Grenze International Journal of Engineering and Technology, January Issue

Performance Benchmarking of Software Defined Networks: A Study of Ryu Controller with D-ITG

Mrs. Medha N Kulkarni[1] and Dr. Jagdish W Bakal[2]
[1,2]Computer Engineering Department, Pillai HOC College of Engineering and Technology Mumbai, India
Email: mnkulkarni75@gmail.com, jwbakal@mes.ac.in

***Abstract*— Software Defined Networking (SDN) is a novel paradigm in the networking industry that is implementing redefined goals of traditional networks. With SDN, inflexibility and complex networks will be replaced by a new flexible and programmable network. In SDN, the data plane and control plane are kept apart and the traditional network constraints are reduced. The SDN controllers like NOX, POX, ONOS, OpenDaylight, Floodlight, RYU etc. are acting as the network brains and oversee its operations. The forwarding decisions are sent by the controller to switches and routers.**

The forwarding of the packets is done by switches. Network traffic generators are widely used in networking research to verify results and evaluate the impact of novel protocols and network features and algorithms.

In this research paper, Mininet is used for SDN emulation. The performance is evaluated for different types of network traffic. D-ITG (Distributed Internet Traffic Generator) is used for traffic generation. A custom topology is created in Mininet using Python. The network performance metrics (bandwidth, throughput, round-trip time, jitter, and packet loss) are evaluated for a variety of traffic types, including TCP, UDP and VoIP.

***Index Terms*— Software Defined Networking, RYU, Performance evaluation, iperf3, DITG, Mininet.**

I. INTRODUCTION

The exponential usage of internet and high demand of flexibility and agility are found as the current computational needs. The traditional networks are inadequate to provide these computational needs. Network configuration in accordance with predetermined policies and reconfiguration in reaction to failures are the two difficult aspects of managing IP networks. Also, vertical integration is followed in traditional networks, in which data and control levels are merged rather than their separation.

Dynamic and adaptable network configurations are made possible by SDN. It is a new pattern or network architecture that offers a centralized approach to network administration. With an SDN, no hardware changes will be necessary to alter network behaviour and functionality.

Alternatively, the network administrator can accomplish this by making quick changes to the source code or software [1].

SDN has also been put forth as a way to solve the problems associated with network optimization and administration. In this study, we first present the SDN architecture and discuss the performance measures of SDN for variety of traffic.

Grenze ID: 01.GIJET.11.1.264

A. Software Defined Networking

Through a higher-level abstraction, Software-Defined Networking (SDN) gives network managers the ability to control network services. This is accomplished by isolating the underlying system that directs packets to the selected destination (data plane) from the system that makes decisions about traffic routing (control plane) [1]. SDN is a dynamic, controllable, economical, and flexible architecture designed to work with today's bandwidth-intensive applications. The network's architecture of SDN provides features like: Directly programmable, Agile, Centrally managed, Configurable by program, Built on open and vendor-neutral standards, and Not dependent on network system manufacturers like conventional networks. One essential component of the SDN architecture is the OpenFlow protocol [1]. Protocols such as the OpenFlow protocol are used to implement communication between SDN switches and the controller. The SDN controller connects the network devices, check the current status of network, and implement forwarding instructions to the control messages defined by the OpenFlow. Additionally, OpenFlow offers thorough and adaptable traffic control.

SDN broke the vertical integration and developed a new architecture that centralizes the control of network and lowers the complexity of network management. In this manner, software is used in a single location to control numerous network devices. In actuality, SDN has offered comprehensive intelligent control in software-defined networks. With SDN applications, network managers may quickly monitor, secure, and optimize the network resources. SDN's flexibility, which sets it apart from older networks, is its most important characteristic. Separation of the data plane and control plane is the secret to its achievement.

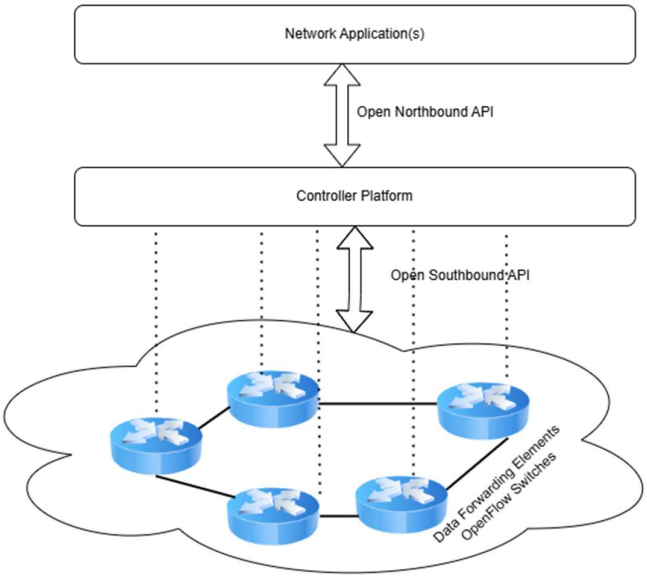

Figure 1 SDN Architecture

The SDN architecture is depicted in Figure 1. The application layer, control layer, and infrastructure layer (data layer) are the three layers that make up SDN architecture, as shown in the Figure 1. The common network applications or features that businesses utilize are found in the application layer. These applications include firewalls, load balancing, and intrusion detection systems. Traditional networks employ specialized modules for load balancing or firewalling. In SDN, the centralized controller that oversees data plane management takes the place of the discrete equipment.

Southbound and Northbound APIs are used for communicating through the three layers of SDN. The brain of SDN is the controller. It resides in the control plane of SDN. The controller is in charge of network flow and policy management. The infrastructure layer of the network contains the actual switches. Applications communicate with the controller via the Northbound interface. The controller and switches communicate with each other using the Southbound interface. One example of a southbound protocol is the OpenFlow protocol.

In the section II of paper discusses Literature Review. Section III describes the methodology used while simulating SDN. It includes which is called Mininet and its characteristics and RYU controller and D-ITG Traffic Generator. Section IV contains the discussion about the implementation of the testbed. Section V deals performance evaluation of SDN for different types of traffic. Section VI conclude the paper.

II. LITERATURE REVIEW

It was discovered that the network architecture needs to be changed from the ground up. The new, evolving network infrastructure known as SDN separates the control plane from the forwarding plane. The authors of this study have introduced SDN and discussed its architecture, applications, various controllers, comparisons, and limits [2]. SDN has given the network flexibility and programmability by separating the control plane from the data plane. The key network component that makes it possible to establish network policies is the SDN controller. The performance of new and various SDN controllers has been compared by the writers in the paper [3]. The conventional network is divided into a programmable data plane and a centralized control plane in SDN. The controller enhances network performance and supplies switches with flow. Therefore, the SDN controller's performance is directly impacted by the controller's performance. Authors of paper [4] have provided a thorough qualitative assessment of various SDN controllers used. Measures of quantitative performance are also used. Three benchmarking tools are utilized to compare nine of the 34 controllers that have been categorized and compared.

In [5], two parameters; throughput and latency have been evaluated in different topologies; Linear, Single, Tree, SDN-SAT, Dumbbell, and DCN in two controllers namely POX and Ryu. The reference controller of Mininet has been used to evaluate the throughput and packet loss in TCP and UDP traffic in three topologies; linear, single and custom tree topology. They have evaluated the parameters in different topologies. Also, they have not evaluated other network parameters like jitter and RTT.

III. METHODOLOGY

A. Mininet

It is a network emulator that creates a system of hosts, switches, controllers and virtual connections. Software CLI (Command Line Interface) and API (application program interface) allows for simple system connectivity. Mininet is commonly used because it is easy to set up, support custom topologies and bundle sending. It also helps to run real projects that are accessible on Linux, PCs, servers, and virtual machines. It has ability to share and recreate content which are easy to use, open source and constantly evolving.

Mininet runs real code, including the real OS kernel, network stack and common network applications. It uses process-based virtualization to simulate entities on a single OS kernel. As a result, a project that functions properly in Mininet can typically proceed straight to real-world networks made out of hardware devices. Without any changes, the code that has to be built for Mininet may also function in a real network. Large networks with a lot of virtual hosts and switches are supported [9, 10, 11].

Mininet is popularly used because of its features like flexibility, interactivity, scalability, realistic and shareable. The network topologies, typically made up switches, controllers, hosts and the links. Topologies can be easily created using MiniEdit software, a feature of Mininet. Both conventional routing and SDN utilizing the OpenFlow protocol are supported by the Mininet simulator. As a result, the SDN architectures could be developed and put into use in a safe virtual setting for research.

B. Distributed Internet Traffic Generator (DITG)

A software called Distributed Internet Traffic Generator (D-ITG) generates traffic according to patterns established by the stochastic processes of packet size (PS) and inter-departure time between packets (IDT). Constant, uniform, exponential, Pareto, Cauchy, normal, Poisson etc. are among the many different types of probability distributions that are available. Additionally, D-ITG simulates the sources of a number of protocols, including VoIP (G.711, G.723, G.729, RTP), TCP, UDP, ICMP, DNS, and Telnet. D-ITG evaluates throughput, jitter, packet loss assessment, one-way delay (OWD), and round-trip time (RTT). Additionally, it allows the TOS(DS) and TTL packet fields to be set. D-ITG makes it possible to keep retrievable data on both the sender and recipient sides.

The DITG has three essential components.

- ITGSend is the sender component of the D-ITG traffic generation platform. It operates in three different modes: single flow mode - generates one flow, multiple flows mode - generates a set of flows and daemon mode - ITGSend remotely controlled by ITGManager using the ITGApi.
- ITGRecv always works as a concurrent daemon. Both ITGSend and ITGRecv can store information either locally or remotely by using the log server ITGLog. Using ITGDec The log file is decoded.
- ITGLog is a “log server”, running on a different host than ITGSend and ITGRecv. It contains the log information from multiple senders and receivers

C. Ryu Controller

Ryu is a one of the controllers used in Software Defined Networking. It is based on various components. Developers can create a variety of network management and control applications in Python with the help of RYU's well-defined API. RYU is compatible with a variety of protocols for network device management, including OpenFlow (extensions 1.0, 1.2, 1.3, 1.4, 1.5), Netconf, OF-config etc. Nicira extensions are also fully supported by Ryu. The Apache 2.0 license makes all of the code freely accessible.

IV. TESTBED IMPLEMENTATION

While studying the performance of SDN with the traffic generation tools like DITG, a custom SDN topology is created in Mininet, using Python. Mininet is installed inside virtual box. Table 6.5 describes the detailed specification of the testbed.

TABLE I. REQUIREMENTS OF TESTBED ENVIRONMENT

Sr No	Name	Specification
1	Operating System	Ubuntu 20.04.1 LTS
2	Ryu controller [Ryu]	Version 4.30
3	Mininet Emulator [Mininet]	Version 2.3.0
4	OpenFlow Protocol [OpenFlow]	Version 1.3

Topology

A custom topology is implemented in python. The topology consists of 2 switches connected to each other and 3 hosts connected to each switch as shown in Figure 2. The Ryu Controller is used as in this emulated network.

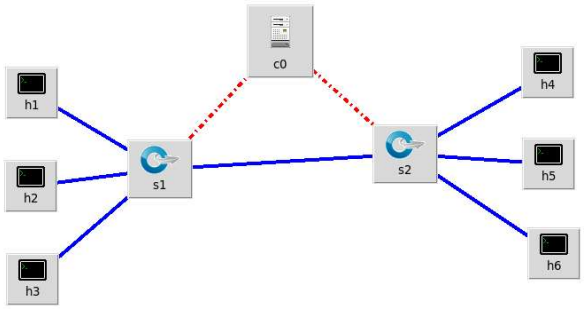

Figure 2. Custom Topology

V. RESULTS AND ANALYSIS

Mininet is used to set up the topology. Ryu Controller is assessed and various network metrics are studied. DITG traffic generator is used to generate the traffic for SDN. The parameters affecting the network performance are observed through the experiments.

The metrics we consider in this study are average delay, average jitter, average bitrate and average packet rate. The said metrics are considered for the traffic – TCP, UDP, VoIP, Telnet and DNS.

In this study in order to evaluate performance of data transfer can be measured using various metrics, including:

- Total Time - It's the total duration taken for the complete data transfer process. It considers from the initiation of the transfer to its completion.
- Average Bitrate - It is the average rate at which bits are transmitted over a specific period of time. It measures the average data transfer speed and is expressed in bits per second (bps) or kilobits (Kbps) or megabits (Mbps).
- Average Packet Rate - It measures the average number of packets transmitted per unit of time. It provides information about the frequency of packet transmission.

The log file is generated at both client and server side. It represents the average jitter and delay, which indicate how long it will take to reach the maximum. It also has the average bitrate and packet rate to assess network performance. Using the D-ITG tool, we sent numerous packets to various hosts, watched the flows, and created a graph showing the various parameters influencing network performance. We produced a graph for various

metrics after analyzing the data generated by DITG tool. We will take into account each of the aforementioned parameters for different kinds of traffic. X-axis represents the type of traffic and Y-axis represents the values of delay, jitter, bitrate and packet rate calculated for different packets in seconds.

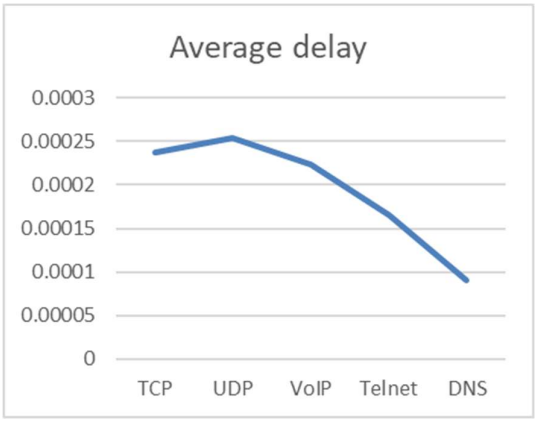

Figure 3 Average delay

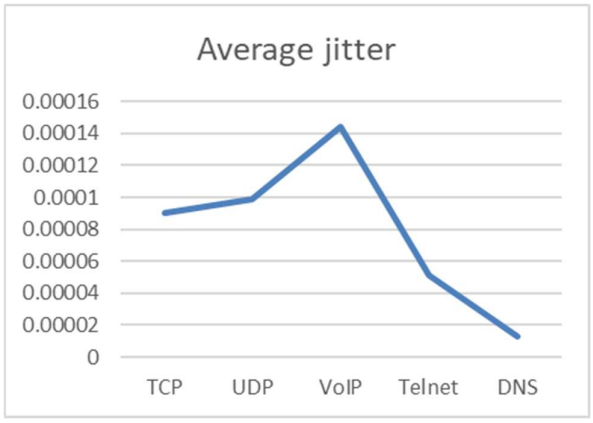

Figure 4 Average jitter

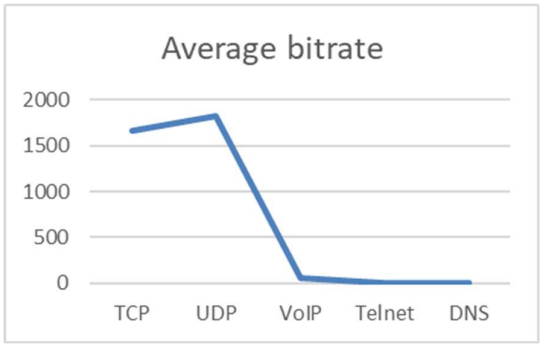

Figure 4 Average bitrate

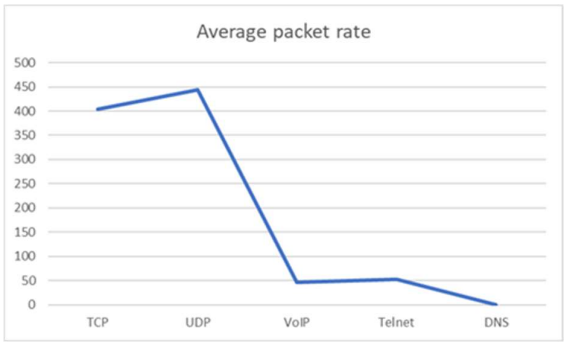

Figure 5 Average packet rate

VI. Conclusion

In this paper, we discussed the performance of Software Defined Networks (SDN) for different variety of traffics. The SDN network topology is emulated in Mininet along with Ryu controller. A custom topology is created for experimentation. The DITG traffic generator is used for generating different types of traffic. We marked the performance of SDN for different types (TCP, UDP, VoIP, Telnet and DNS) of traffic. The output generated by DITG are stored on log archives. The log files include various metrics like average delay, jitter, packet rate and bitrate. The values for various metrics are studied and represented the graph for each metric. This study provides insights on various networking metrics. The average jitter, bitrate and packet rate decreased for VoIP, Telnet and DNS traffic compared to TCP and UDP. This study helps in designing and development of SDN applications like routing, load balancing and testing such applications in a real-like environment.

References

[1] Diego Kreutz, Fernando M. V. Ramos, Paulo Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, Steve Uhlig, “Software-Defined Networking: A Comprehensive Survey”

[2] Elazim NMA, Sobh MA, Bahaa-Eldin AM. “Software defined networking: attacks and countermeasures” in 2018 13th International Conference on Computer Engineering and Systems (ICCES). 2018;555-567

[3] Dr. Bhargavi Goswami, “Software Defined Network, Controller Comparison”, Journal of Innovative Research in Computer and Communication Engineering, Vol.5, Special Issue 2, April 2017

[4] Chaymae EL KHALFI, Abderrahim EL QADI, Hamid BENNIS, “A Comparative Study of Software Defined Networks Controllers”, ICCWCS'17, November 14–16, 2017, Larache, Morocco © 2017 Association for Computing Machinery. ACM ISBN 978-1-4503-5306 9/17/11

[5] Liehuang Zhu, Md Monjurul Karim, Kashif Sharif, Fan Li, Member, IEEE , Xiaojiang Du, Mohsen Guizani, “SDN Controllers: Benchmarking & Performance Evaluation”
[6] Ali, J., Lee, S. and Roh, B. “Performance Analysis of POX and Ryu with Different SDN Topologies” Proceedings of the 2018 International Conference on Information Science and System - ICISS '18. pp 244-249, Apr, 2018, Jeju, Republic of Korea.
[7] M. T. Naing, T. T. Khaing, and A. H. Maw, “Evaluation of TCP and UDP Traffic over Software-Defined Networking,” 2019 International Conference on Advanced Information Technologies (ICAIT), 2019.
[8] Zahra Askarinejadamiria et al “Comparative Study of Data Transfer in SDN Network Architecture in IoT” International Journal of Web Research, Vol. 6, No. 1, Winter- Spring, 2023
[9] Mininet: An Instant Virtual Network on your Laptop (or other PC), available online: http://mininet.org/, access on April 2019.
[10] Introduction to Mininet - mininet/ mininet wiki – GitHub, available online: http:// github.com/mininet/mininet/wiki/ April 2019.
[11] K. Kaur, J. Singh, and N. S. Ghumman, "Mininet as software defined networking testing platform," in International Conference on Communication, Computing & Systems (ICCCS), 2014, pp. 139-42.