IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

NETWORK TRAFFIC PARAMETER ANALYSIS AND HISTORICAL DATASET RELATIONSHIP

[1]Veeru, [2] Rishi Raj Dwivedi, [3]Somveer Singh, [4]Vikash Kundu

Dept. of Computer Engineering

Army Institute of Technology, Pune, India

Abstract: SDN (Software-Defined Networks) is a novel concept proposed by a research institute to achieve a specific goal in Next-Generation Networks (NGN). For operational purposes, SDN removes control aircraft and a plane of data from the main network equipment. The control aircraft consists of a central control unit known as the SDN controller, which operates as a Network Operating System (NOS). The data plane is responsible for transferring only data packets that are controlled by the central controller and are available within network devices. OpenFlow is a description of the first generation of SDN defined by the Open Networking Foundation. It is one of the most popular uses of SDN (ONF). In this paper, network building based on OpenFlow is compared to conventional network construction. The design and operation of the OpenFlow-based network is also implemented. Performance analysis was performed by analysing network performance based on bandwidth usage, packet transfer rate, back-to-back distribution delays, and the high acquisition of Mininet, a model network model, used to build all OpenFlow-enabled issues.

***Index Terms* – SDN, OpenFlow, Mininet, Token Bucket Algorithm**

I. INTRODUCTION

Computer networks are constantly evolving, and network researchers and engineers from various companies are constantly proposing new ideas. The main purpose of creating new computer network emergence concepts is to make networks much easier to test and manage, more flexible and more efficient by adding / removing network devices without interfering with other devices. things. Networks defined by Software are one way of using a modelling, scary, and dynamic network (SDN). SDNs are a type of digital technology network that can be configured by the central control unit to meet a variety of needs and objectives. SDNs, OpenFlow networks are the most widely used network structures. In 2008, Stanford University introduced the OpenFlow network protocol as a restart project. In 2011, the Open Networking Foundation (ONF), founded by a group of major communication firms, began sponsoring OpenFlow and SDN, declaring OpenFlow as the first SDN. The data plane and control plane are separated from the basic network devices by SDN. The data plane is the only one that stays on the basic network devices and only makes the data packet forward. The control plane, on the other hand, sits on top of the data plane and oversees the central link action using a single controller defined by the software. A system that contains a data plane and is responsible for data transfer is known as an OpenFlow-enabled switch, and a central control unit or control plane is known as an OpenFlow controller on an OpenFlow network. For testing purposes, various types of control software are available; these software are called Network Operating Systems (NOS) as defined. This paper is organized by defining OpenFlow-approved network formats. Mininet, an open-source platform, was used to introduce OpenFlow network technology. Based on the simulation results, performance comparisons were made for different OpenFlow network topologies. Finally, a conclusion is drawn from research.

II. RELATED WORKS

[1] provides an idea that the right choice of program under the multimedia network is influenced by many factors. It is possible to define a set of network parameters and parameters / systems for programs that contribute to the entire network very efficient, however, at the same time, a particular network topology, network load level and application design should be considered. In addition, total shipping and storage costs play a major role.

[2] provides the Performance analysis of bandwidth, throughput, and other network parameters between nodes of Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) connection.

[3] gives the basic idea of Performance analysis by comparing the entire network topologies based on bandwidth usage, packet delivery rate, travel and return delays between the last nodes and the size of the acquisition. The creation of all the topologies enabled by OpenFlow it is done using a prototype network emulator called Mininet.

[4] gives the study of operation of two SDN controllers - RYU and POX, which are used in Python using Mininet and D-ITG, the Distributed Internet Traffic Generator. During the study two network topologies are used, single and direct. Performance parameters used for high delay, medium jitter, medium bitrate. Test results have shown that a direct topology with a RYU controller works better compared to a single topology with a POX control of different network sizes.

[5] present the main features of a different network simulator and think about its pros and cons. It proves a good source of reference for those people who find it difficult to choose the right network characters in their research.

III. THEORY

A. Limitations of existing networks:

1. Real-world lessons on large production networks are difficult to implement.
2. Suspension research - expensive equipment must be purchased, and networks must be developed for each research team.
3. For many years, the networks have not changed.
4. As the contracts are self-explanatory and there is a lack of high output, the level of innovation in the networks is slow.
5. Closed applications
6. Due to the lack of open spaces, logical interaction is difficult.
7. Vendors are starting to open, but not in a significant way.
8. Only seller / seller partners can invent new things.
9. Communicating with people has many obstacles to new ideas.

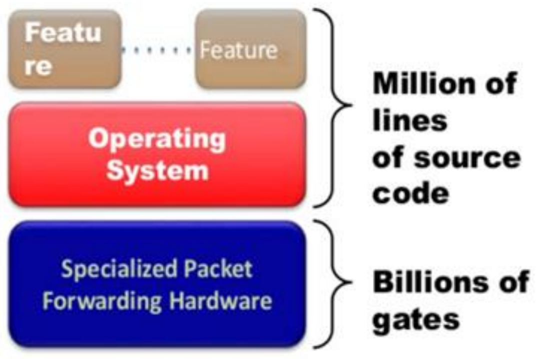

Featu re
Feature
Operating System
Specialized Packet Forwarding Hardware
Million of lines of source code
Billions of gates

Fig 1: Limitations

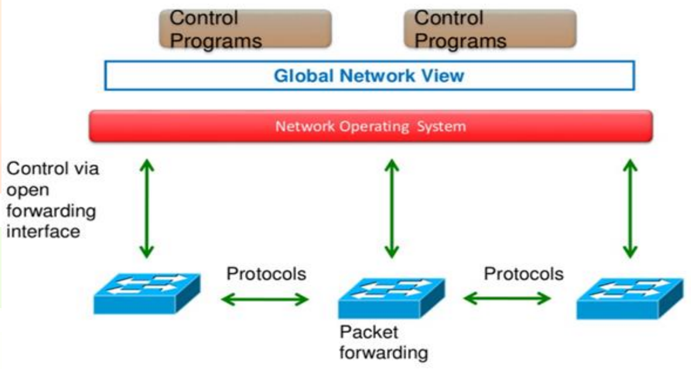


Fig 2: Software Defined Networking

B. Software Defined Networking:

Data plane: packet processing and distribution based on router and destination. For example, IP, TCP, Ethernet, and so on. *Control plane*: is responsible for defining the status of routers and determining how and where packets are sent. Route, traffic engineering, and firewall status. Different aircraft control aircraft and data planes. Store, monitor and organize data flights from a medium-sized enterprise, such as a control flight software known as a controller. A network to control the entire network, not just a computer network. The SDN flow is shown in Fig 2.

C. Need for SDN:

1. Facilitate network creativity.
2. Layered architecture and open interfaces that are all the same.
3. Experiment and testing with lightweight, low-cost equipment.
4. More accessibility due to the ease with which applications can be built by a larger number of vendors.
5. With programmability, you will have more options.
6. Customization and integration with other applications are easy.
7. The terms "programmed a network" and "configure a network" are interchangeable.

D. Architecture of SDN:

As shown in Fig 3 The control and data planes are decoupled in the SDN architecture, as are network intelligence and state, and the underlying infrastructure is abstracted from the applications.

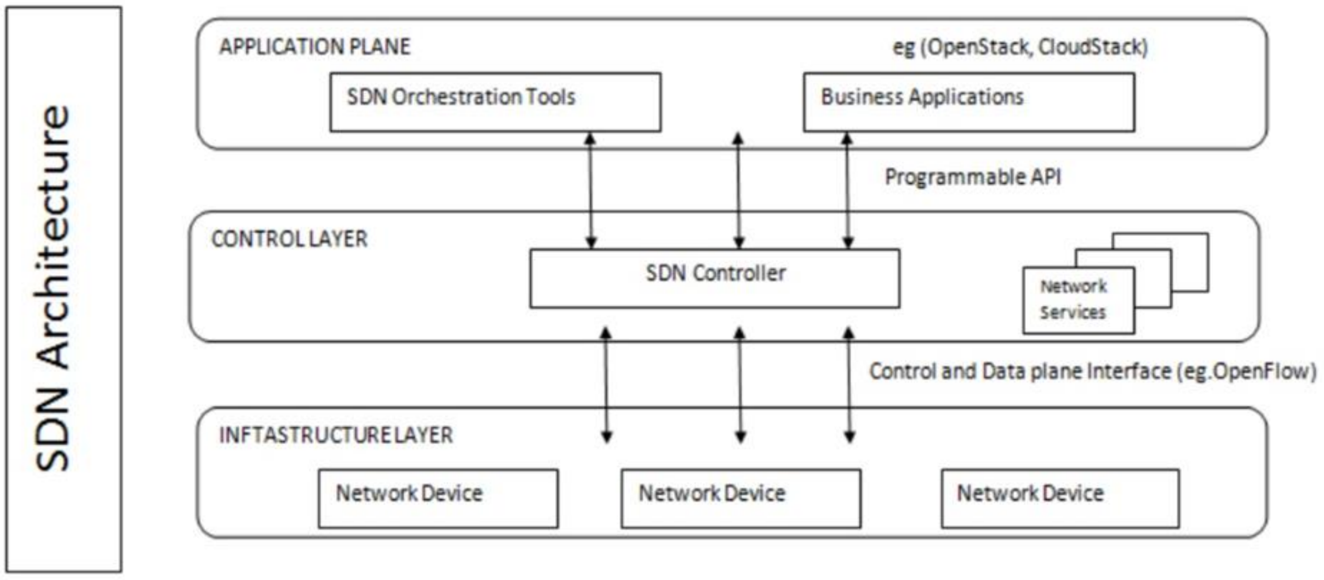


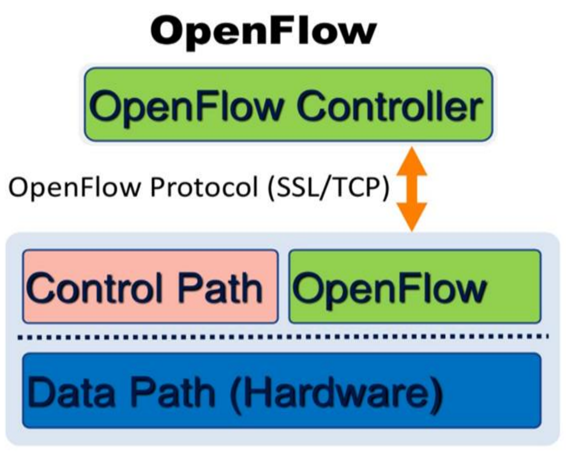
Fig 3: SDN Architecture

E. *SDN Layers:*

1. Infrastructure layer: This layer is made up of both physical and virtual network devices like switches and routers. To enforce traffic forwarding rules, all network devices can use the OpenFlow protocol.
2. Control layer: This layer consists of a decoupled control plane that provides a centralised global view of the entire network. To communicate with the infrastructure layer, the layer can use the OpenFlow protocol. protocol to communicate with below layer i.e infrastructure layer.
3. Application layer: Network resources, applications, and orchestration software make up the application layer, which interacts with the control layer. It provides an open framework for communicating with the architecture's other layers.

F. *OpenFlow Protocol:*

OPENFLOW is an open API for programming data plane switches that offers a standard GUI. It is a protocol for remotely manipulating a switch or router's forwarding table, and it's one of SDN's components. It is implemented on Ethernet switches to allow a controller or control plane in SDN architecture to handle the forwarding plane, or data plane. OpenFlow-based controllers can discover and keep track of all the network's connections, then generate and store all the network's possible routes. The OpenFlow protocol will tell switches and routers how to guide traffic by giving them software-based access to flow tables, which can be used to automatically adjust the network configuration and traffic levels to meet the needs of users.



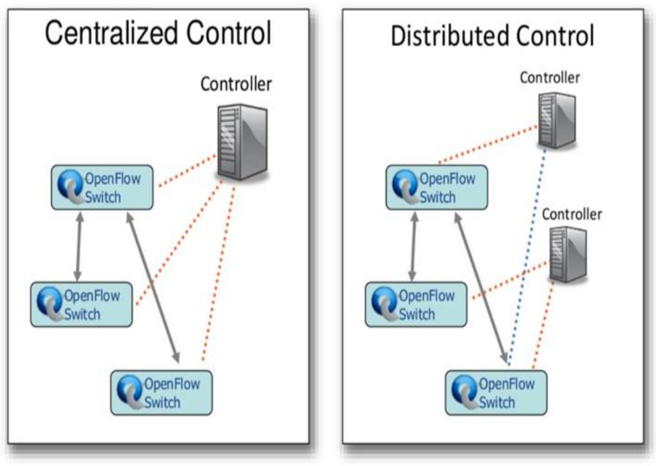


Fig 4: OpenFlow Protocol

Fig 5: Centralized/Distributed Control

G. *OpenFlow Switch and Controller:*

One or more flow tables that enforce packet lookups and forwarding, as well as an OpenFlow channel to link to an external controller, are included in an OpenFlow Switch. The switch communicates with the controller, which uses the OpenFlow protocol to guide the switch. Using the OpenFlow protocol, the controller can remove, add, or update flow entries in existing flow tables in the switch, both reactively (in response to packets) and proactively (ahead of time). The controller takes this decision based on the administrator's rules or the network's circumstances, and the decision it makes is forwarded to all the switches in the network's flow table entries.

IV. UML DIAGRAMS

A. Flow Diagram:

The basic flow of the programme is given by taking the input network model or topology from a user or a client then the network model is created in the Mininet emulator and the traffic is generated using various protocol like OpenFlow. After that the data of various parameters is used to generate an analysis report by using graphs and based on evaluation analysis is done and the efficient solution about the improved network model is provided. The flow diagram is shown in Fig 7.

B. Use Case Diagram:

A use case diagram (as shown in Fig 6) at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. In the network model taken there are two actors a client and the developer. A client provides the Network requirements of its organisation. The diagram is used to tell the actors reaction to various scenarios represented in the form of use cases.

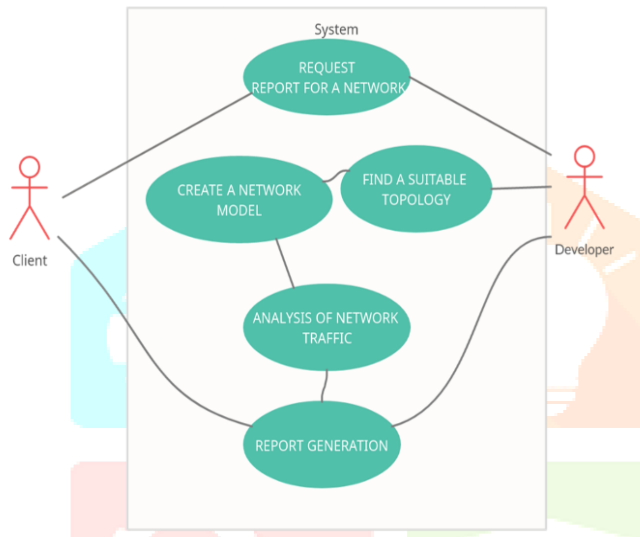


Fig 6: Use Case Diagram

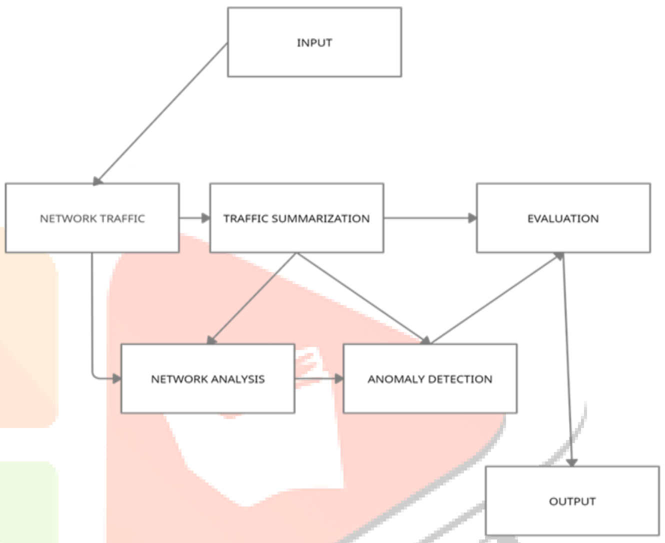


Fig 7: Flow Diagram

C. Sequence Diagram:

A sequence diagram (as shown in Fig 8) shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. In the proposed model the sequence of actions occurring between various classes like consumer client, developer and Mininet. The sequence diagram is used to represent the timely and orderly action that are occurring in relation to various actors interacting with the system. Various actions to be taken and their order of occurrence is represented on the arrows and the direction of their occurrence is as well represented using arrow direction.

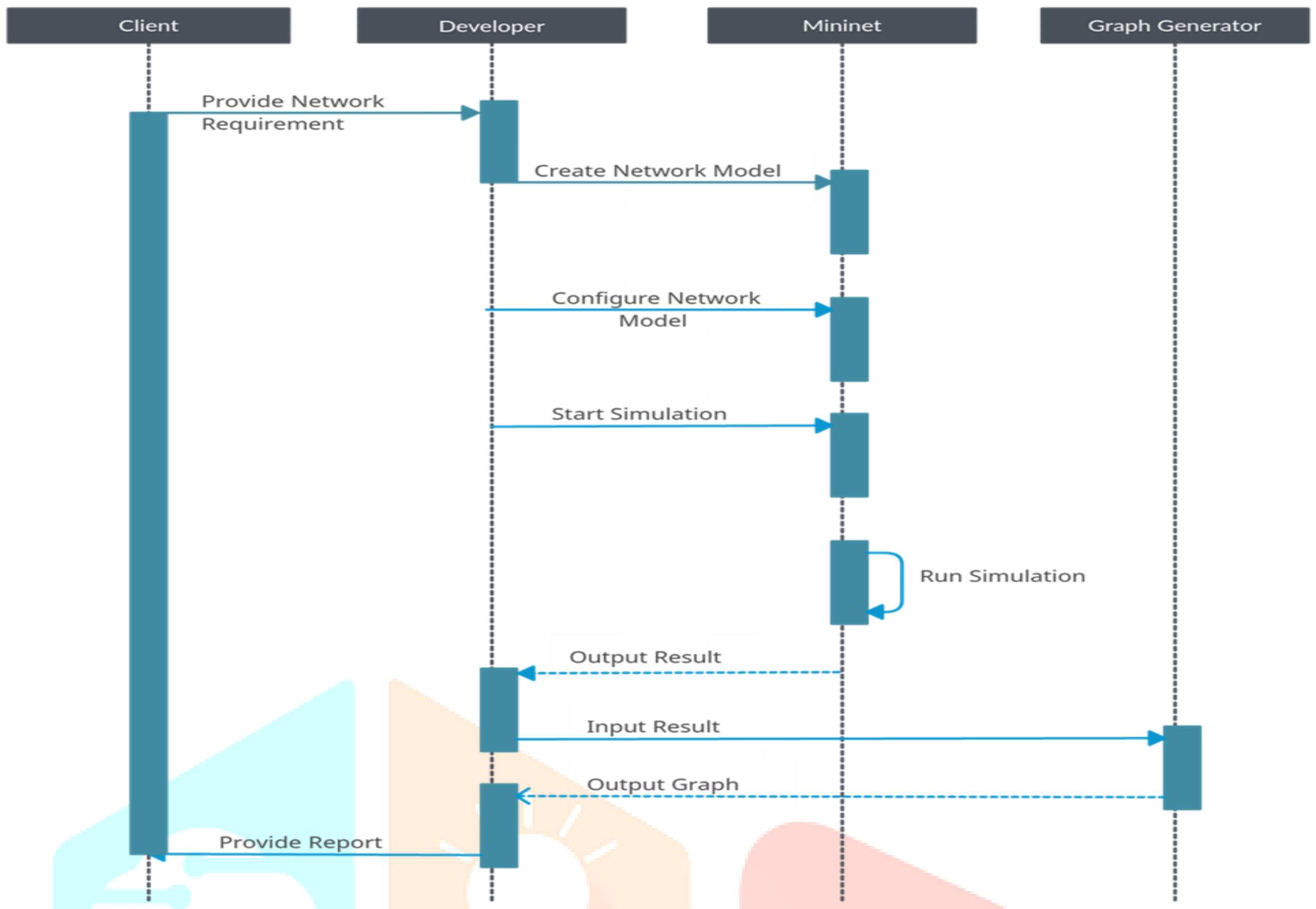


Fig 8: Sequence Diagram

V. METHODOLGY

A. Selection of Tool:

A network simulator is software that predicts how a computer network behaves. Network simulators are used because communication networks have become more complex in the traditional design methods to provide an accurate description of system behaviour. Characters with processors, links, applications, etc. are used to model a computer network, and network output is registered. Network / configuration model defines the network (nodes, routers, switches, and connections) and the operations in which it is performed (data transfer, packet error, etc.). Network-level metrics, connection metrics, app metrics, and other metrics will be included in the output function. Tracking files are used to record and analyse all packages and events that occurred during the simulation. The most common way to send real packets from a live app to a simulation server (where the virtual network is simulated). The simulation package is made by 'simulating' the actual package. After experiencing losses, errors, delays, jitter, and other network effects, the simulation packet is downgraded to the actual packet, transferring these network effects to the actual package. As a result, it appears that the actual package travels through a real network while, in fact, it travels through a created network. Network simulators are available in both free / open source and related facilities.

The most well-known network simulators/emulators are:

1. NS2 / NS3
2. OPNET
3. NetSim
4. QUALNET
5. MININET

Mininet is a virtual test site that allows for the development and testing of network tools and agreements. With a single command, Mininet can build a virtual network for any type of machine (Virtual Machine (VM), cloud-based, or native). Therefore, it provides an affordable solution and systematic development that works in conjunction with production networks1. Mininet offers the following features:

Mininet includes many excellent features for emulators, hardware testbeds, and simulators.

Compared to the methods based on the presentation of the complete system, Mininet:

1. Quick boots: seconds instead of minutes
2. Larger scale: hundreds of hosts and switches compared to single numbers.
3. Provides maximum bandwidth: usually 2Gbps total bandwidth on humble hardware.
4. Easy installation: Pre-installed VM works using VMware or VirtualBox for Mac / Win / Linux with OpenFlow v1.0 tools already installed.

Compared to hardware test boards, Mininet:

1. inexpensive and always available (even before meeting times)
2. It can rearrange quickly and can restart.

Compared to the characters, Mininet:

1. uses real, random code including application code, OS kernel code, and flight control code (both OpenFlow control code and Open vSwitch code)
2. connects easily to real networks.
3. Provides interactive function - you can type it.

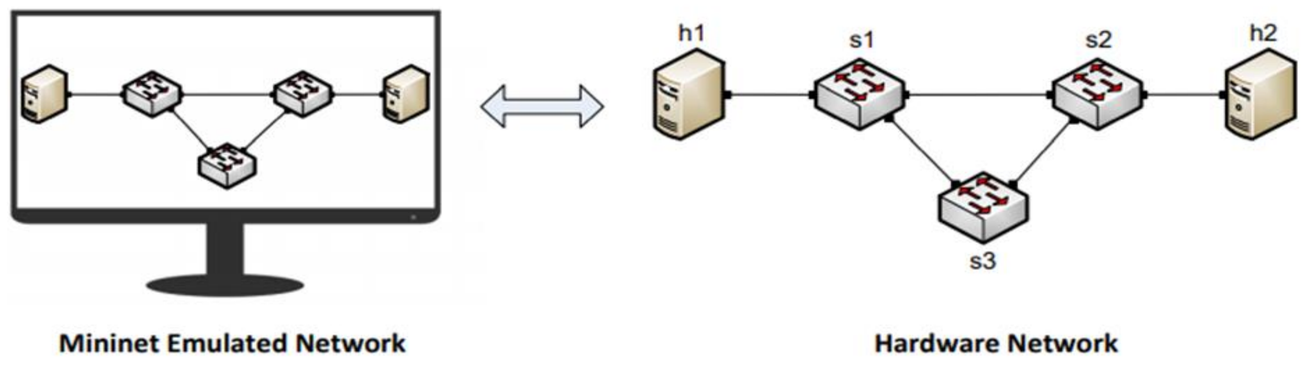


Fig 9: Hardware Network vs Mininet Emulated Network

B. *Creatin of Network in Emulator:*

Mininet provides a comprehensive Python API for network design and testing. We will now be building a drug network for our simulation.

Step 1: Launch the Mininet Command Line (CLI) interface using Linux Terminal.

Step 2: Build a tree network with 3 depths and a fanout 4 value, so there will be 1 controller in our network, 21 switches in the network and 64 hosts in the network.

Command: Sudo mn –topo tree, depth = 3, fanout = 4

Step 3: Testing the connection

Mininet's default topology provides IP addresses 10.0.0.1/64 and 10.0.0.2 / 64 h1 hosting and h2 hosting, respectively. To check the connection between them, you can use the ping command. The ping command works by sending the Internet Control Message Protocol (ICMP) Echo Application to a remote computer and waiting for a response. Available information includes how many responses are returned and how long it takes to get them back.

Command: h1 ping 10.0.0.2

Step 4: Now with the help of NETEM we will add delays, jitter, packet loss, packet corruption, packet redistribution, packet duplication to external network so that we can compare it to the real network.

User calls NETEM using a command line service called tc. Without additional restrictions, NETEM behaves like a basic FIFO line without delay, loss, duplication, or package rearrangement. The basic tc syntax used with NETEM is as follows:

Syntax: Sudo tc qdisc [add | del | replace | switch | show] dev dev_id impande netem opts

a) Sudo: allow the execution of orders with higher security rights.
b) tc: command used to communicate with NETEM.
c) qdisc: line code (qdisc) is a set of rules that determine the order in which packets arrive from IP protocol output are provided. A queue line is used in the packet line to determine when to send each packet.
d) [add | del | replace | switch | show]: this is working on qdisc. For example, to add a delay to a specific interface, the function will be added. To change or remove the delay in a specific interface, the function will be a change or del.
e) dev_id: this parameter indicates the interface that should be displayed for simulation.
f) opens: this parameter indicates the number of delays, packet loss, duplication, corruption, and more.

Command: Sudo tc qdisc add dev <link> root netem delay 100ms 10ms 25% loss 10% corrupt 1% reorder 25% 50% duplicate 20%

The above command when issued will add 100ms delay with 10ms jitter, 10% packets will be lost on the network, 1% packets will be damaged on the network, 25% of the packets will be shipped as well as rest 75% of the packets will be reconfigured online 20% of the packets will be doubled for that particular <link> person.

Step 5: We will now set the link bandwidth with the help of Token Bucket Filter (TBF)

Syntax:

sudo tc qdisc [add | ...] dev [dev_id] root tbf limit [BYTES] burst [BYTES] rate [BPS] [man BYTES] [peakrate BPS] [latency TIME]

a) tc: Linux traffic control tool.
b) qdisc: line code (qdisc) is a set of rules that determine the order in which packets arrive from IP protocol output are provided. A queue line is used in the packet line to determine when to send each packet.
c) [add | del | replace | switch | show]: this is working on qdisc. For example, installing a token bucket algorithm on a specific interface connector, will add functionality. To change it or remove it, the function will be a change or del, respectively.
d) dev [dev_id]: this parameter indicates that the interface must be under simulation.
e) tbf: this parameter specifies the bucket and token filter algorithm.
f) limit [BYTES]: packet line size in bytes.
g) explode [BYTES]: the number of bytes that can enter the bucket.
h) rate [BPS]: transfer speed, determined by the frequency at which tokens are placed in the bucket.
i) man [BYTES]: the maximum unit of transmission in bytes.
j) Peakrate [BPS]: high explosion speed.
k) Delay [TIME]: the maximum time a package can wait in line.

Command:

Sudo tc qdisc add dev <link> root tbf rate 10gbit limit 5000000 explode 15000000.

The command above when used will set the bandwidth to 10Gbps for that link.

Step 6: Now after setting up the link we will make the data transfer between the various hosts.

Here we will host host h1 as Server and host h64 as Client. In the last version of Server, the next command to start iperf3 in server mode.

Command in h1 terminal: perf3 -s

When this command is executed, the server will start in host h1.

By typing the client to keep the next command to launch iperf3 in client mode

Command on h64 terminal: perf3 -c 10.0.0.1 -t 20 -J> test_results.json

When executing the above command, the client will start on Host h64 and will request data from 10.0.0.1 i.e., host h1 (server). It will keep receiving data for the next 20s and all logs will be stored in the test_results.json file in Json format.

Step 7: We will now edit the json data in a graphical format for better understanding.

For this we will be using the usable plot_iperf.sh usable file that already exists on our machine.

VI. ALGORITHMS

The algorithm used is the Token Bucket Algorithm. When emulating Wide Area Network (WAN), it is sometimes necessary to limit the bandwidth of devices (end hosts and communication devices) to detect network performance in a variety of situations. Bucket Token is an algorithm used for packet switching networks to limit bandwidth and traffic explosion. In summary, a bucket of tokens consists of inserting tokens (represented as packets or bytes of packets) at a fixed rate in a fixed volume bucket. When a new package arrives, the bucket is checked to check the number of tokens available; if at least n tokens are available, n tokens are removed from the bucket, and the packet is sent to the network. In some cases, no tokens are removed, and the package is considered incompatible. In that case, the package may be discarded, lined, or transferred but marked as inconsistent.

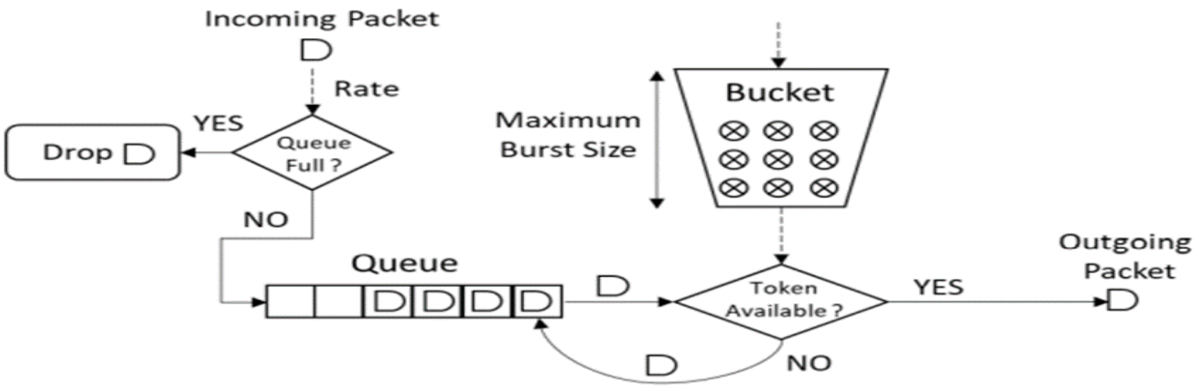


Fig 10: Token Bucket Filter

The rate, which is the transfer speed, is determined by the frequency at which the tokens are placed in the bucket. Another important asset of the algorithm for bucket tokens is explosion; when the bucket stays full (meaning there are no packets eating the tokens), the new packets will consume the tokens immediately, without limitation. Explosions are defined as the number of tokens that can fit in a bucket, or in the size of a bucket. Providing limits and explosive controls, the installation of buckets of tokens often creates another small bucket equal to the Maximum Transmission Unit (MTU), and a much faster rate than the original bucket (maximum value). Its level defines the maximum explosion speed. The token bucket algorithm installed on Linux is the Token Bucket Filter (tbf), which is a linear tool used in conjunction with Linux Traffic Control (tc) to create traffic. The image below shows the main parameters used by tbf.

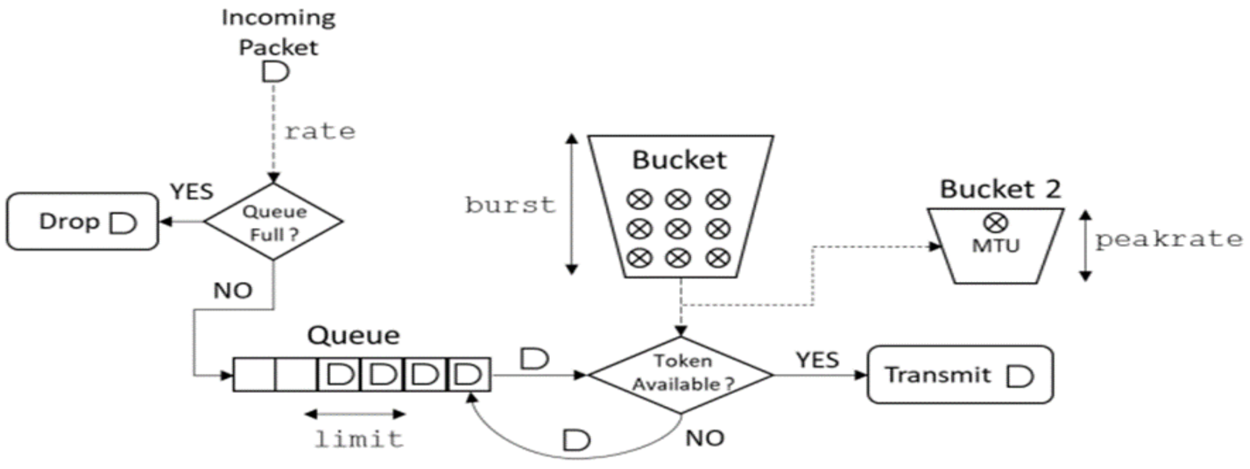
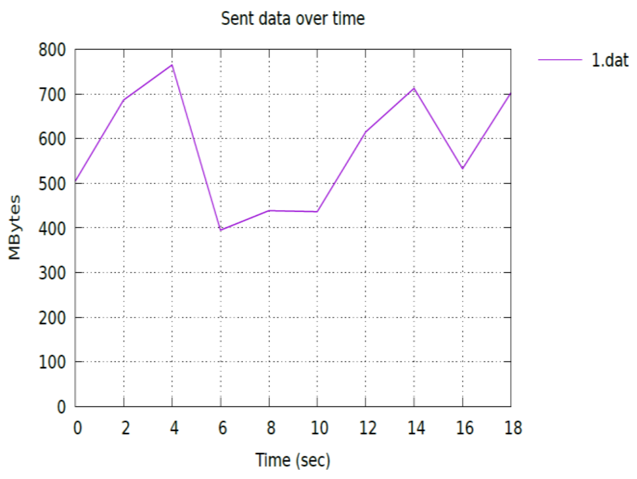
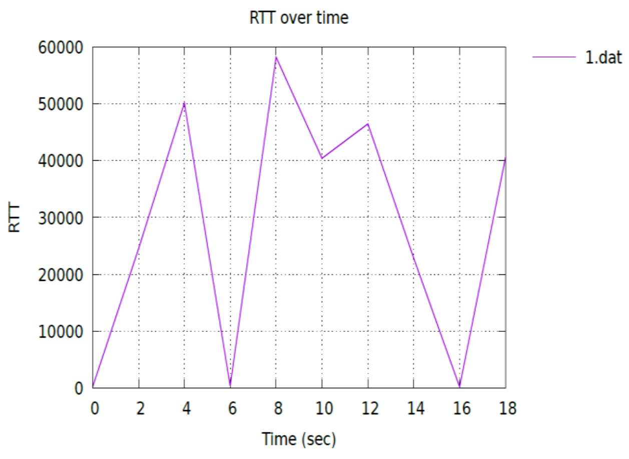


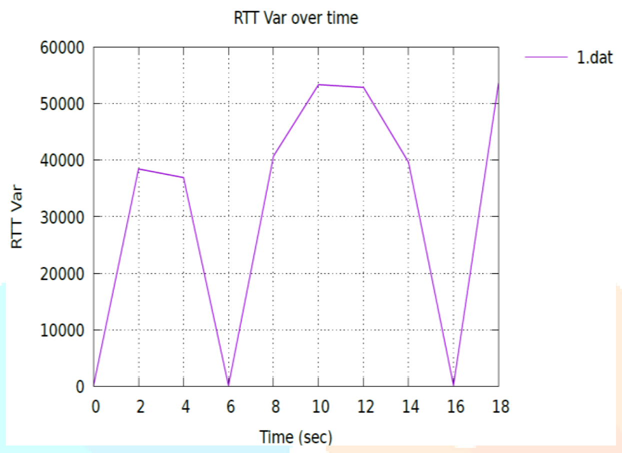
Fig 11: tbf parameters and architecture

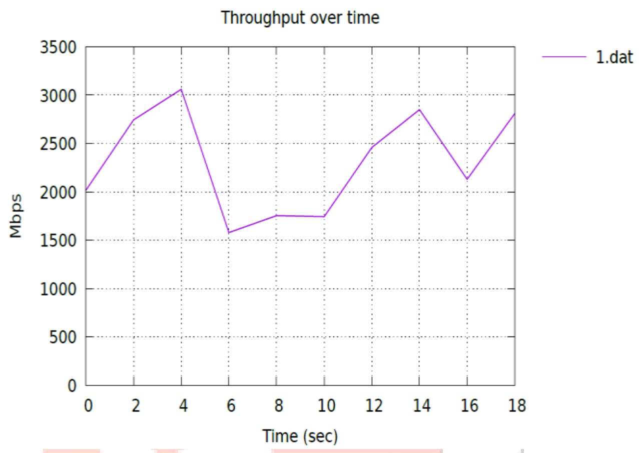
VII. RESULTS

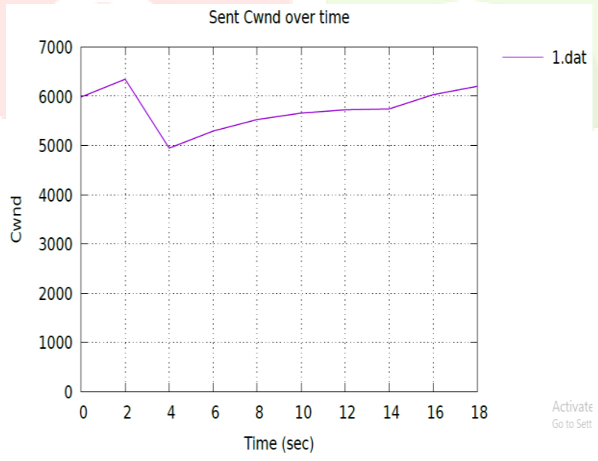
After implementing the above methodologies, algorithm and creating a network model in the Mininet emulator, we get the different set of values for various parameters like traffic delay, throughput etc. These values can be taken as dataset and using GUI graphs for various parameters can be generated. The output graphs of a sample network created in Mininet is shown below.

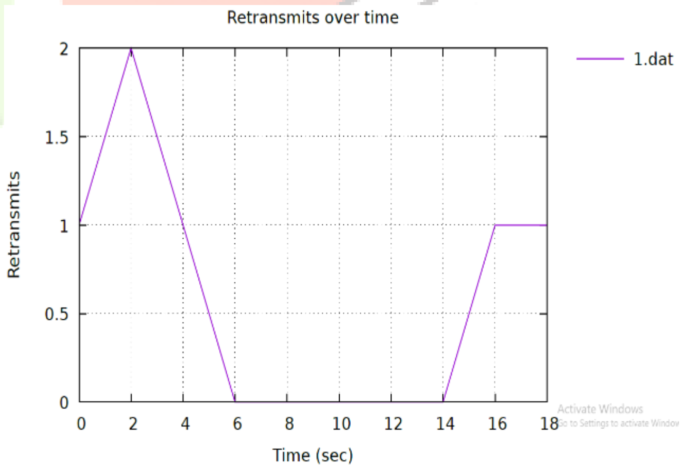

Sent data over time
MBytes
Time (sec)
1.dat


RTT over time
RTT
Time (sec)
1.dat


RTT Var over time
RTT Var
Time (sec)
1.dat


Throughput over time
Mbps
Time (sec)
1.dat


Sent Cwnd over time
Cwnd
Time (sec)
1.dat


Retransmits over time
Retransmits
Time (sec)
1.dat

VIII. CONCLUSION

In similar manner we can obtain and compare results for various networks and choose the efficient one. Software Defined Networking using OpenFlow technology can be very effective network building for future Internet technology and Future Generation Networks (NGN). A lot of research is being done in this area by network engineers to improve flexibility and a network plan to produce different management and implementation strategies a safer network. Depending on the nature of the future network concerns, the deployment of The OpenFlow network in a small office or campus area can be better built than ever The Internet is not going to be fast, scary, reliable, secure, and so on advanced features. In this paper the OpenFlow network design is proposed for the various courses available done. The proposed network construction is implemented using a specific type of software emulator: Mininet. Mininet supports OpenFlow agreements and is based on Python writing. The performance of the proposed model has been assessed via network connectivity a test between hosts through network management agreements (ICMP). A ping test is performed between two servers in a network connected by a separate OpenFlow switch. Based on the results obtained concludes that all control work is done by flight control, OpenFlow controller, and switch only perform transfer function. First The echo application package has taken a lot of time to address the address with the controller and simultaneously time input is included in the flow tables, with the specified closing time, approx. control to send the following packets without resolving addresses. Ping test it is also done between available and offline networks, and functionality different ceremonies in the network depending on the results in making the command has updated. Network penetration testing is also performed, depending on the results the findings concluded that the pass can be enhanced by increasing the size of the TCP window for TCP connectivity and for reducing network jitter, packet loss, and out of order packet arrival of UDP connection. Based on the above tests we concluded that the OpenFlow network is centralized performance is like a traditional network with a data separation difference flight from control plane. This separation reduces the network load on a single transmission hardware and distributes the full load, ultimately improving the efficiency of the network with reducing network overload generated from relay attempts due to packet drops. This technology can be used for a variety of complex network situations including interface between wireless and wireless structures.

REFERENCES

[1] Pavel Masek, Dominik Kovac, Jiri Hosek, Mariya Pavlova and Ondrej Krajsa: “Analysis of Network Parameters Influencing Performance of Hybrid Multimedia Networks”. International Journal of Advances in Telecommunications Electrotechnics Signals and Systems: December 2013. DOI: 10.11601/ijates.v2i3.69

[2] Idris Zoher Bholebawa, Rakesh Kumar Jha and Upena D. Dalal: “Performance Analysis of Proposed OpenFlow-Based Network Architecture Using Mininet”. Wireless Pers Commun DOI 10.1007/s11277-015-2963-4

[3] Idris Zoher Bholebawa and Upena D. Dalal: “Design and Performance Analysis of OpenFlow-Enabled Network Topologies Using Mininet”. International Journal of Computer and Communication Engineering. Volume 5, Number 6, November 2016

[4] Pramod B Patil., Kanchan S. Bhagat., D K Kirange and S. D. Patil: “Software Defined Networks using Mininet”. International Journal of Recent Technology and Engineering (IJRTE). ISSN: 2277-3878, Volume-9 Issue-1, May 2020

[5] Arvind T: “A Comparative Study of Various Network Simulation Tools”. International Journal of Computer Science & Engineering Technology (IJCSET). ISSN: 2229-3345. Vol. 7 No. 08 Aug 2016