic concepts of the main topics in this paper. We introduced some of researches related to resource management in section III. The proposed solution and the algorithm is shown in section IV, section V contains simulated results, section VI shows performance evaluation, and finally, section I includes conclusion.

2. BASICS CONCEPTS
This section provides an overview of the basic concepts and terms used in this paper. Mininet simulation, FNSS toolchain, and Floodlight and OVS controllers will be explained briefly.

A. Mininet
It is a software emulator used to run SDN topologies with a

suitable controller [14]. As known emulator is hardware or software that enables one computer system (called the host) to behave like another computer system (called the guest). Though Mininet emulator supports research, development, learning, testing, etc. Adaptability, appropriateness, intelligence, versatility, sensible, and impart capable models to different teammates are some of attributes that guide the formation of Mininet.

B. OVS Controller [15]

Open Virtual Switch is a generation quality, multilayer virtual switch authorized under the open source Apache 2.0 permit. It is intended to empower huge system robotization through automatic expansion, while as yet supporting standard administration interfaces and conventions (e.g. NetFlow, sFlow, IPFIX, RSPAN, CLI, LACP, 802.1ag). Open vSwitch can operate both as a software-based network switch running within the virtual machine (VM) hypervisors, and as the control stack for dedicated switching hardware; as a result, it has been ported to multiple virtualization platforms, switching chipsets, and networking hardware accelerators. OVS controller is the default switch running with Mininet emulator.

C. FNSS [1]

It is a toolchain (i.e. a set of programming tools that is used to perform a complex software development tasks or to create a software product) that allows network researchers to facilitate the process of setting up the network experiment scenario. The quintessence library provides all the susceptibility to beget the suitable experiment scenario. The transformers allow researchers to export scenarios produced with the quintessence library to NS-2, NS-3, Mininet, Omnet++, Auton etkit, etc.

The scenario is composed of network model and workload. Figure 1 describes the scenario more clearly. After completing the topology configuration, the scenario is released to FNSS to be accomplished and prevailed in the preferred target simulator.

D. Floodlight Controller [18]

Floodlight Controller is a SDN Controller introduced by Big Switch Networks that works with the OpenFlow convention to arrange activity streams in an SDN environment. It can be invaluable for designers, since it offers them the capacity to effectively adjust programming and create applications and is composed in Java programming language. It has the representational state Transfer application program interfaces (REST APIs) that make it less demanding to program interf ace with the item, and the Floodlight site offers coding cases that guide designers in building the item.

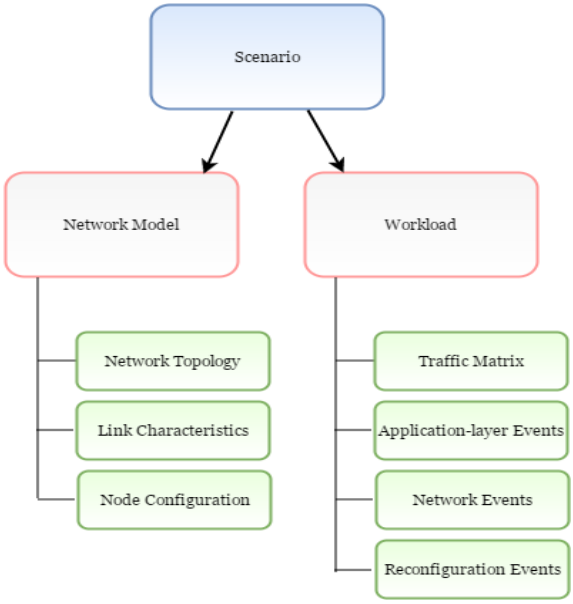

Fig. 1: The Components of FNSS Scenario

3. RELATED WORK

As resource management is a very important factor in any system, software defined networking researchers tried to improve these networks and evaluate the performance of these networks. Reference [7] was the first step towards stringent QoS requirements and manage the handoff between different network technologies, e.g., Wi-Fi and WiMAX. The suggested model provides an optimal results to be implemented on the campus networks. Reference [8] provided an overview to investigate possible enhancements and solutions to enable SDN as future network paradigm. A complete solution was provided to increase the robustness of Ethernet IP networks to the level of carrier-grade and industrial networks with the application of a link-based protection scheme and optimal routing algorithm, combined into a Software Designed Networking solution. Reference [9] summarized resource management mechanisms provided by available SDN approaches based on OpenFlow and exemplary evaluates implementation prospects and challenges. A Ph.D. dissertation in [10] proposed a complete secure SDN architecture that used multiple controllers to confirm the update of flow tables in each switch. The presented results have demonstrated some effectiveness to improve SDN by resource optimizations. The authors in [11] presented the simulation experiment to validate the performance of their framework. Furthermore, they deployed this framework into their physical testbed to proof the feasibility and expandability. Experiment results had shown that an effectively improve on QoS can be achieved for various services using their framework. The paper in [12], proposed a dynamic resource management scheme called EnterpriseVisor engine that managed the distribution of network resources among slices. The authors in [13] presented an SDN-based Wi-Fi data offloading and load balancing algorithms and analyzed the performance of the proposed algorithms under realistic load conditions. Reference [19], investigated and tested the multi-path chunked video streaming with the Open vSwitch (OVS)-enabled Mininet-emulated network testbed. The results demonstrated that multi-path video streaming method can be gainfully applied in the network when the capacity of the main path alone is not enough

to carry the whole incoming packets of video stream and the employed chunk splitting ratio decomposes the incoming packet rate to match the capacity of available paths. The thesis in [20] focused on implementing an optimized frame work that can provide end-to-end QoS guarantee in large scale multi-service provider network.

In summary, we can conclude that our work is one of the first researches that attempts to optimize resource management using both Mininet and FNSS simulators. Although references [19] and [20] are the nearest to our implementation. Our proposal enhances capacity, bandwidth, and delay through transportation of packets.

4. PROPOSED SOLUTION AND ALGORITHM

The resource management problem considered in this paper can be described as follows. Let G (E, L) be a graph of a datacenter topology implemented in python programming language under PULP basis and run in Mininet as a Software-Defined Network with OVS controller. Where, $L = \{l_1, l_2, \dots, l_n\}$ represents vertices and $E = \{e_1, e_2, \dots, e_m\}$ represents edges of the proposed topology. Edges have capacities belonging to the set $C = \{c_1, c_2, \dots, c_p\}$. The goal is to find suitable capacity for each link in the topology regarding the primary delay of the link. Increasing the capacity of the link results in an increased bandwidth of the path which in turn reduces the delay of the path between two hosts.

The issue is that when increase packet size there are more delays. Though, there must be some correlation between the amount of the capacity assigned to the link and packet size. Based on this assumption, the proposed solution is to presume an Integer Linear Programming proposition (ILP) that attempts to identify a list of capacities C assigned to links based on packet size according to statistical regression approach. Regression analysis is a statistical process for estimating the relationships among variables.

The basic idea of the method of least squares is easy to understand. In this work, we based on formulating the least square estimation [17] according to collected data (e.g. delay and bandwidth) from datacenter network topology.

Table 1: Measured Data before Enhancement

X	**1**	**10**	**1**	**10**	**1**	**10**
Y	14.122	14.506	381.568	45.137	1036.240	110.644

$$y = mx + b \quad (1)$$

The above linear equation [17] is going to be the base of our solution.

$$A\bar{c} = \bar{y} \quad (2)$$

We need to solve

$$\bar{c} = A^T \bar{y} \quad (3)$$

$$\begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} M & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum y_i \\ \sum x_i y_i \end{bmatrix} \quad (4)$$

Where c_0 represents b and c_1 represents m. The final equation used in the implementation will be as follows:

$$x = \frac{524.0375 - y}{46.7275} \quad (5)$$

Where x represents the capacity associated with the link l_i according to the average delay of that link, y. This ILP has been implemented on FNSS in python function using PULP library.

The proposed solution has been implemented in simulators according to the following algorithm.

Step 1. Import all necessary libraries (i.e. Mininet, FNSS, PULP, and networkx) in python file
Step 2. Implement suitable datacenter topology.
Step 3. Set edge leaf and core edge links of the topology.
Step 4. Set the weights of the topology.
Step 5. Set the units of the delay to 'ms'.
Step 6. Get the values of delay.
Step 7. Set the capacity of each link according to eq. 5.
Step 8. Set link loads of the topology.
Step 9. Draw the topology with nx library.
Step 10. Export the topology to Mininet
Step 11. Run the topology and test the performance.

5. SIMULATED RESULTS

The proposed algorithm above has been implemented using Mininet emulator, python package, networkx library, PULP library, and FNSS. We implement it with both OVS and Floodlight controllers. The complete graph topology implemented with OVS controller can be displayed in networkx python core library only as shown in Figure 2. The complete graph topology implemented with Floodlight controller can be displayed in both networkx python core library as well as on Floodlight Web UI which shows more flexible graphs.

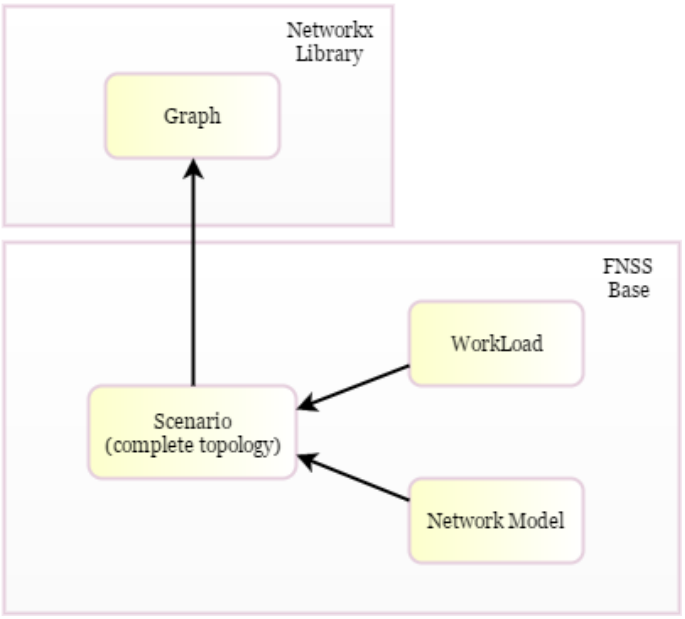

Fig. 2: Plotting the Topology

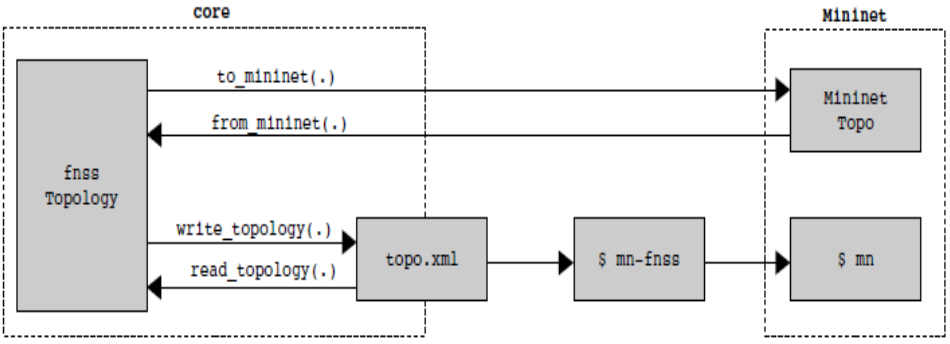

Fig.3: Exporting FNSS scenario to Mininet [16].

In this paper, the preferred target simulator to be used in FNSS is the Mininet, since SDN is the main core of the tests and it is hold with Mininet emulator. Figure 3 shows a block diagram of this exportation process.

All these software are implemented on Dell Inspiron 5559, Intel Core i5, 4GB RAM, 500GB hard drive, and Ubuntu 16.04 operating system. For measuring the performance of the proposition in section IV, we implement our proposed solution on three topologies of datacenter networks, which are two_tier_topology, three_tier_topology, and fat_tree_topology. We test this implementation on video stream with VLC media player. The properties of the tested video are of 360x240 pixels (i.e. 2.00 MByte) of 26 second and 720x480 pixels (i.e. 5.00 MByte) of 30 second. It should be noted that we implement our proposal based on using only one controller (i.e. we did not use multiple controllers).

First of all, we will show the simulation results of OVS controller. This controller can support tier topologies with one core only according to the available hardware resources. The topologies of the implementation are shown in Figure 4. We test three lengths of ping packets on each of these topologies; 1KBytes, 24KBytes, and 64.4KBytes. For each packet length, the test has been done for 10 packets and results before and after implementation are shown in Tables 1, 2, and 3.

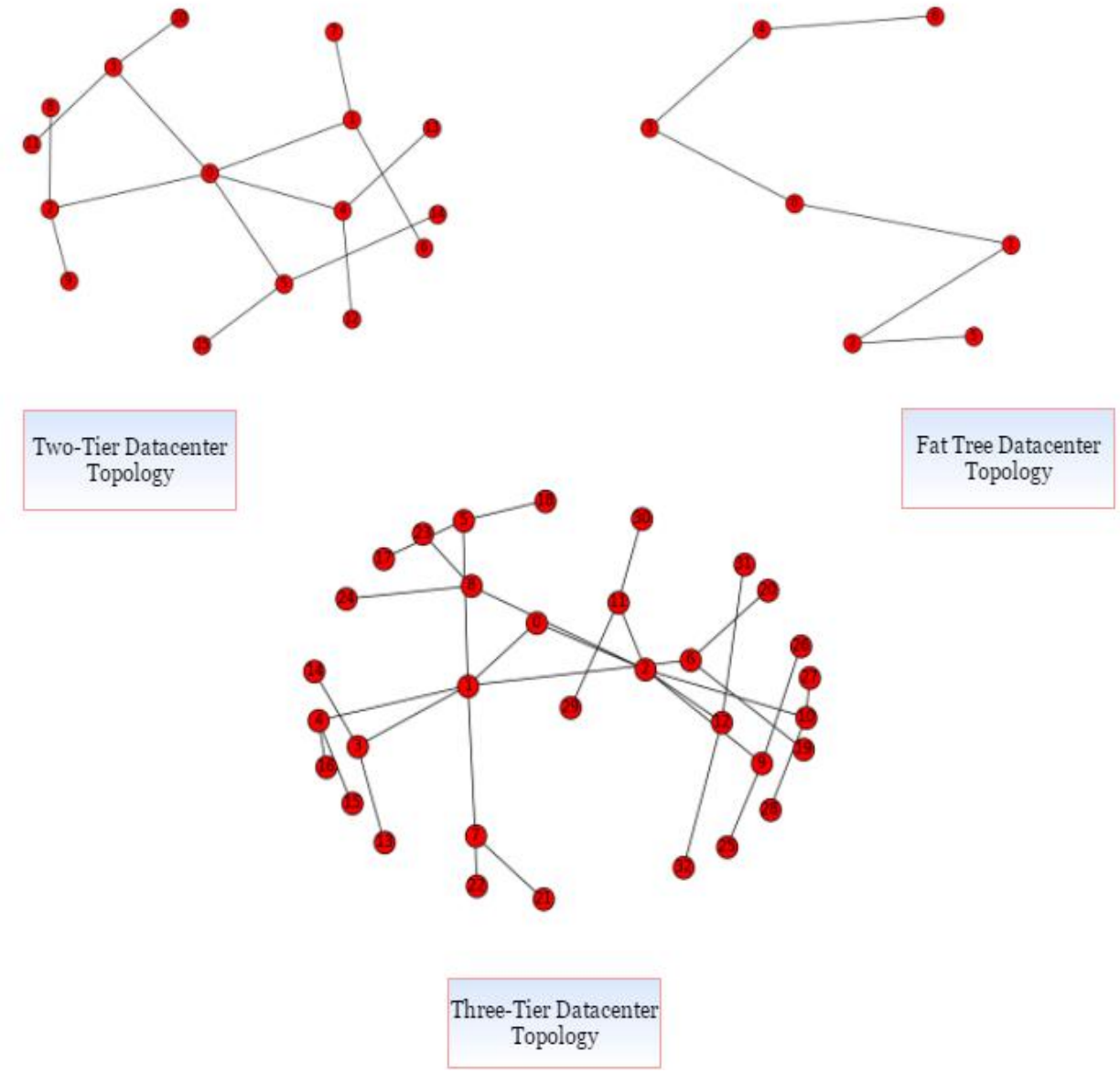

Fig. 4: Implemented Network Topologies

Table 1: OVS Simulation Results of Three Tier Datacenter Topology

Packet Size	Average Delay in (ms)	
	Before Using The Proposed Solution	*After Using The Proposed Solution*
1 Kbyte	17.712	8.78
24 Kbyte	385.549	41.409
64.4 Kbyte	1037.191	100.038

Table 2: OVS Simulation Results of Two Tier Datacenter Topology

Packet Size	Average Delay in (ms)	
	Before Using The Proposed Solution	*After Using The Proposed Solution*
1 Kbyte	8.831	9.048
24 Kbyte	371.503	41.283
64.4 Kbyte	1025.229	100.157

Table 3: OVS Simulation Results of Fat Tree Datacenter Topology

Packet Size	Average Delay in (ms)	
	Before Using The Proposed Solution	*After Using The Proposed Solution*
1 Kbyte	12.962	22.19
24 Kbyte	375.67	57.92
64.4 Kbyte	1029.848	120.411

Secondly, we will show the simulation results of Floodlight controller. This controller can support multi-core tier topologies. The topologies' implementation are shown in Figures 5, 6, and 7. We use the same suggestions of the first case; Tables 4, 5, and 6 shows almost same results. We can conclude that the larger the packet size the more efficient results and the smaller delay values we get, and vice versa.

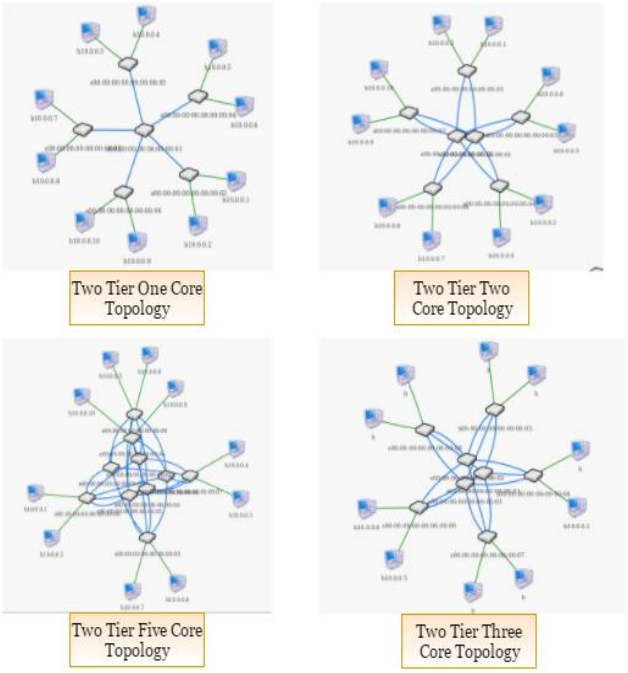

Fig. 5: Two Tier Datacenter Topologies using Floodlight Controller

Table 4: Simulation Results for Two Tier Datacenter Topology Using Floodlight Controller

Packet Size	Average Delay in (ms)	
	Before Using The Proposed Solution	*After Using The Proposed Solution*
1 Kbyte	19.721	15.614
24 Kbyte	370.901	55.522
64.4 Kbyte	1038.980	113.702

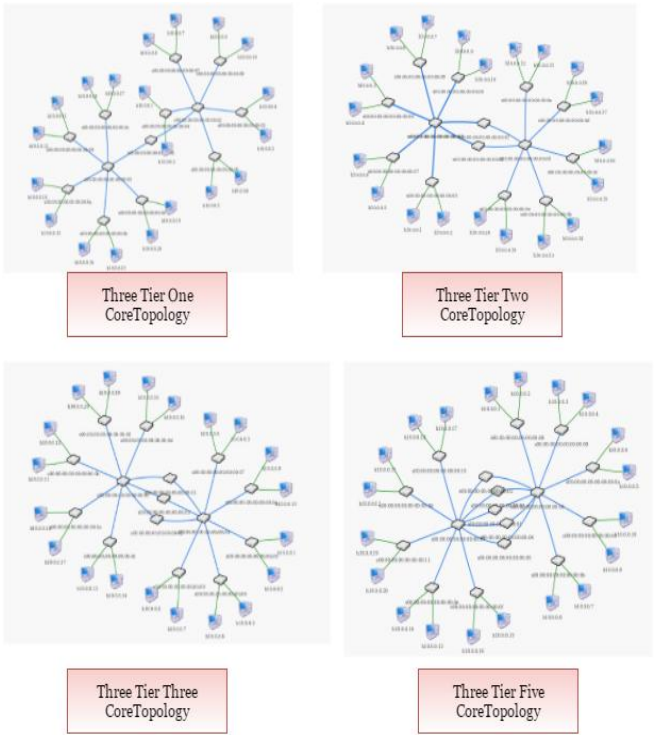

Fig. 6: Three Tier Datacenter Topologies using Floodlight Controller

Table 5: Simulation Results for Three Tier Datacenter Topology Using Floodlight Controller

Packet Size	Average Delay in (ms)	
	Before Using The Proposed Solution	*After Using The Proposed Solution*
1 Kbyte	23.689	17.679
24 Kbyte	388.254	57.226
64.4 Kbyte	1044.579	120.094

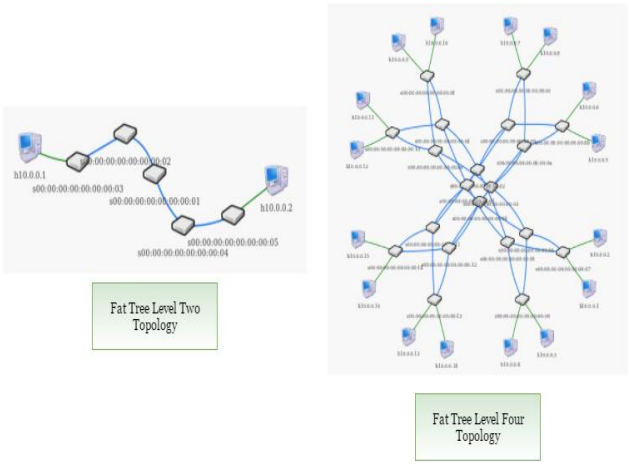

Fig. 7: Fat Tree Datacenter Topologies using Floodlight Controller

Table 6: Simulation Results for Fat Tree Datacenter Topology Using Floodlight Controller

Packet Size	Average Delay in (ms)	
	Before Using The Proposed Solution	*After Using The Proposed Solution*
1 Kbyte	17.544	16.943
24 Kbyte	389.022	50.973
64.4 Kbyte	1047.974	109.233

6. PERFORMANCE EVALUATION

From simulated results in section V, we can show the performance evaluation of the three datacenter topologies for both SDN controllers.

The performance of OVS controller can be described in Figures 8, 9, and 10. We can see the improvement percentage of reducing the delay in 89.6%. As a result of decreasing delay, the bandwidth increased from 950 Mbps- 1.82 Mbps to 10.7 Mbps- 13.5 Mbps after optimization.

The performance of Floodlight controller can be shown in Figures 11, 12, and 13. We can see the percentage of reducing the delay in 87.16%. As a result of decreasing the delay, the bandwidth increased from 952Kbps- 1.7Mbps to 9.6Mbps- 11.3Mbps after optimization. The efficiency of our solution can be shown on played video streams in Figures 14 and 15.

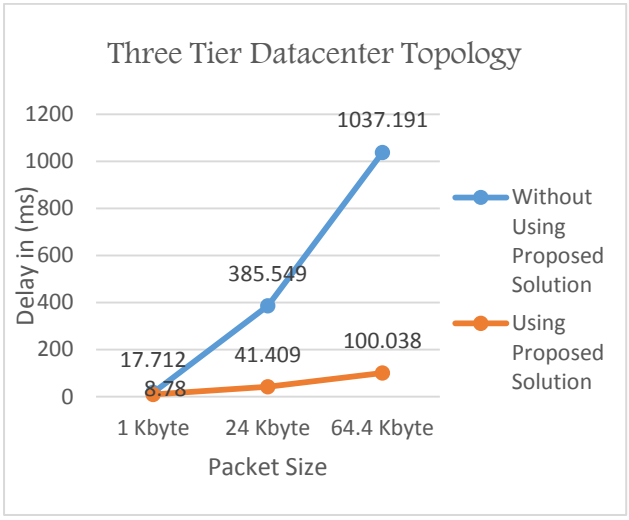

Fig. 8: Performance Evaluation for Three Tier Datacenter Topology Using OVS Controller

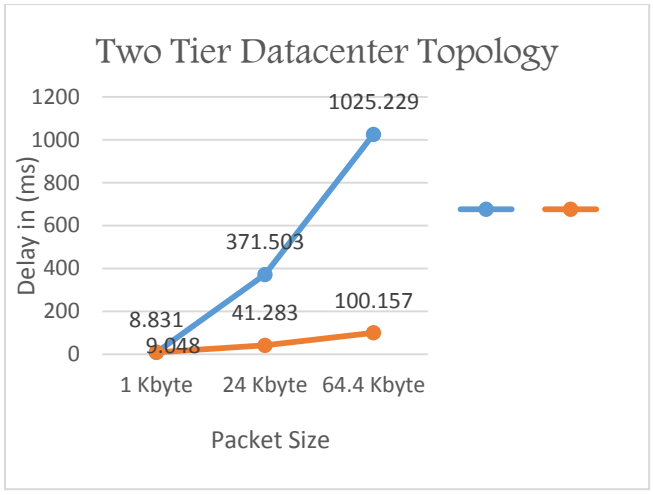

Fig. 9: Performance Evaluation for Two Tier Datacenter Topology Using OVS Controller

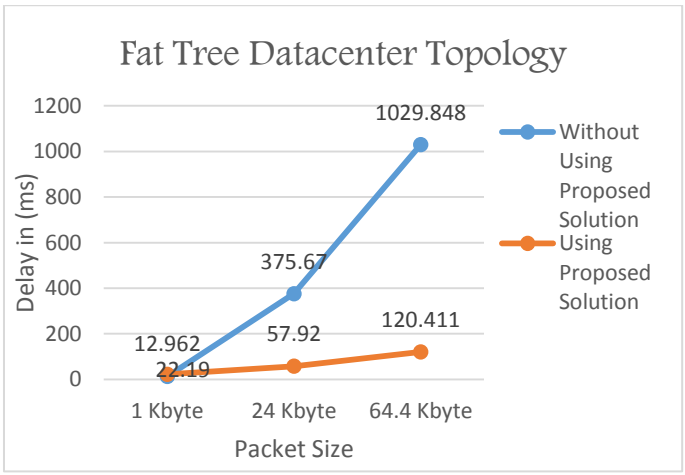

Fig. 10: Performance Evaluation for Fat Tree Datacenter Topology Using OVS Controller

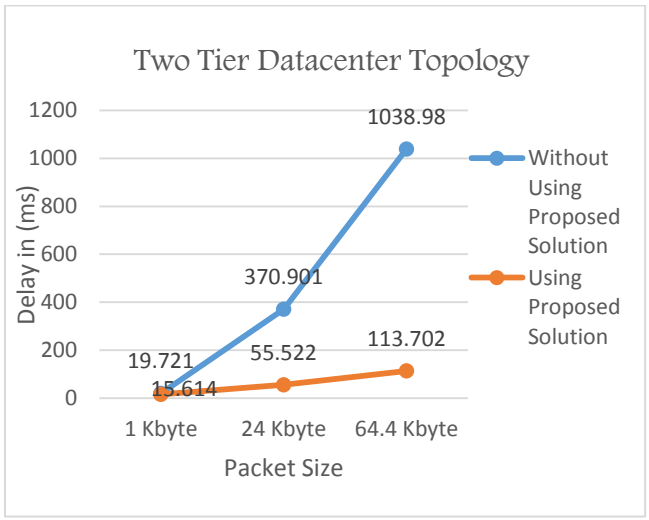

Fig. 11: Performance Evaluation for Two Tier Datacenter Topology Using Floodlight Controller

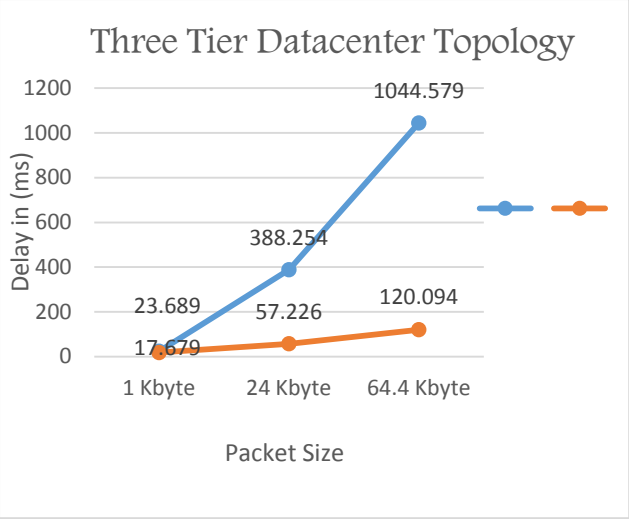

Fig. 12: Performance Evaluation for Three Tier Datacenter Topology Using Floodlight Controller

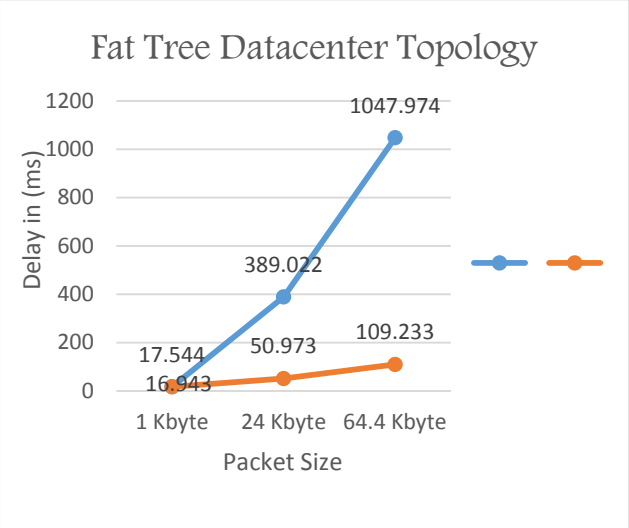

Fig. 13: Performance Evaluation for Fat Tree Datacenter Topology Using Floodlight Controller

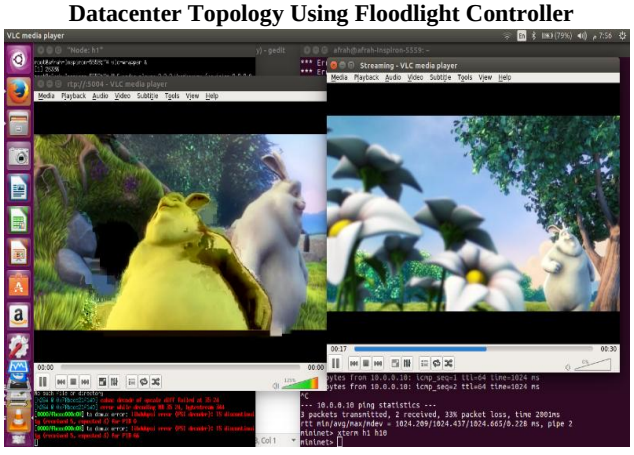

Fig. 14: Efficiency of Video Streaming Before Using the Proposed Solution

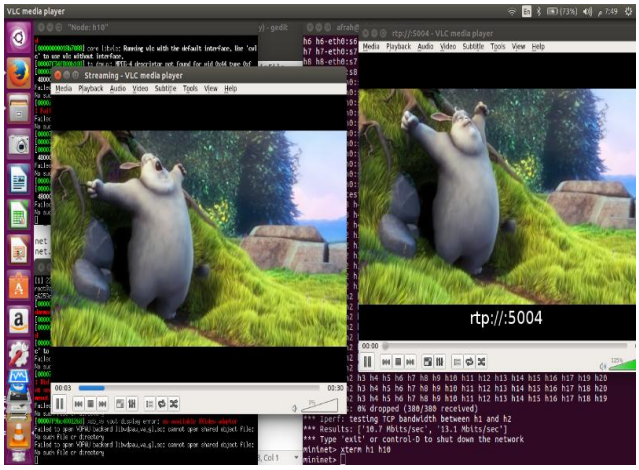

Fig. 15: Efficiency of Video Streaming After Using the Proposed Solution

7. CONCLUSION

The Mininet, FNSS, networkx, PULP, and (OVS or Floodlight) controller platforms has been merged to provide full network scenario with performance evaluation results. This network scenario has been tested with ping and iperf commands on three datacenter topologies for two video streams of 2.00 MByte and 5.00 Mbyte transmitted between selected hosts and played on VLC media player with both OVS and Floodlight controllers. From these results, we can conclude that the difference between two tier and three tier topologies is indiscernible and delays in both has been reduced as much as possible where as in fat tree, delays has not been reduced to minimum since fat tree topology is less efficient and thus two and three tier topologies are most widely used topologies in datacenter networks. We also can conclude that the larger the packet length in ping command the more efficient results and the smaller delay values we get, and vice versa.

From previous results, a difference between OVS and Floodlight controllers arise. We can see that OVS controller, which is the default Mininet controller can be used more efficiently for light topologies, whereas Floodlight controller can be used for more complex topologies; thus it can be said that Floodlight controller can maintain sustainability over OVS controller. Moreover, Floodlight controller has more suitable graphics and display than that in OVS controller.

8. REFERENCES

[1] L. Saino, C. Cocora, and G. Pavlou, A Toolchain for Simplifying Network Simulation Setup, in *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques (SIMUTOOLS '13)*, Cannes, France, March 2013.

[2] L. Saino, C. Cocora, and G. Pavlou, A Toolchain for Simplifying Network Simulation Setup, SIMUTools 2013 March 5–7, Cannes, France, Copyright 2013 ICST, ISBN.

[3] D. Tuncer, M. Charalambides, S. Clayman, and G. Pavlou, Adaptive Resource Management and Control in Software Defined Networks, IEEE transactions on network and service management, vol. 12, no. 1, march 2015.

[4] D. Tuncer, M. Charalambides, S. Clayman, and G. Pavlou, On the Placement of Management and Control Functionality in Software Defined Networks, CNSM ManSDN/NFV paper, @2015 IFIP.

[5] P. A. Morreale, J. M. Anderson, Software Defined

Networking: Design and Deployment, © 2015 by Taylor & Francis Group, LLC.

[6] R. Mijumbi, J. Serrat, J. R.-Loyolay, N. Boutenz, F. D. Turckz and S. Latre´, Dynamic Resource Management in SDN-based Virtualized Networks, ManSDN/NFV Full Paper, ISBN 978-3-901882-67-8, 10th CNSM and Workshop ©2014 IFIP.

[7] F. Ongaro, Enhancing Quality Of Service In Software-Defined Networks, Department of Computer Science and Engineering Master Degree in Computer Engineering, 2013-2014.

[8] B.J Asten, Increasing Robustness Of Software-Defined Networks, Fast Recovery Using Link Failure Detection and Optimal Routing Schemes, master of science thesis, Network Architectures and Services Group, Delft University of Technology, 2014.

[9] T. Zinner, M. Jarschel, A. Blenk, F. Wamser, and W. Kellerer, Dynamic Application-Aware Resource Management Using Software-Defined Networking: Implementation Prospects And Challenges, funded by the BMBF (Project ID 16BP12308), © 2014 IEEE.

[10] H. Li, Resource Optimizations in Software Defined Networking, Graduate Department of Computer and Information Systems University of Aizu, Ph.D. 2015.

[11] C. Xu, B. Chen, and H. Qian, Quality of Service Guaranteed Resource Management Dynamically in Software Defined Network, Journal of Communications Vol. 10, No. 11, November 2015.

[12] J. CHEN, Y. MA, H. KUO, C. YANG, AND W. HUNG, Software-Defined Network Virtualization Platform for Enterprise Network Resource Management, IEEE transactions on Emerging Topics in Computing, Date of Publication 22 September 2015; date of current version 8 June 2016. Digital Object Identi_er 10.1109/TETC.2015.2478757.

[13] X. Duan, A. M. Akhtar, and X. Wang, Software-Defined Networking-Based Resource Management: Data Offloading with Load Balancing in 5g HetNet, Duan et al. EURASIP Journal on Wireless Communications and Networking (2015) 2015:181.

[14] A.S. Dawood, M.N. Abdullah, A Survey and a Comparative Study on Software-Defined Networking, International Research Journal of Computer Science (IRJCS) ISSN: 2393-9842, issue 08, Volume 3 (August 2016).

[15] http://openvswitch.org/

[16] L. Saino, Fast Network Simulation Setup, Lab 1: AIMS conference 2014, Communications and Information Systems Group, Department of Electrical and Electronics Engineering, University College London.

[17] D. Cherney, T. Denton, and A. Waldron, Linear Algebra, Edited by Katrina Glaeser, Rohit Thomas and Travis Scrimshaw First Edition. Davis California, 2013.

[18] https://www.sdxcentral.com/sdn/definitions/sdn-controllers/open-source-sdn-controllers/what-is-floodlight-controller/

[19] P. M. Thet, P. Panwaree, and C. Aswakul, Design and Functionality Test of Chunked Video Streaming Over Emulated Multi-Path OpenFlow Network, ResearchGate, Conference Paper · June 2015.

[20] B. Sumanth, Designing an Openflow Controller for Data Delivery with End-to-End QoS Over Software Defined Networks, Master of Technology, in Department of Computer Science and Engineering Indian Institute of Technology, Kharagpur, May 2016.