iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 8 **Issue:** VI **Month of publication:** June 2020

DOI: http://doi.org/10.22214/ijraset.2020.6182

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue VI June 2020- Available at www.ijraset.com

Implementation of Firewall Functions on Software Defined Network using Floodlight Controller

Yamini Chavan[1], Purvi Shesware[2]
[1, 2]*Department Of Information Technology, Bharati Vidyapeeth College of Engineering,*

Abstract: ***Numerous people have acknowledged the need to re-engineer the contemporary internet industry effort into a much more high-octane central network which is facing the challenge of sustaining firewalls up to date. It is difficult for today's inflexible infrastructure to cope with the fast-changing necessity of the users. Hence, Software-Defined Network (SDN) was introduced in the mid-2000s to transform today's network to have centralized rapid transformation, centralized management, and programmability by separating the data plane and the control plane. Here we focus on initiating firewall functions that run on an SDN controller based over an OpenFlow protocol, to show that most of the firewall functionalities can even be built easily via software, without the help of a resolute hardware. Among many OpenFlow controllers that already exist for the public, we have chosen Floodlight written in Python for the project and to create the SDN network topology, we have used VirtualBox and Mininet. In our study, we incorporate the implementation details of firewall functions, in addition to the experimentation outcome.***

Keywords: layer 2 Based firewall, Mininet emulation tool, OpenFlow protocol, Software Defined Networking (SDN), FLOODLIGH, Java.

I. INTRODUCTION

Software Defined Network is a developing area of the research in the sector of networking. Firewalls are one of the most important components used in networks, and new challenges have been driven by the software-defined networks in implementing firewalls. The main issue of the firewall is its speed. The speed of the firewall is a hindrance; often firewall link speeds are slower than the supported network interface and can cause the traffic burst from the host to be buffered until packets are processed. To overcome these issues, the aim of our project is to solve upcoming speed related problems by implementing duplicate instances of the firewall. By designing two topologies, a single and multiple controllers, and implementing them in a simulated environment multiple controllers in a network environment tend to show an improved performance With the increased number of attacks, firewalls are also becoming slower and more vulnerable to lags and firewall explosions. With the evolution of Software-Defined Networking (SDN), it has become difficult to implement network components including firewalls in traditional networks. That is why an upgrade is necessary to adapt to new changes. Software firewall components configured from a single controller is very beneficial, making it easy to set rules around the network.

II. PROGRAMMING REQUIREMENTS

A. SDN Technology

Software-defined networking (SDN) technology is an approach to network management that enables dynamic, programmatically efficient network configuration in order to improve network performance and monitoring making it more like cloud computing than traditional network management. SDN is meant to address the fact that the static architecture of traditional networks is decentralized and complex while current networks require more flexibility and easy troubleshooting. SDN attempts to centralize network intelligence in one network component by disassociating the forwarding process of network packets (data plane) from the routing process (control plane). The control plane consists of one or more controllers which are considered as the brain of the SDN network where the whole intelligence is incorporated.

1) Features of SDN

a) *Programmatically Configured:* SDN lets network managers configure, manage, secure, and optimize network resources very quickly via dynamic, automated SDN programs, which they can write themselves because the programs do not depend on proprietary software.

b) *Directly Programmable:* Network control is directly programmable because it is decoupled from forwarding functions.

c) *Agile:* Abstracting control from forwarding lets administrators dynamically adjust network-wide traffic flow to meet changing needs.

d) *Centrally Managed:* Network intelligence is (logically) centralized in software-based SDN controllers that maintain a global view of the network, which appears to applications and policy engines as a single, logical switch.

e) *Open standards-based and vendor-neutral:* When implemented through open standards, SDN simplifies network design and operation because instructions are provided by SDN controllers instead of multiple, vendor-specific devices and protocols.

B. Open-Flow Protocol

Open-Flow enables network controllers to determine the path of network packets across a network of switches. The controllers are distinct from the switches. This separation of the control from the forwarding allows for more sophisticated traffic management than is feasible using access control lists (ACLs) and routing protocols. Also, OpenFlow allows switches from different vendors often each with their own proprietary interfaces and scripting languages to be managed remotely using a single, open protocol. The protocol's inventors consider OpenFlow an enabler of software-defined networking (SDN).

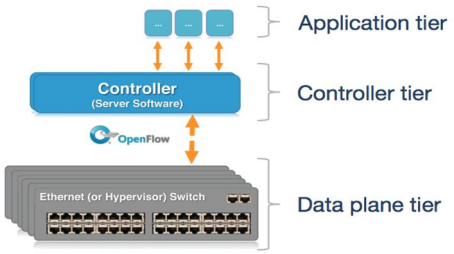


Fig. 1. SDN Architecture

1) *Features of OpenFlow*

a) Protocols other than OpenFlow are:

b) Border Gateway Protocol(BGP)

c) Netconf

d) Extensible Messaging and Presence Protocol(XMPP)

e) Open vSwitch Database Management Protocol(OVSDB)

f) MPLS Transport Profile(MPLS-TP)

2) *Abbreviations and Acronyms Advantages of Open Flow over above mentioned protocols*

a) Open-Flow based SDN creates flexibility in how the network is used, operated, and sold. The software that governs it can be written by enterprises and service providers using ordinary software environments.

b) Open-Flow based SDN enables virtualization of the network, and therefore the integration of the network with computing and storage. This allows the entire IT operation to be governed more sleekly with a single viewpoint and toolset.

C. FLOODLIGHT Controller

Floodlight Controller is an SDN Controller developed by an open community of developers, many of which from Big Switch Networks, that uses with the OpenFlow protocol to orchestrate traffic flows in a Software-Defined Networking (SDN) environment.Floodlight is Java-based and intended to run with standard JDK tools and ant and can optionally be run in Eclipse.

1) Features of FLOODLIGHT Controllers available other than are:

a) Rich Set of Modules

b) The Ability to Easily Adapt Software

c) Develop Applications written in Java

2) Advantages of FLOODLIGHT over above mentioned controllers are:

a) “Pythonic” OpenFlow interface.

b) Reusable sample components for path selection, topology discovery, etc.

c) “Runs anywhere” – Can bundle with install-free PyPy runtime for easy deployment.

d) Specifically targets Linux, Mac OS, and Windows.

e) Topology discovery.

f) Supports the same GUI and visualization tools as NOX.

g) Performs well compared to NOX applications written in Python

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue VI June 2020- Available at www.ijraset.com

III.METHODOLOGY

A network topology is created using the Mininet tool which is a python based interface which enables users to create virtual network topologies which will be further virtualized.

Network consists of 6 hosts, 2 switches and 1 controller. This controller will be a remote controller where IP address and the port number of remote controller will be specified and on another terminal we will be running FLOODLIGHT controller on the same IP address and port.

Out of the all OpenFlow enabled switches, we will consider 1 switch on which the Firewall application will be deployed and act as a specified permission for the whole network. DPID to that switch will be given so that we can refer that switch while we run the controller.

Out of the 6 hosts, 2 of them will act as HTTP Servers which will serve on port 80 and the rest hosts will be the users which will try to send requests to these 2 servers.

Using the Curl command-line tool for transferring data using various network protocols. The name stands for "Client URL". We add ACL rules into the ACL REST INTERFACE of Firewall Actions. While the Floodlight Controller is initialized and working

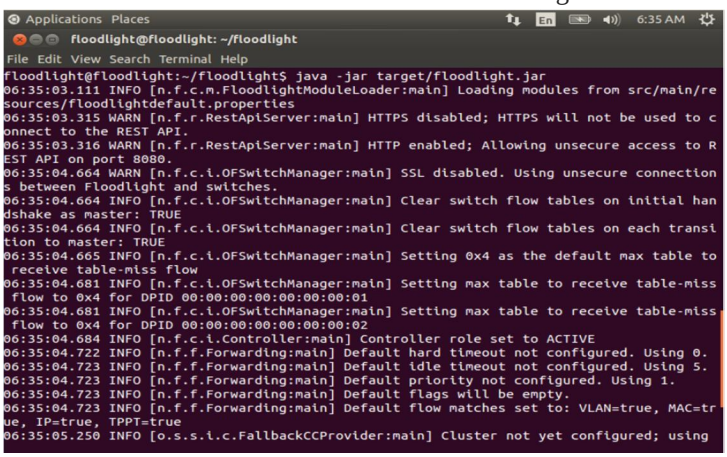


Fig 2. Initializing the Floodlight

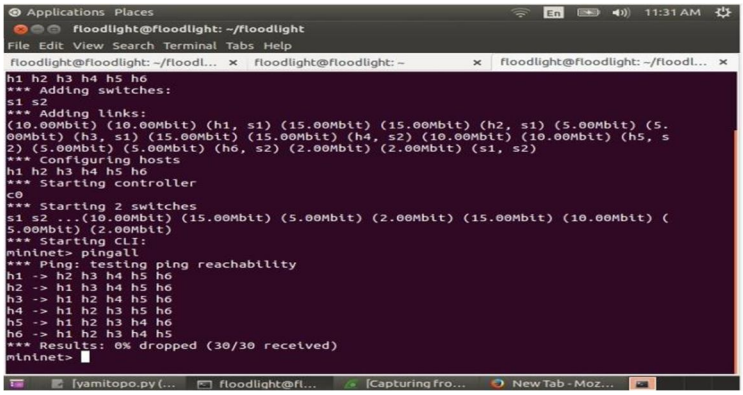


Fig 3. Adding Controller, Switches, Hosts and Testing Connectivity

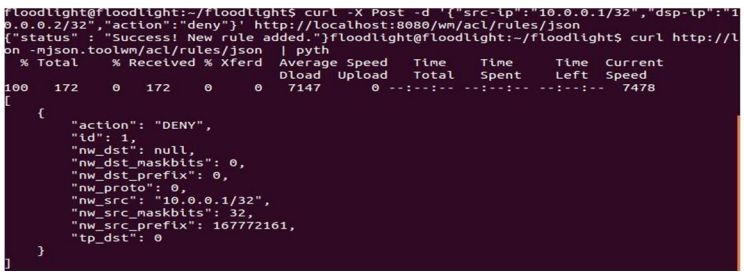


Fig 4. Applying firewall rules

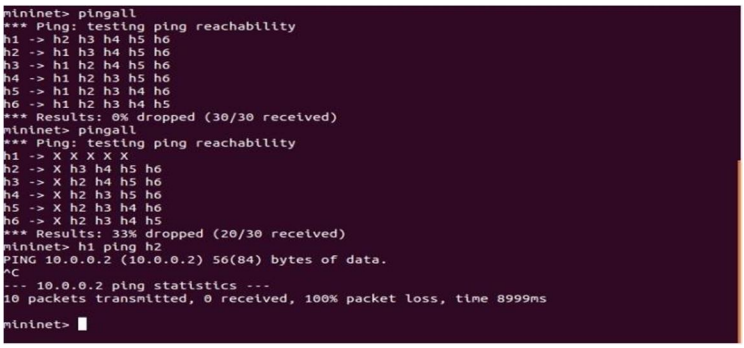


Fig 5. Testing connectivity after Applying firewall rules

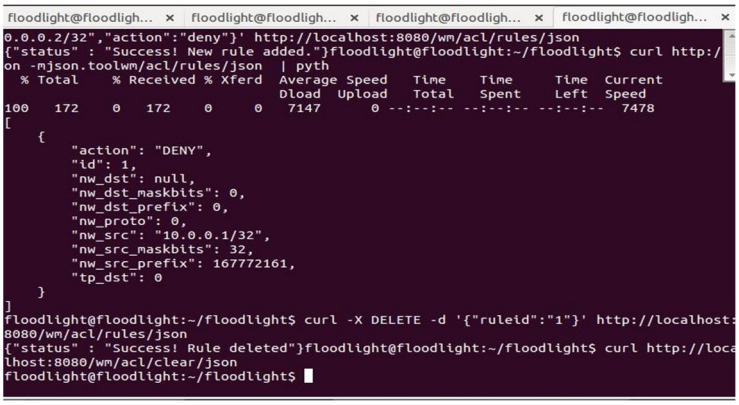


Fig 6. Removing previously made rules of firewall

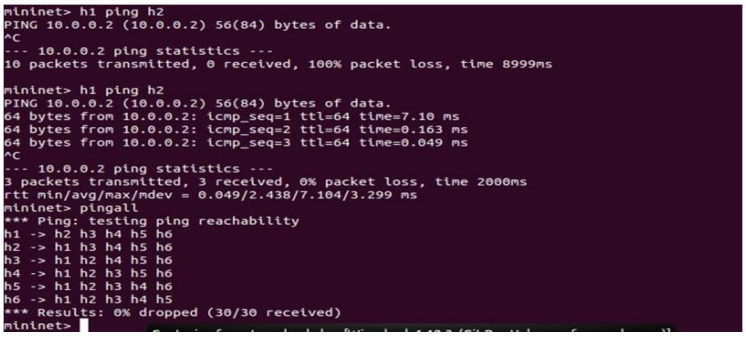


Fig 7. Packet transfer after application of rule removal on firewall

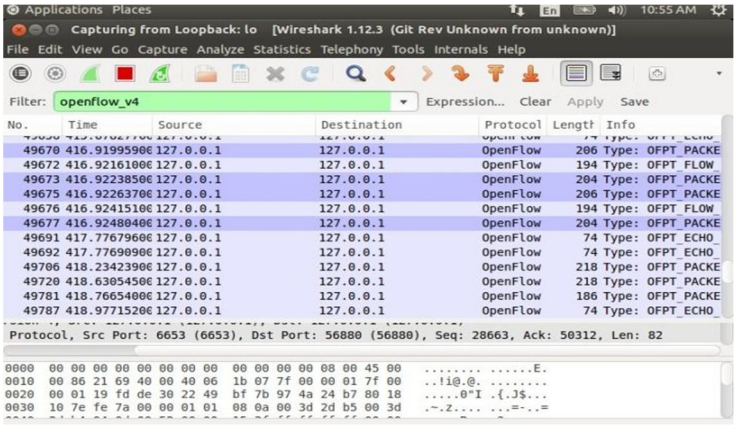


Fig 8. Representing source & destination of packet by using Openflow protocol (h1 ping h2 Highlighted)

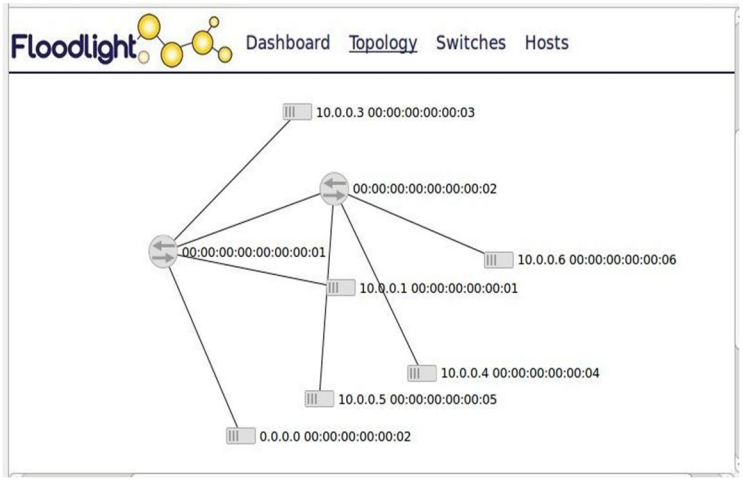


Fig 9. Topology Created

Fig 10. Details of packet transferred via switches

Fig 11. After Firewall Rules were Added

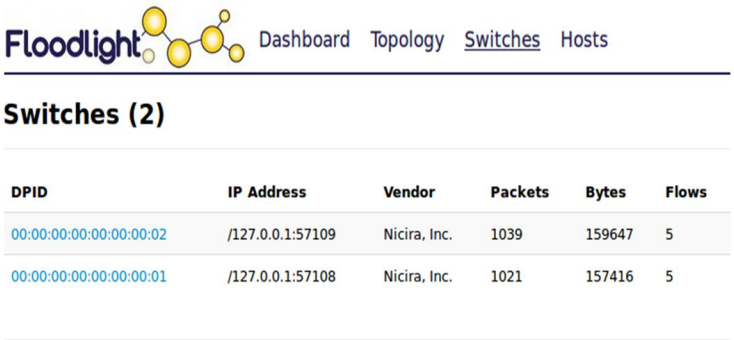


Fig 12. FigAfter Firewall Rules were Removed

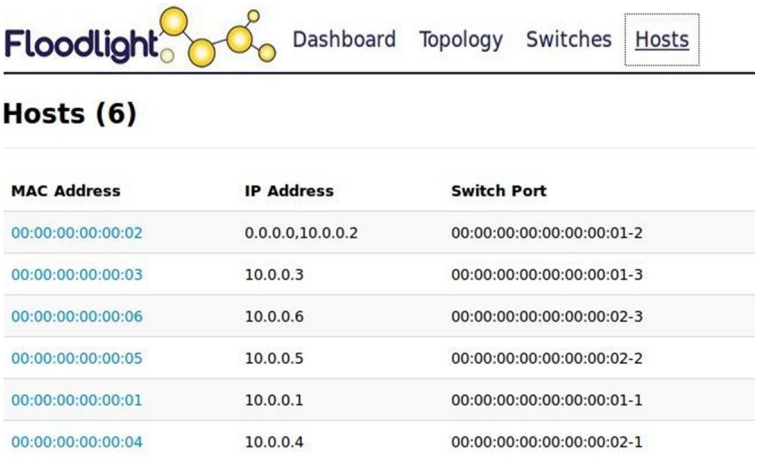


Fig 13. Switch - Host Connectivity

IV. CONCLUSIONS

The scalability features of the Floodlight controller by implementing a scenario in a simulation experimental environment. In this paper, authors have provided a clear idea on how to create experimental tests with analysis of obtained statistical results keeping the performance as the central focus. We would conclude this paper by presenting an option of the implementation of SDN in simulation and emulation environments to inspire researchers to ideate and practically implement their objective aims by means of simulations via various applications to discover and produce with contributions pushing towards the technology ahead.

V. ACKNOWLEDGMENT

We express esteemed gratitude and sincere thanks to our worthy project supervisor Prof. S. H. Darekar Sir, our vocabulary is yet to find suitable words benefiting from the high standard of knowledge set by him and extreme sincerity and affection with which he has regularly encouraged us to put our hearts and souls in this work. We are much obliged to our honourable Head of Department Prof H. B. Sale Sir whose support and cooperation was always helpful and encouraging. We also convey great thanks to our Honourable Principal Sandhya Jadhav Ma'am. Our parents who always bear with us in every critical situation and provide the support whenever required. As we give expression to our love and appreciation our heart is filled. In addition, we sincerely appreciate your valuable help. Please accept our respect and gratitude.

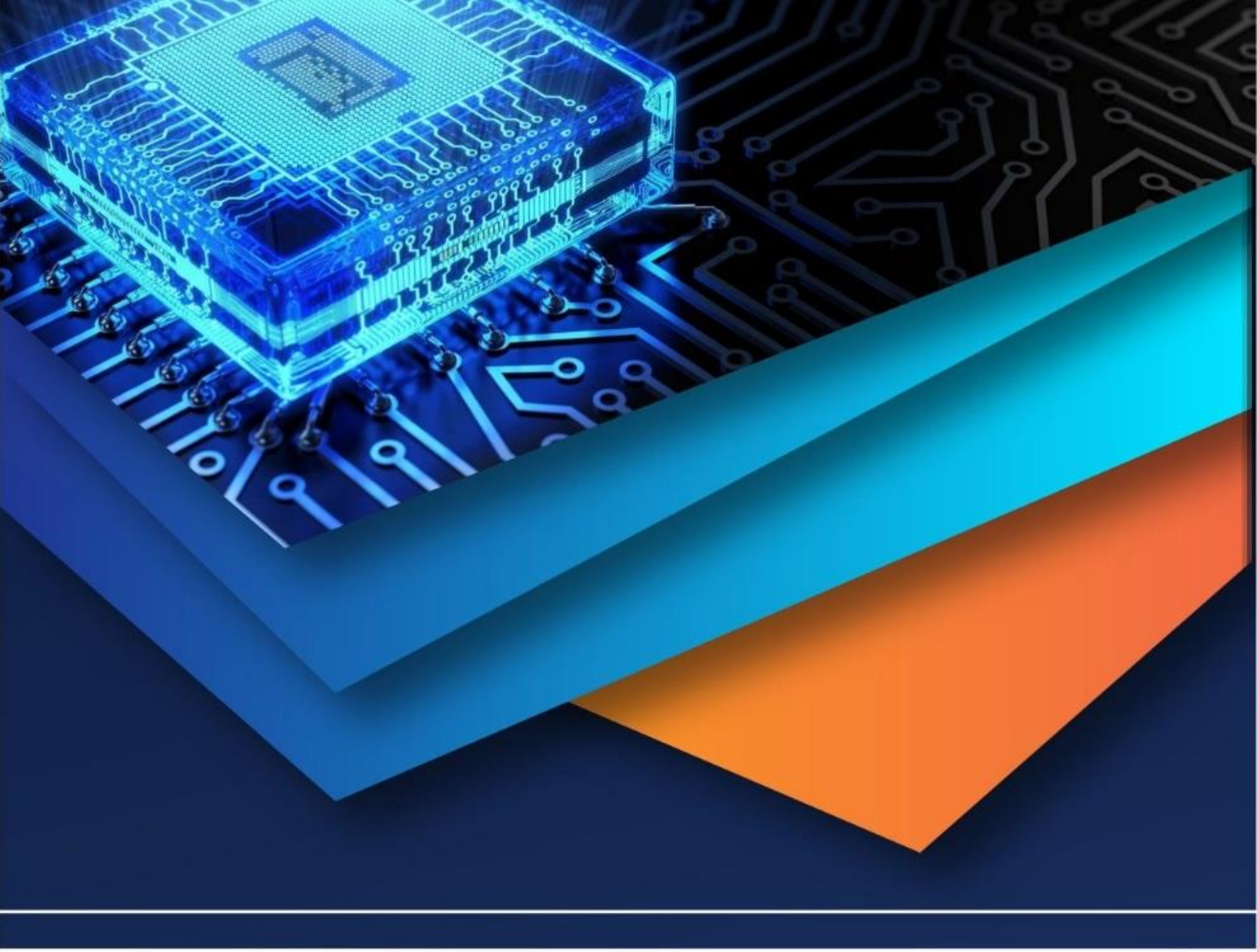
REFERENCES

[1] Yaning Zhou, Ying Wang, Jinke Yu, Junhua Ba, Shilei Zhang "Load Balancing for Multiple Controllers in SDN Based on Switches Group," State Key Laboratory of Networking and Switching Technology Beijing,

[2] Hailong Zhang, Xiao Guo, "SDN-BASED LOAD BALANCING STRATEGY FOR SERVER CLUSTER" School of Information Engineering, Communication University of China, Beijing 100024, China.

[3] MaoQilin,Shen WeiKang, "A Load Balancing Method Based on SDN,"

[4] Wenjing Lan1, Fangmin Li, Xinhua Liu, Yiwen Qiu, "A Dynamic Load Balancing Mechanism for Distributed Controllers in Software-Defined Networking," School of Information Engineering, Wuhan University of Technology, Wuhan 430070, China.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY