Traffic Flow Control Algorithm in Software Defined Networking

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Technology

in

Computer Science and Application

Submitted By

Jatin Sharma

(Roll No. 601634009)

Under the supervision of:

Dr. Rajesh Kumar

Professor

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY

PATIALA – 147004

June 2018

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Traffic Flow Control Algorithm in Software Defined Networking"**, in partial fulfillment of the requirements for the award of degree of Master of Technology in *Computer Science and Application* submitted in **Computer Science and Engineering** Department of Thapar Institute of Engineering and Technology, Patiala, is an authentic record of my own work carried out under the supervision of **Dr. Rajesh Kumar** and refers other researchers' work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Jatin Sharma

Jatin Sharma

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

12/07/2018

Dr. Rajesh Kumar

Professor, CSED

Acknowledgement

First and foremost, I would like to express my sincere gratitude to my supervisor **Dr. Rajesh Kumar**, Professor Computer Science and Engineering Department, for his immense help, guidance, stimulating suggestions and encouragement all the time with this thesis work. He always provide me motivating and enthusiastic atmosphere to work with, it was a great pleasure to do this thesis under his supervision.

I would also like to express my appreciation to my colleagues at Thapar Institute of Engineering and Technology, Patiala.

Finally my special gratitude to my family for never ending love and support.

Jatin Sharma

601634009

Abstract

Software Defined Networking (SDN) is an architecture which used to split up the control plane and data planes in routers and switches to make networks programmable and scalable. In SDN, the control can be decoupled from hardware and applied in software. As a total effect, it is easier to accordingly modify the network. Traditional network cannot guarantee the data transfer and the traffic flow from the network. Traffic engineering is an important technique and network application to optimize a network and improve its robustness. Due to increase in traffic demand, traffic engineering has the ability to minimize the degradation of service and failure in network.

In this research the traffic flow control (TFC) algorithm in SDN traffic engineering is introduced so as to prevent too much traffic flow on the given link. The shortest path algorithm is a part of traffic flow control algorithm which is used to find the shortest path from source to destination. The proposed algorithm defines a limit of bytes on data travelled from multiple source without any drop of packets so as to prevent the traffic on the link.

Comparison is made between Congestion Avoidance Algorithm and the proposed algorithm. Results of the evaluation work prove that our approach has improved the network performance in terms of throughput, jitter and latency.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

SDN	Software Defined Networking
TFC	Traffic Flow Control
CA	Congestion Avoidance
VM	Virtual Machine
ONOS	Open Network Operating System
ONF	Open Networking Foundation
FIB	Forwarding Information Base
RIB	Routing Information Base
NETCONF	Network Configuration Protocol
API	Application Programming Interface
RFC	Request for Comments
SNMP	Simple Network Management Protocol
SSH	Secure Shell
TE	Traffic Engineering
QOS	Quality of Service
CSPF	Constrained Shortest Path First
MPLS	Multiprotocol Label Switching
SPF	Shortest Path First
NIC	Network Interface Card
NOS	Network Operating System

TLS	Transport Layer Security
OVSDB	Open vSwitch Database
ARPANET	Advanced Research Projects Agency Network
IP	Internet Protocol
XMPP	Extensible Messaging and Presence Protocol
HTTPS	Hypertext Transfer Protocol
QOE	Quality of Experience
ICT	Information and Communication Technology
IP	Internet Protocol
ATM	Asynchronous Transfer Mode
TCP	Transmission Control Protocol
CRUD	Create, Read, Update and Delete

Chapter-1

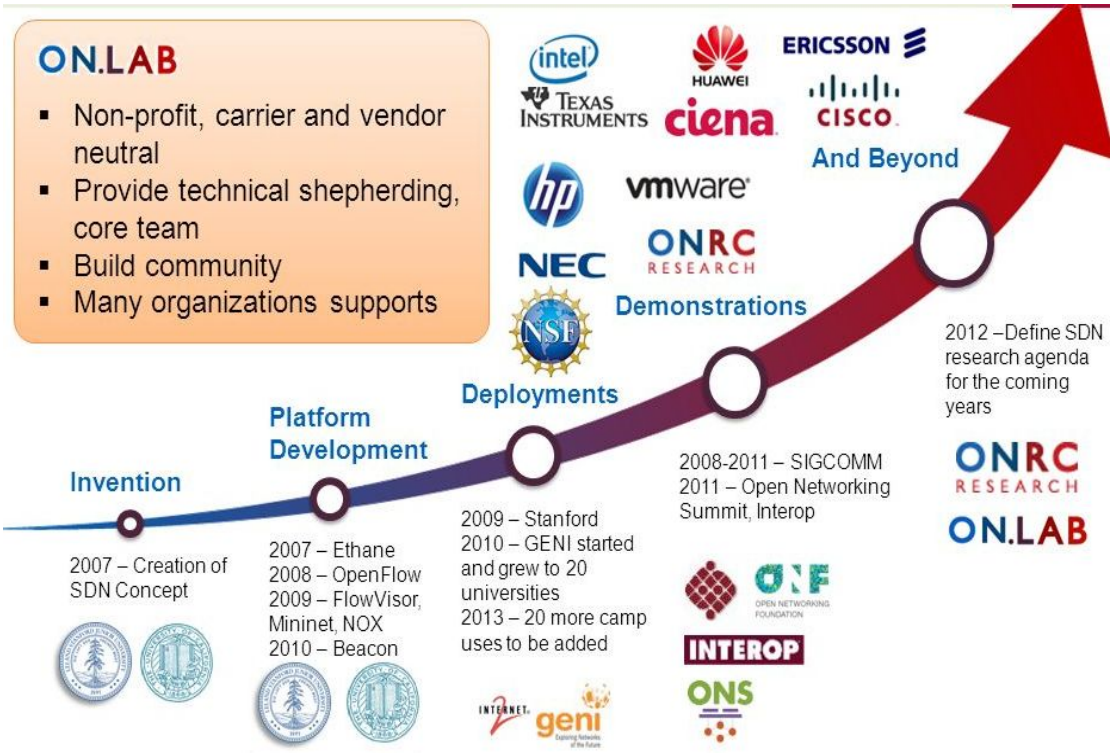
Introduction

This chapter presents the Software Defined Networking concept, its advancements and portrays the motivation along with characteristics in detail.

1.1 Evolution of Software Defined Networking

Software Defined Networking (SDN) is an architectural invention that proposes the separation of control plane and the info plane allowing their independent advancement and evolution. Ethane was among the initial SDN initiatives. This model –was proposed for enterprise uses and networks centralized control architecture. Nevertheless, this implementation needed the deployment of custom switches (Linux , OpenWrt) to support Ethane Protocol. Figure 1.1 show the timeline of SDN.

Figure 1.1 Software defined networking Timeline [2]



In present scenario , the principal crystallizing of SDN is usually Openflow , an open architecture initially developed as a way of allowing researchers to run experiments on heterogeneous networks without affecting the real users traffic. Motive of Openflow is to provide more configuration options in the switch dataplane. The Openflow specification establishes the rules of communication between data plane and a centralized control plane and allows the entire network to be controlled through users software applications i.e. Application Programming Interface(APIs). The Open Networking Foundation ONF[22] brings together about 90 companies and is focused on releasing, adopting and promoting of OpenFlow specification.[1]

1.2 Traditional approach and Modern approach

In traditional networking devices, the control plane and data plane was tightly coupled. The software is bundled with hardware and interfaces are vendor specific which is the limitation of this approach. Due to vendor specific, vendor writes the code and there are long delays in providing new features and functions. Figure 1.2 shows the traditional networking devices.

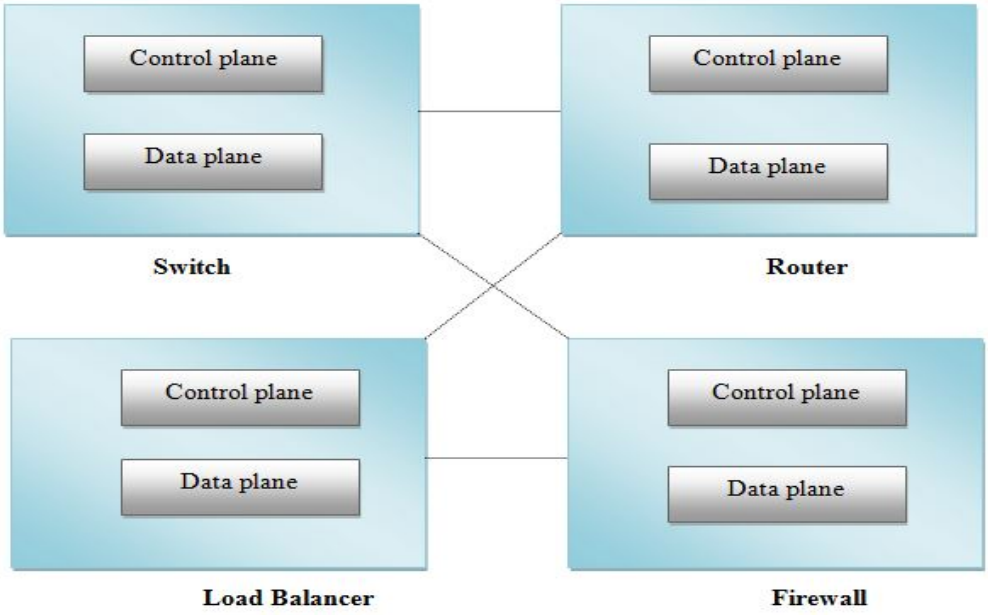


Figure 1.2 Traditional Network Devices

In modern approach, change in behavior of existing devices and creating new application is very difficult task due to vendor specific.

By changing the control devices to the central place, the SDN can help to resolve this issue. This changed control method is known as SDN/Open Flow controller.

SDN is a rising approach to use open protocols that has become increasingly popular in recent years. The basic idea behind this network architecture is to split up the control and data planes [2] in switches and routers or we can say the control is decoupled from hardware and implemented in software. Using central console anyone can shape network traffic without altering the configuration of individual devices like switches. Figure 1.3 shows the modern approach

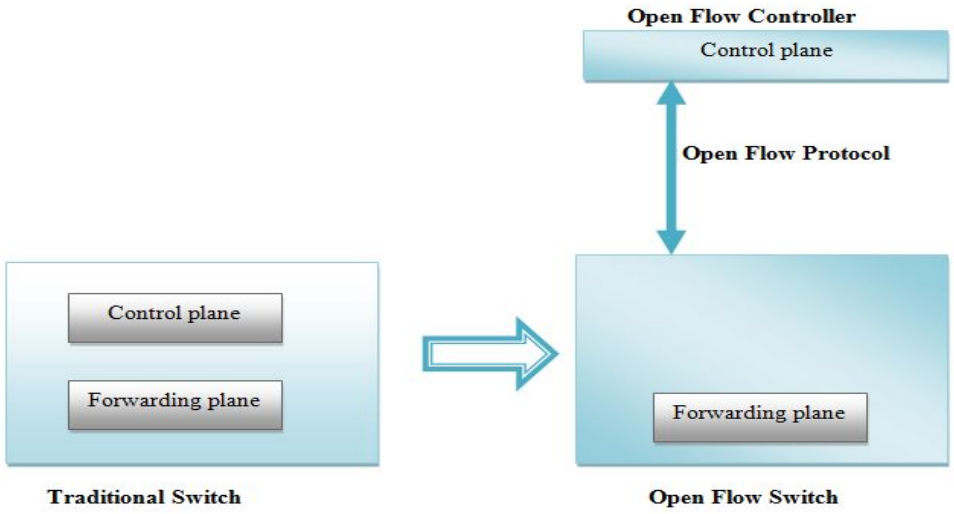


Figure 1.3 Modern approach SDN

1.3 Motivation

Software Defined Network has brought big modification in the networking field. It has changed the concepts of network totally. Because of SDN the control plane has become centralized, programmable, flexible and more scalable. As a total effect, it is easier to accordingly modify the network. Traditional network cannot guarantee the data transfer and the traffic flow from the network. It is one of the disadvantages of the traditional network. Day by day the network size is increasing and data circulation also increasing so the problems faced in the current systems are rising. If this proceeds it shall become a hindrance for the technical advancement. The current network will be changed by SDN thus. Because a new field it needs customization and even more research to improve the SDN network in the event that not it might backfire. That is why I have tried to work with the SDN network and lead on its development. The architecture of SDN is shown in Figure 1.4.

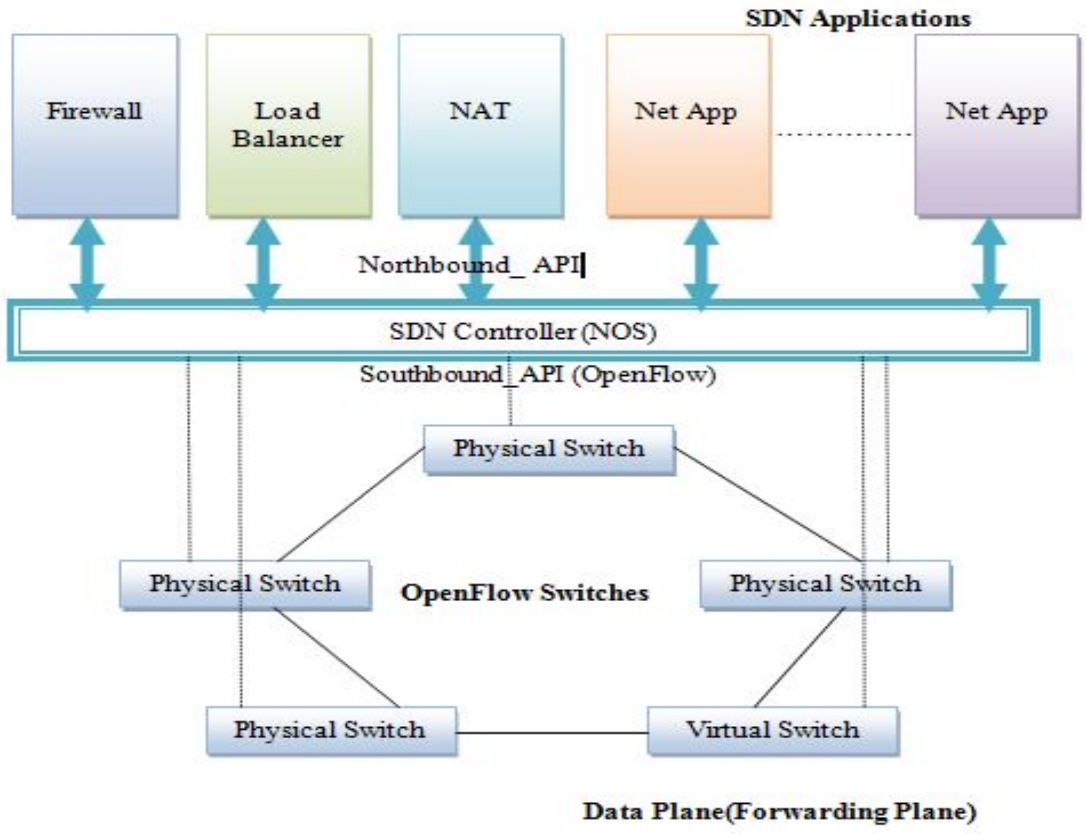


Figure 1.4 SDN architecture

1.4 SDN Control plane and Data plane

1.4.1 Control Plane

The control plane creates a local data set which in turn creates the forwarding table entries, which in turn are hold by the data plane to forward datagram's between incoming and outgoing ports on a network device. The functions used to store the network details are called the routing information bottom (RIB)/Routing tables, Topology Tables, Neighbor tables. The RIB can be kept consistent via the exchange of messages between other control planes within the distributed system. Forwarding Details base contains the forwarding table entries. Forwarding Information Base(FIB) is often stored on both data plane and control plane. FIB is certainly generated after RIB turns into consistent and stable. The control entity develops global look at of the topology that satisfies certain control parameters. This abstract watch of network is definitely programmable and builds from all the details gathered [3]. The figure is shown below:

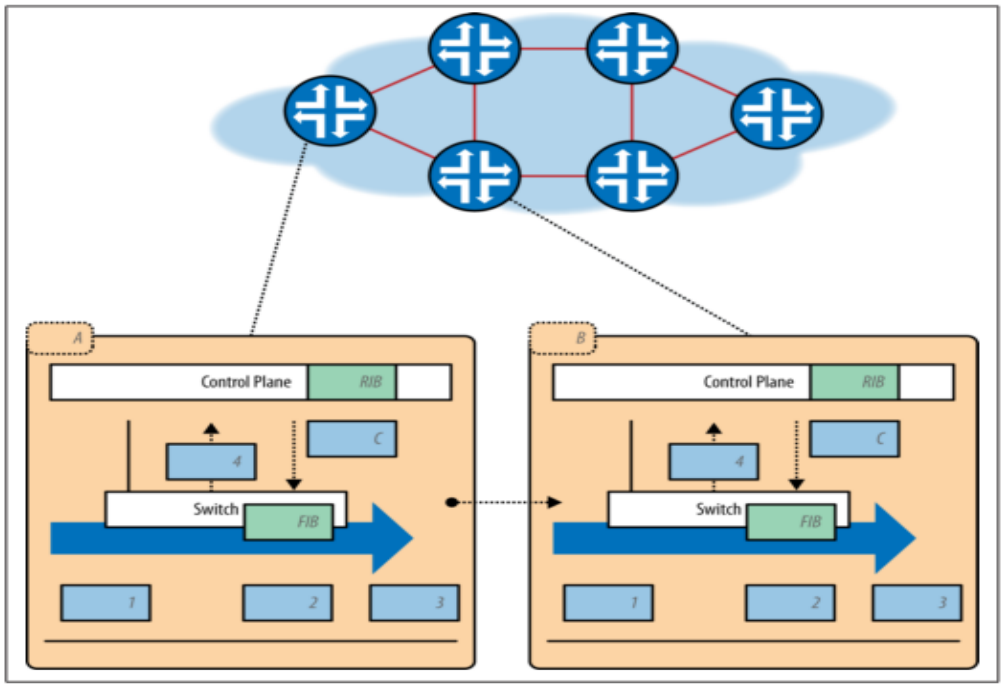


Figure 1.5 Control and data planes of a typical network [3]

1.4.2 Data Plane

Data plane is also recognized as Forwarding plane The data plane manages incoming traffic with the help of a number of operations which perform various authorization and authentication tests. A verified datagram is then processed by the data plane with the help of FIB which was earlier created by the Control Plane. The process is relatively fast as nothing is done except matching of specific rule from the FIB. When a packet doesn't match any specific FIB entry it is received by the control plane for further operations with the help of the RIB. RIB table can reside in number of locations software, CPU, GPU, Intel or ARM etc [3]. The figure 1.6 shows separation between two planes.

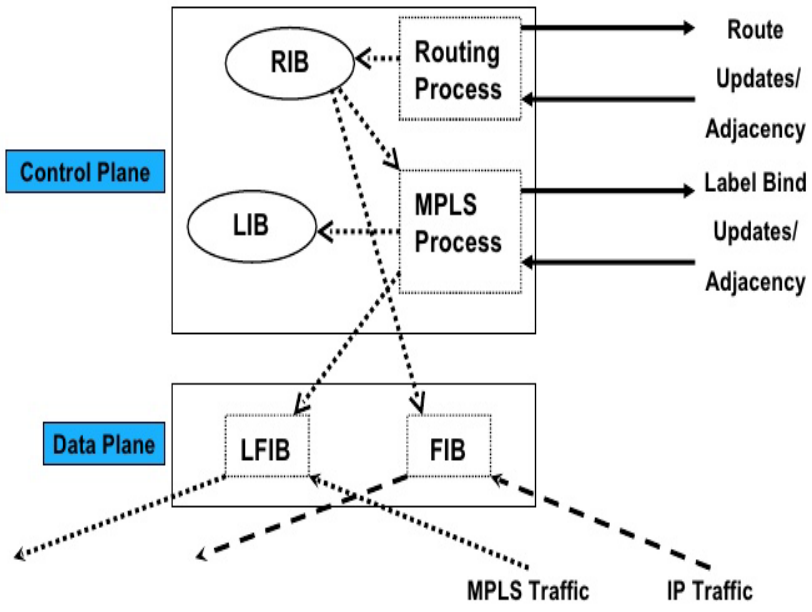


Figure 1.6 Typical network device of control and data plane [3]

1.5 Components of SDN

The components of Software Defined Networking are explained below:

1.5.1 Northbound interface

It is a network component which is used to communicate with a higher level component (South bound interface)

1.5.2 South bound interface

It is used for communication between forward device and controller. Popular southbound interfaces are OpenFlow and ForCES.

1.5.3 Controller

The controller instructs the devices to what actions they should perform through southbound API. OpenFlow is becoming the most efficient way for communication between SDN controller and switches.

1.5.4 Forwarding device

Pure SDN switch acts like dumb switch which performs only those functions that are provided by the controller.

1.6 Characteristics of SDN

The main characteristics of SDN are following:

- Control plane and data plane are decoupled in SDN.
- Network is programmable through software applications.
- It is agile.
- It is centrally managed.
- It is open standard based and vendor neutral.

The **Table 1.1** of various SDN protocols is mentioned on next page

Protocols	Release Date	Purpose	RFC	Interface	SDN Controllers	Benefits	Short Comings	Existing Security
OpenFlow	28th Feb, 11	Open Flow[4] is a protocol which guide network switches with help of server so as to send packets. This methodology, known as SDN, allows for more valuable use of network resources as compared to the traditional networks.	No RFC (part of ONF)	Southbound	Open Day light, RYU, ONOS, Open Vertex	Flexibility in how the network is used, operated, and sold.	ForCes: can control the entirety of the device's datapath and acknowledge request mechanism NETCONF: devices may support routing protocols. OVSDB: used to configure the OVS XMPP: immediate messaging and online existence recognition	TLS 1.2

ForCES	March, 10	Uses the capabilities of existing transport protocols to specifically address protocol message transportation issues, , such as how the protocol messages are associated with different transport media (like Ethernet, ATM)[5]	5810	Southbound	Vendor specific	Reliable, secure, congestion control	NETCONF: configuration management SNMP: meet ForCES requirements XMPP: used for instant messaging in the market today	-----
NETCONF	December , 06	The NETCONF protocol specification is an Internet Standards Track document. NETCONF provi des a better way to establish, operate, and delete the configuration of network devices. The NETCONF protocol keeps an Extensible Markup Language (XML) for the configured data and protocol messages.[6]	4741	Southbound	Open Day light, RYU, ONOS, Open Contrail	big advantage of NETCONF over SNMP is how the protocol works when manipulating a group of semantically related configuration data	ForCes: provide access ctrl mechanism. SNMP:Comparing to snmp ignores state data RESTCONF: provide an additional simplified interface that follows REST-like principles and is well-suited with a resource-oriented device abstraction.	SSH/TCP

SNMP	December 8th ,14	Simple Network Management Protocol (SNMP) is a popular protocol for management of networks. It is used for gathering information from, and configuration setting of a, network devices, such as servers, switches, and routers on an Internet Protocol (IP) network.[7]	1157	Southbound	Open Day light	The main advantage of SNMP is that it is an open standard. Open protocols are made to fight the wasted work and costs when one particular producer develops its "proprietary" process that only it'll support	NETCONF: Not support retrieval of complete device config as whole OpenFlow: Suffers poor performance of bulk data transfer.	SNMP V2
BGP	January , 89	Border Gateway Protocol (BGP) is a standard gateway protocol [8] designed to switch routing and information within autonomous systems (AS) over the Internet. The protocol is likely as a path vector protocol.	1105	Southbound	Open Contrail	• Fast convergence and fault tolerance. • Enhanced load balancing capabilities.	OpenFlow: used on a internal corporate network OSPF: support for routing.	TCP

OVSDB	July 18th ,13	The OVSDB protocol is used in a control cluster, along with other managers and controllers, to deliver configuration information to the switch database server. Controllers use Open Flow to recognize packet flows detail with help of the switch.	7047	Southbound	Open Day light, Open Contrail, RYU, ONOS	IT professionals can determine the number of individual virtual bridges within an Open vSwitch implementation, allowing a network engineer to create, build up and remove ports and tunnels from a bridge	OpenFlow: perform management for individual data flows NETCONF: provide programmatic access to the management plane of a network	TCP
XMPP	January ,99	Extensible Messaging and Presence Protocol (XMPP) is a communications protocol for message-oriented , middleware based on XML (Extensible Markup Language). It enables the real-time switch of structured with extensible data involving two or more network entities.	6120	Southbound	Open Contrail	robust and powerful, owing mostly to its standardized and decentralized nature	There is no official client or server for XMPP. A decentralized system, when someone *is* using an XMPP service—Google Hangouts, Messenger, WhatsApp, etc…—lack of branding or openness from service provide they often do not know it is XMPP.	TCP

RESTCONF	January ,17	RESTCONF uses HTTP methods to supply CRUD(Create.Read,Update and Delete) operations on a conceptual data store containing YANG-defined data, which is well-matched with a server that apply NETCONF data stores. The RESTCONF protocol does not specify any mandatory operation resources.[9]	8040	Northbound	Opendaylight, ONOS.	RESTCONF uses HTTP methods to provide CRUD operations on a conceptual datastore containing YANG-defined data, which is compatible with a server that implements NETCONF datastores.	----------	HTTPS

1.7 OpenFlow Protocol

- OpenFlow is a set of protocols and an API.
- It is a standard protocol for interacting with forwarding behavior of switches from multiple vendors.
- It provides us a way to manage the activities of switches throughout the network dynamically and programmatically.

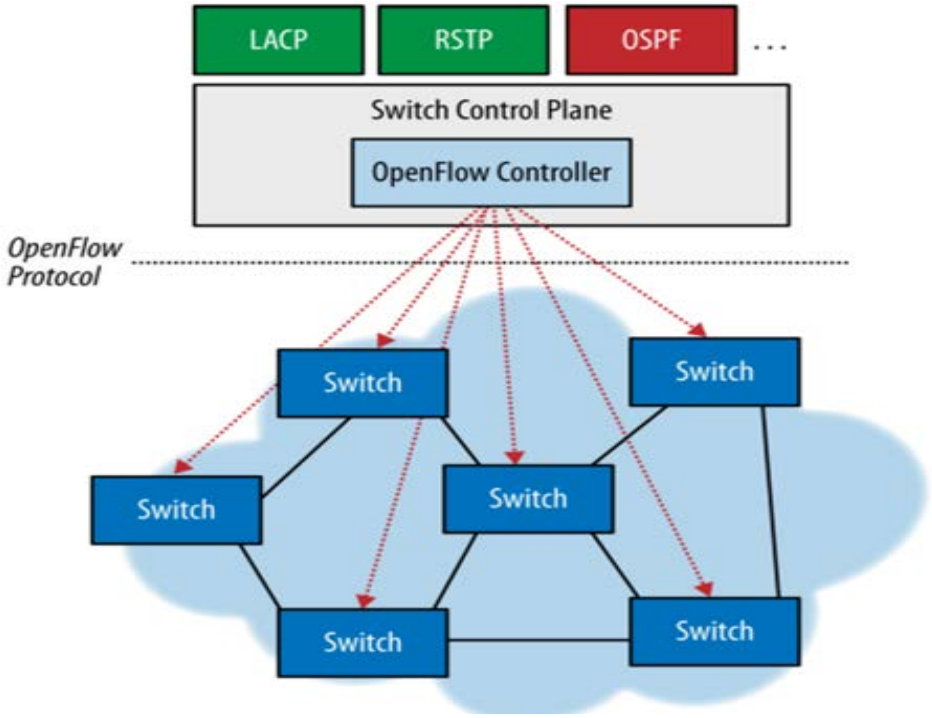


Figure 1.7 OpenFlow Protocol [3]

1.8 Methodology

- A shortest path first algorithm i.e. Bellman Ford is used to find shortest path from source to destination such that total sum of cost is minimum and this algorithm is part of TFC algorithm.
- In TFC algorithm, When the traffic flows from 1st client (h1) to 1st server (h5) and simultaneously if the 2nd client (h2) send the traffic flow on same path to the 2nd server(h6) then network card may fails.
- To overcome this problem the limit is fixed i.e. 300000 bytes.
- If the quota of any link h1-h4 or h2-h6 is greater than 300000 bytes then that link is blocked so as to control the volume of traffic.

Chapter-2

Literature Review

This chapter includes the review of traffic engineering concepts their objectives, techniques and existing algorithm along with related work.

2.1 Review of Traffic Engineering Concept

Traffic engineering (TE) is a network application which studies Traffic measurement and management of network traffic. [10] In Traffic Measurement, there is monitoring and analyzing real time traffic or we can say prediction of traffic and Traffic Management consists of Traffic load balancing, QOS(Quality of service) guarantee scheduling, Energy saving scheduling.

The objective of TE is to optimize the performance of networks. The traffic engineering involves the path computation between a source and destination or computation of various paths to share the load or we can say a way of traffic splitting.

Traffic engineering has designed better routing procedure to guide network traffic to improve utilization of network resources.

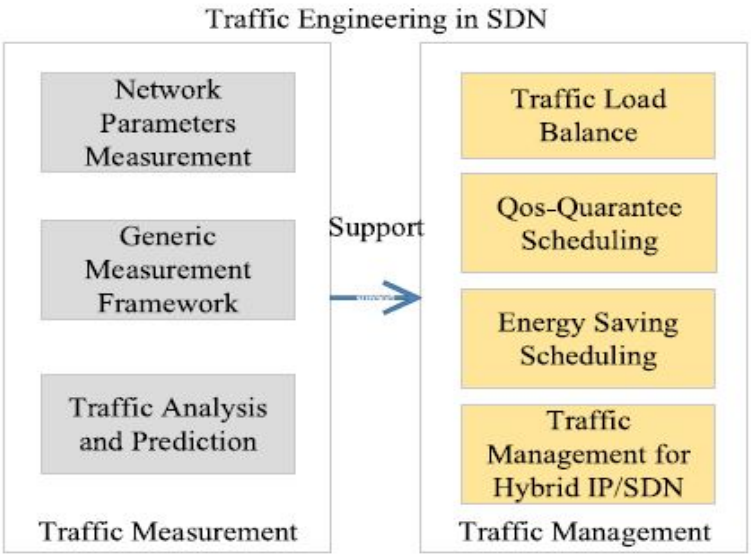


Figure 2.1 TE Framework [10]

2.2 Traffic Engineering Objectives and Techniques

The TE involves the various new technologies and optimization mechanisms [11]which are required continuously. After designing of TE solution, it was required to select the performance objective very carefully, as various objectives can be mutually exclusive.

The various performance objectives and techniques [12] for optimization are following:-

2.2.1 Congestion Minimization

Congestion is one of the biggest problem in networks as it affects packet loss, delay and jitter.[11]

Congestion minimization can be done by various techniques:-

- The network resources can be shared by multiple traffic streams.
- The network resources can be reallocated by redistributing traffic over the network infrastructure.

  The access to congested resources can be denied. The TE system can only allocate uncongested resources to new demands if there is congestion.

2.2.2 End to End delay Minimization

This type of objective has a great impact on Quality of service and quality of experience. E2E (End to End) delay minimization is necessary for critical real time communications. The most common techniques to minimize this delay is Constrained-shortest path first(CSPF), where the E2E delay is used as a requirement for the path selection.

2.2.3 Packet loss Minimization

This objective can also be evaluated as per-flow or network wide. Same as congestion, failures in the network can also be the reason for packet loss such as forwarding switches and links which requires various techniques to increase the failure recovery capabilities of network .This objective is to increase flexibility by

means of unnecessary resources to be used in case of failure. If multiple paths are existing then to express traffic between a given source-destination pair, traffic must be re-routed among available paths.

2.2.4 Energy consumption Minimization

This objective does not essentially match a network performance parameter. It is used in scope of green computing, which is having purpose to lower the environmental impact of Information and communication technology.

This objective is optimized either by reducing amount of dynamic resources or by modification in the rate of network operation to the existing workload. Consumption of energy is reduced when there is less traffic gathered into few paths. The two different objectives can be mutually exclusive as the energy consumption minimization cannot be achieved at the same time when this approach is applied.

2.2.5 Quality of Experience Maximization

QOE(Quality of Experience) measures the performance of ICT (Information and Communication Technology) product with the objective considering technical parameters like QOS(Quality of Service)[12]and subjective psychological parameters. It is a parameter which is affected by the elements defined in end to end transmissions, including end devices, factors such as light and the network performance. Thus, the optimization of network performance is required by QOE maximization.

2.2.6 Resource utilization Optimization

Resource utilization optimization is another objective. For example, Bandwidth, buffer space and computation ate the resources that need to be used efficiently as they can impact congestion & other parameters.

Moreover, a utilization of resources helps the network operators to complete the higher no. of services demand without rising prices which means the good utilization of network resources keep the network operators to allocate much amount of traffic.

2.3 History of Traffic Engineering

TE is understood as control issue where element in charge of TE act as a controller in an adaptive feedback control system in view of Awduche[12]. The various control actions must includes the modification of parameters associated with routing and modifications of the attributes and constraints associated with resources.

TE in packet networks has been controlled using different approaches, as mentioned in RFC(Request for comments) 3272. At last this first proposal was not suitable for TE because they did not satisfy the requirements.

When the internet came into existence the adaptive routing protocols used in ARPANET (advanced research projects agency network) were substituted by dynamic routing protocols. Protocols such as Interior Gateway Protocols that run on internet were neither perfect for Traffic engineering because selection of root was based on shortest path algorithms.

Overlay models were used like IP(Internet Protocol) over ATM(Asynchronous Transfer Mode) as an approach to take advantage of TE, with the introduction of technology like virtual circuits and point to point links, the TE operation become easier.

Shortest path first (SPF) algorithms were introduced in next iteration in which there was unfair usage of the network resources, where the shortest paths end were congested and other are underutilized.

Then, Traffic splitting was introduced by Equal cost multipath protocol in which traffic was splitted among shortest route.

2.4 Review of traditional traffic engineering techniques:

The traditional network includes IP based TE and and MPLS based TE.

2.4.1 Internet Protocol based Traffic Engineering

The issue of multipath traffic load balance is solved by optimizing IP routing algorithm to avoid network congestion.[12]

E.g.:- F & T introduces a neighborhood search algorithm which is based on link weights OSPF which are static to maintain the calculation of routing strategy and then get multiple shortest path to achieve traffic load.[13]

2.4.2 Multiprotocol label switching based Traffic Engineering

In order to avoid IP based TE effects so researchers introduced another solution in which network packets are forwarded by MPLS (Multiprotocol label switching) instead of IP headers.[14]

In MPLS, the broadcasting of packets between edge nodes of an MPLS domain with label switched paths (LSP) and the forwarding at each node are done on the basis of previously allocated labels which enhances the network performance. MPLS is good for TE because it provides most of functionalities available from the overlay model in integrated manner and at lower level.

Jeong *et al.*[16] "**Application aware Traffic management"** This paper focuses on the traffic management scenario with two network functions, firewall and bandwidth manager. This scenario is implemented as ONOS application communicating with packet inspection instances for classification of traffic. This system will manage network traffic over SDN with low cost as compared to traditional approach.

Ren *et al.*[17] "**Enhancing Traffic Engineering performance and flow management**" This paper describes the a method named as Flow routing and splitting which determine an appropriate SDN switch for every flow and optimizing the traffic splitting fractions for every SDN switch with outgoing connection to reduce the maximum link utilization.

Chaitra and Shylaja[18] "**Intelligent Traffic management**" The idea is to utilize the SDN model to manage network traffic over wide area network(WAN) from nodes of wireless networks depending on changing network conditions and type of applications. The application named as routing system allows mobile network operators to gain better quality of networks.

2.5 Existing Algorithm

In this section, brief description of existing algorithm is given:

2.5.1 Congestion Avoidance algorithm

The SDN controller links almost all switches in the network. After that, it calculates the very best path between switches using its personal routing algorithm and begins sending data. Up to this true point, this works seeing that the heritage network. Nevertheless, unlike the legacy network, the status is collected by it information of most ports regularly. With this given information, the controller computes the utilization of every interface. When there is any port in whose utilization is more than 70%, it predicts that switch shall make congestion. It changes all of the routes passing during the change into new route which in turn bypasses that switch. So that it can block any fresh data become inserted in to that busy change which will outcomes in the congestion. Following setting fresh route, the network works as typical. However, it checks the utilization of the also switch that was busy. If the use of that switch is decreased to 50%, the controller decides to recuperate the bypasses in to the original path. For the reason that, in congestion- much less situation, the initial route increases results compared to the bypass based on the routing formula. Therefore, Congestion Prevention algorithm repeats it in order to detect the usage of any kind of change is a lot more than 70 percent [15].

The Figure 2.2 shown on the next page is a flow chart of this algorithm.

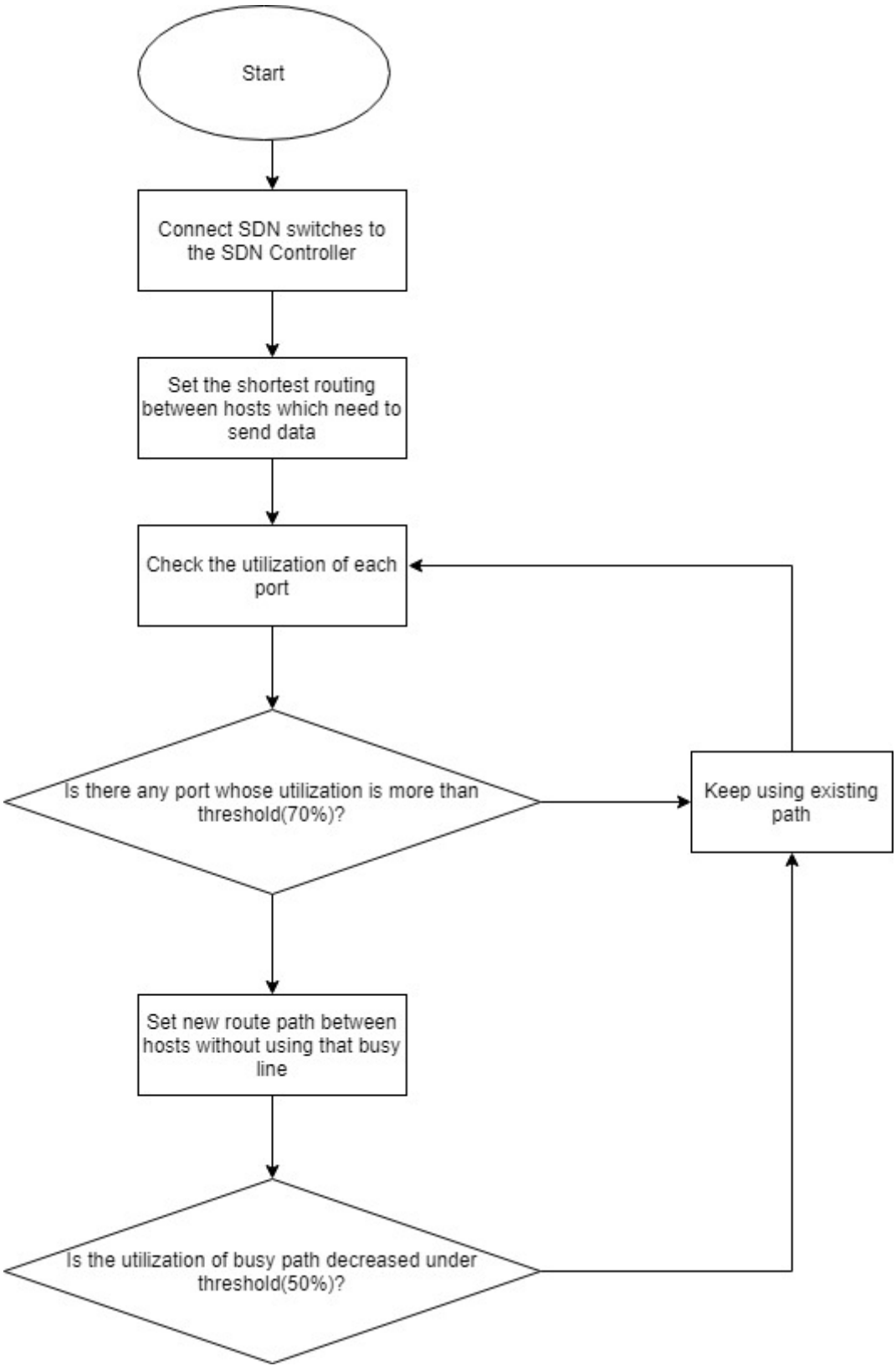


Figure 2.2 Flowchart of the algorithm [15]

Chapter-3

Problem Statement

This chapter discusses the brief overview of the problem with objective and topologies considered.

3.1 Problem Description

Traffic flow control is the crucial factor in Traffic Engineering. To address this issue a simple and efficient algorithm which controls too much traffic need to be implemented. The problem here is how to utilize the given link or path so as to control traffic flow without affecting normal data traffic.

3.1.1 Objective

- To control too much traffic flow from affecting normal data traffic in any organization or university.
- Due to increase in number of hosts in a broadcast domain or broadcast storm in the university, it is necessary for the network administration department to control traffic. Thus, NIC (Network Interface Card) will not work correctly when the particular NIC exceeds the predefined value. The limit is reset after one data and NIC will work again.
- The proposed algorithm, TFC with using functionality of shortest path algorithm, bellman ford in the SDN network need to be implemented so as to utilize the same link better.
- Comparison of proposed framework with Congestion Avoidance algorithm in terms of network performance.

3.1.2 Problem Formulation

A SDN design consists of four parts: OpenFlow switches, POX controller, links and Hosts. I have referred to a topology already created as my initial topology for my work purpose and then reformed the initially processed topology into a new topology structure as per the research requirements. I have re-constructed the already existing topology for performing the flow management over the traffic passing through the network. Basically the flow needs to be controlled on the basis of some specific attributes. Below is the initially used topology:

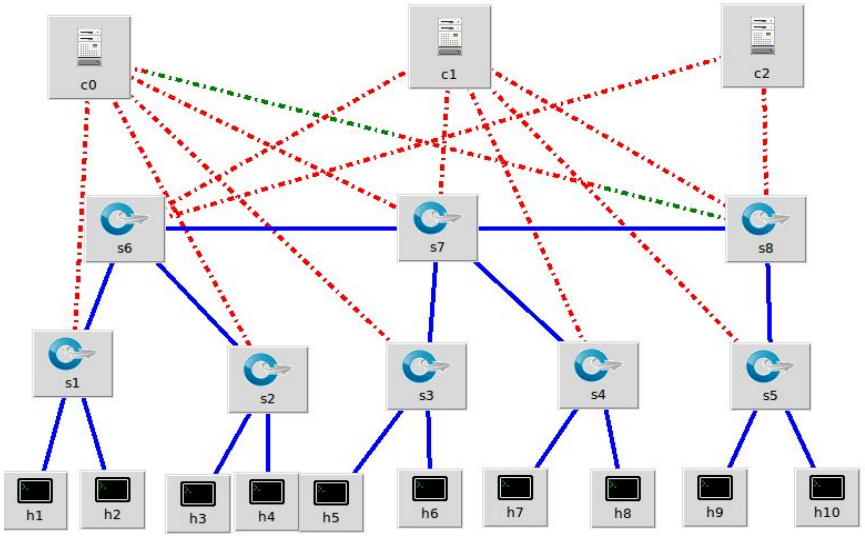


Figure 3.1 Initial Topology

After reforming the initially processed topology, I have added the link from S1-S7 so that shortest path is calculated by bellman ford algorithm and Traffic Flow control algorithm will be implemented. The new topology looks like:

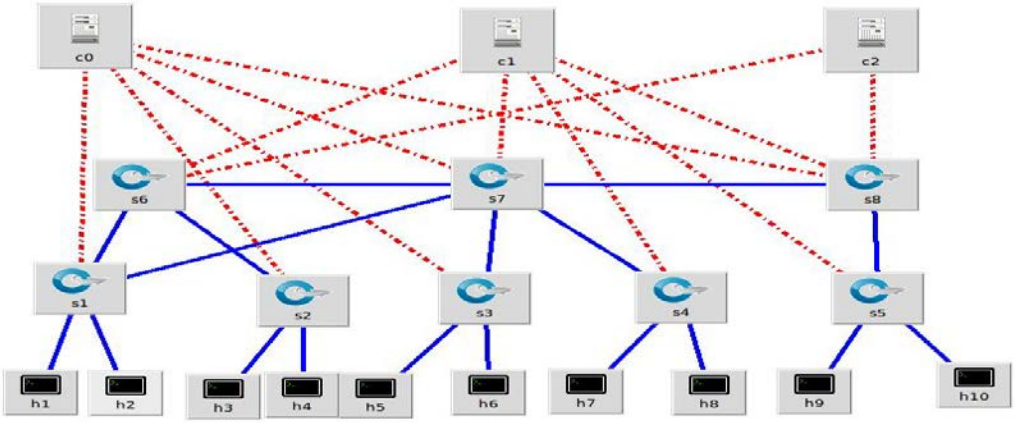


Figure 3.2 New-topology

Chapter-4

Proposed Framework

This chapter describes the proposed framework to the problem stated in the previous chapter.

4.1 Traffic Flow Control Algorithm

The proposed algorithm defines a limit i.e. 300000 bytes on the data travelled from multiple sources without any drop of packets so as to prevent the traffic on the link. The shortest path algorithm is a part of traffic flow control algorithm which is used to find the shortest path from source to destination. The function install path checks whether a path limit is greater than predefined limit or not.

The function handle flow status received counts the bytes received at each host and compare it with predefined limit and if any host limit greater than predefined limit then that path will be blocked or vice-versa.

The improved algorithm has better network performance as compared to congestion avoidance algorithm. Moreover, it can utilize the same link when there are multiple sources to send the packets on given link by reset the timer again.

The steps involved are given below:

Traffic Flow Control Algorithm

1. set limit = 300000 bytes, year, month //**Initialization**
2. **renew()**

//Function to check whether the year , month and day are valid or not

3. global year, month, day
4. flock = time.localtime()
5. if flock.tm_year > year:
6. return True
7. elif flock.tm_mon > month:*

8. return True
9. elif flock.tm_mday > day:
10. return True
11. else
12. return False

13. get_raw_path (src,dst)

//Bellman-Ford algorithm to get the shortest path

14. dist = {}
15. prev = {}
16. sws = switches.values()
17. for dpid in sw:
18. dist[dpid] = 10000
19. prev[dpid] = None
20. dist[src]=0
21. for m in range(len(sw)-1):
22. for u in sw:
23. for v in sw:
24. if adjacency[u][v]!=None:
25. wt = 1
26. if dist[u] + wt < dist[v]:
27. dist[v] = dist[u] + wt
28. prev[v] = u
29. w=[]
30. u=dst
31. w.append(u)
32. v=prev[u]
33. while q is not None:
34. if v == src:
35. w.append(v)
36. break
37. u=v
38. w.append(u)
39. v=prev[u]

40. w.reverse()
41. return w
42. path_install (self, dst_sw, last_port, match, event)

/*Function to install a path between this switch and some destination

and check whether path limit is greater than predefined limit or not */

43. if _renew():
44. global year, month, day
45. used.clear()
46. flock = time.localtime()
47. year=flock.tm_year
48. month=flock.tm_mon
49. day=flock.tm_mday
50. print “Renew”
51. u = _get_path(self, dst_sw, event.port, last_port, match, event)
52. #print time.time(), “path _install is called at:”, event.connection.dpid, u
53. if used[(match.dl_src)] is not None and used[(match.dl_src)]>limit and limit:
54. u = None
55. handle_FlowStatsReceived (self, event)

/* This function counts the bytes received at each host and compare it with predefined limit and if any host limit greater than predefined limit then that path will be blocked or vice-versa.*/

56. log.debug(“Got FlowStatsReceived: %s”, event.connection)
57. print time.time(), event.connection, “Got FlowStatsReceived”
58. for f in event.stats
59. for a,b in mypath.items()
60. #print “a=”, a, “b=”, b
61. #upload
62. if f.match.dl_src == a[0] and f.match.dl_dst == a[1] and f.match.tp_src==a[2] and a[2] is not None and a[3] is not None and f.match.tp_dst==a[3] and f.match.nw_proto==a[4] and f.match.tp_src!=0 and f.match.tp_dst!=0

and mypath[(f.match.dl_src, f.match.dl_dst, f.match.tp_src, f.match.tp_dst, f.match.nw_proto)][0]==switches[event.connection.dpid]:

63. #print
    “SSS:”, f.match.dl_src, f.match.dl_dst, f.match.tp_src, f.match.tp_dst, f.match.nw_proto,mypath[(f.match.dl_src, f.match.dl_dst, f.match.tp_src, f.match.tp_dst, f.match.nw_proto)][0]
64. print “upload) “, a[0], “->”, a[1], “:”, f.byte_count, bytes_received[(b[0],f.match.dl_src,f.match.dl_dst,f.match.tp_src,f.match.tp_dst,f.match.nw_proto)], f.byte_count – bytes_received[(b[0],f.match.dl_src,f.match.dl_dst,f.match.tp_src,f.match.tp_dst,f.match.nw_proto)]
65. if f.byte_count >=bytes_received[(b[0],f.match.dl_src,f.match.dl_dst,f.match.tp_src,f.match.tp_dst,f.match.nw_proto)]:
    used[(f.match.dl_src)]+=(f.byte_count bytes_received[(b[0],f.match.dl_src,f.match.dl_dst,f.match.tp_src,f.match.tp_dst,f.match.nw_proto)])
    bytes_received[(b[0],f.match.dl_src,f.match.dl_dst,f.match.tp_src,f.match.tp_dst,f.match.nw_proto)] = f.byte_count
    print “upload) used[“, f.match.dl_src, “]=”, used[(f.match.dl_src)]

66. if used[(f.match.dl_src)] > limit and limit:
67. msg = of.ofp_flow_mod()
68. msg.command=of.OFPFC_DELETE_STRICT
69. msg.match = f.match
70. switches[event.connection.dpid].connection.send(msg)
71. **timer**

    //Function to forward the data from source which is blocked after reset the time.

72. sent_sw={None}
73. for a,b in mypath.iteritems():
74. if b[0] not in sent_sw:
75. print time.time(), “send out ofp_flow_stats_request to dpid:”, b[0]
76. core.openflow.getConnection(b[0].dpid).send(of.ofp_stats_request(body=of.ofp_flow_stats_request()))
    sent_sw.add(b[0]) **//End**

The **figure 4.1** of flowchart is shown on the nextpage.

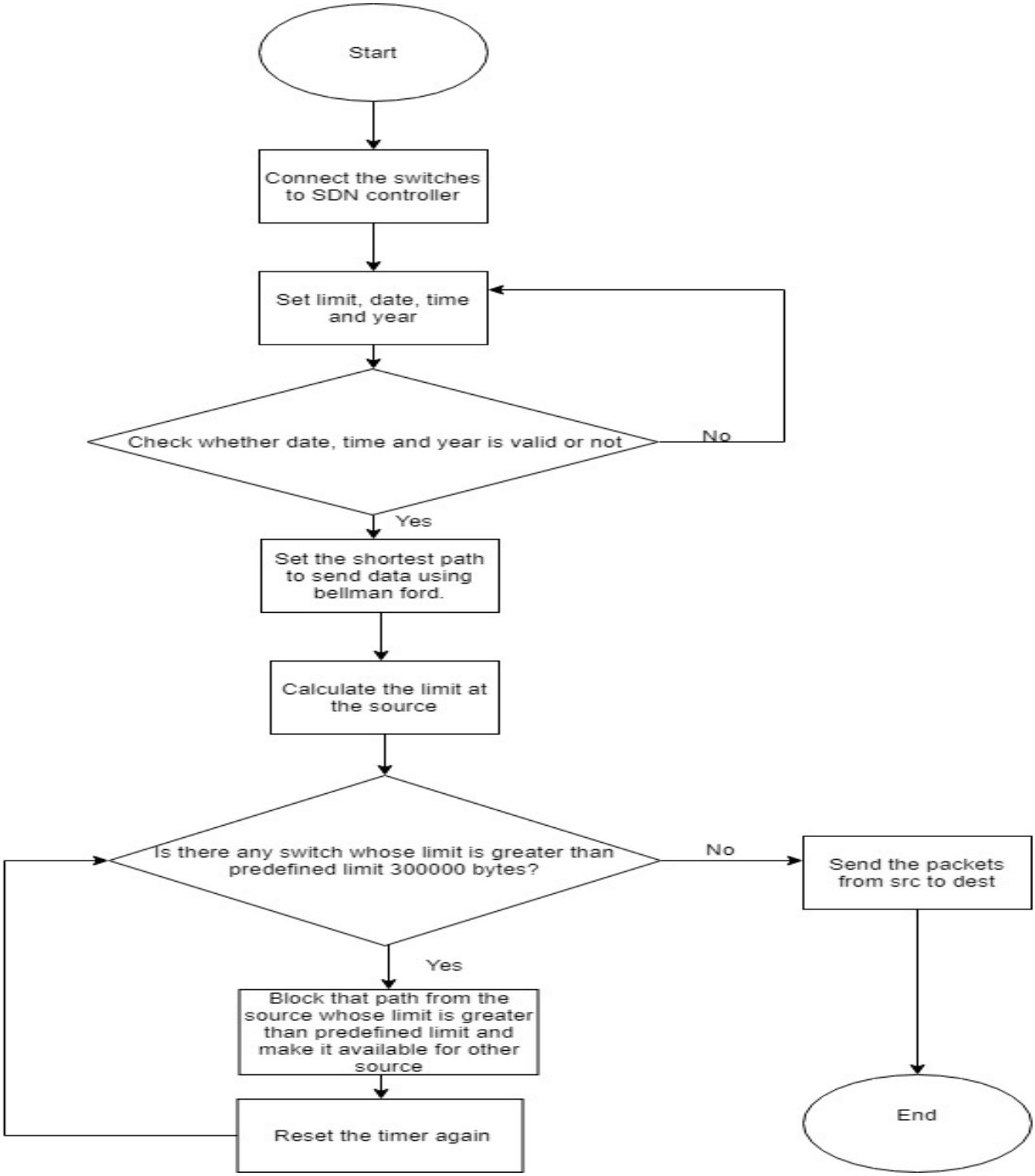


Figure 4.1 Flowchart of TFC Algorithm

4.2 Comparative Analysis

Parameters	**TFC algorithm**	**CA algorithm**
Objective	To control traffic on the same link.	To prevent the network congestion with rerouting.
Complexity	Less complex as it does not need to reroute the traffic on path other than original again and again.	More as traffic need to reroute .
Cost	Low as it works on original path.	High as it works on other paths as well
Throughput	High throughput which means more no. of data moved successfully from one place to another.	As compare to TFC algorithm , less no. of data can be moved from one place to another.
Jitter	jitter is lower and stable in case of TFC algorithm	Higher in case of CA algorithm.
Latency	Lower in case of TFC which means no. of packets can be transferred from one place to other with small delays.	Higher in case of CA algorithm.

Table 4.1 Comparative analysis

Chapter-5

Implementation and Experimental Results

This chapter emphasize on the implementation of algorithm with tools and comparison graph.

5.1 Mininet Emulator and Iperf tool

5.1.1 Mininet Emulator

[19]Mininet is the powerful tool to create the network of virtual links, switches , host and controllers. Mininet tool is used to create the topology in a network such as linear ring, tree topology etc. Mininet also supports the pox controller functionality which is used to control the forwarding device such as switches, routers and bridges etc.

Research, development, prototyping, testing, learning debugging is supported by mininet.

Mininet Functionalities:

- **CLI**
  Mininet has CLI (command line interface) which is aware of topology OpenFlow .
- **Custom topologies**
  Supports random topologies with set of parameterized topologies.
- **Out of box usage**
  Mininet can be used out of the box without programming.
- **Python**
  Mininet supports Python as a programming language.
- **System level regression test**
  Mininet support this feature which are repeatable and easily packaged.

Mininet is the best tool to experiment with the topologies. Mininet tool is installed in a Linux operating system and proposed application traffic_ctrl.py is built on the pox controller.

Advantages:

- **Easy installation**
  It is a prepackaged VM that can be launched on Vmware or VirtualBox.
- **High bandwidth**
  Provides typically 2Gbps total bandwidth on every hardware.
- **Scalability**
  Hundred of switches and hosts can be added.
- **Loads Faster**
  Boots at faster rate i.e. in seconds.

Limitations:

- The networks cannot increase the bandwidth available on single server.
- Mininet cannot run on NON-LINUX compatible applications and OpenFlow switches and It is not a major issue.

The figure 5.1 shown below is the Mininet Emulator:

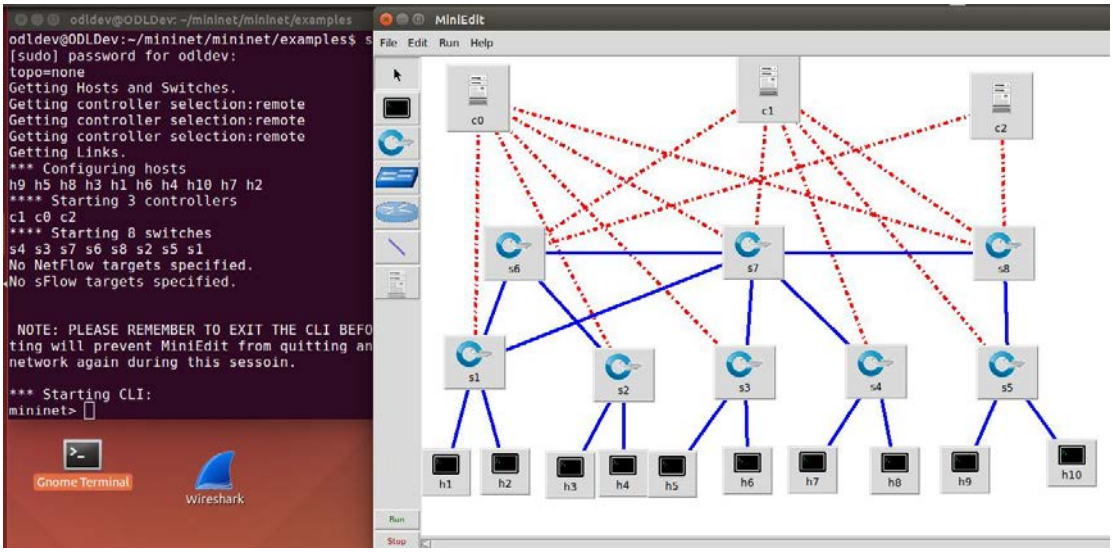


Figure 5.1 Mininet Emulator

5.1.2 Iperf

Iperf is the widely network testing tool to measure the performance of a network on client-server machine like measuring the TCP-UDP bandwidth , jitter, delay, data loss and throughput performance in a network. The Traffic flow control algorithm and congestion avoidance algorithm with the help of this tool.[20]

Iperf commands:

- iperf –s –u –i 1

  -this command starts the server machine. Figure 5.2 shows the running command of Iperf server on terminal.

- iperf –c <dest-ip> -u –b <bandwidth> -t <time interval>

  -this command is used to ping the host from client machine. Figure 5.3 shows the

  Iperf client command on terminal**.**

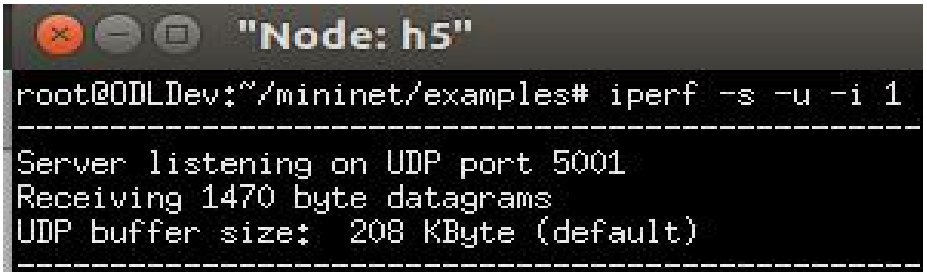


Figure 5.2 Iperf server command on terminal

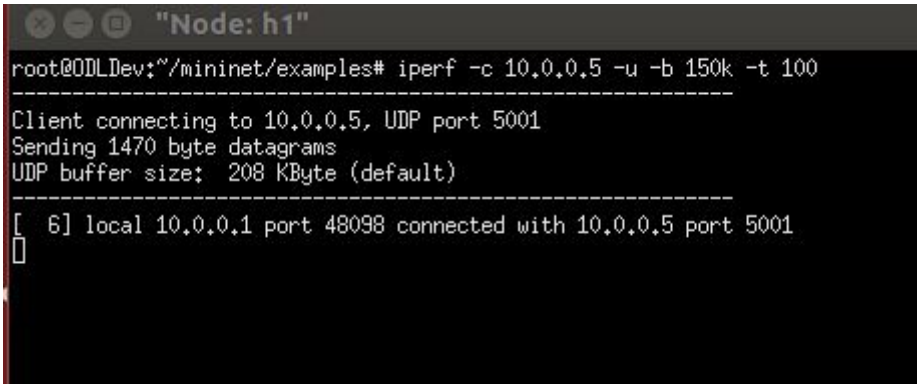


Figure 5.3 Iperf client command on terminal

5.2 POX Controller

POX is a freeware development platform for python based SDN applications. There is a rapid development and prototyping in this controller. Developers uses pox to develop the SDN applications with the help of python programming language.[21]

5.2.1 Components of POX

POX Components are the python programs that are called using command line. These components have different functionality.

Eg. Forwarding.l2_learning component is in the ~/pox/pox/forwarding directory

5.2.2 Installation of POX

POX Controller is already installed on mininet 2.2 VM image and if you want to install it on LINUX then you need to install mininet script.

5.2.3 Launching of POX

POX can be launched by pox.py program and specifying the components to use

Eg. Sudo python pox.py log.level –CRITICAL traffic_ctrl openflow.discovery

Here the application traffic_ctrl can be launched using above command.

The Simulation parameters, hardware requirements and their description is following in the Table 5.1. Simulation is performed on Linux Operating System.

Parameter	**Description**
Platform	Mininet 2.2, Iperf
Programming Language	Python
Operating System	Linux(Ubuntu 14.04)
RAM	4GB
Controller	POX

Table 5.1: Experiment Parameters

5.3 Procedure and Results

The proposed algorithm is implemented and results are mentioned below:

1) Run the python script(new_topo.py) shown in figure 5.4 and thus formed topology is shown in figure 5.5 as:

```
odldev@ODLDev:~/mininet/mininet/examples$ sudo python new_topo.py
*** Adding controller
*** Add switches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
h8 h4 h9 h1 h7 h3 h10 h5 h6 h2
*** Starting controllers
*** Starting switches
*** Post configure switches and hosts
*** Starting CLI:
mininet>
```

Figure 5.4 Python Script(new_topo.py)

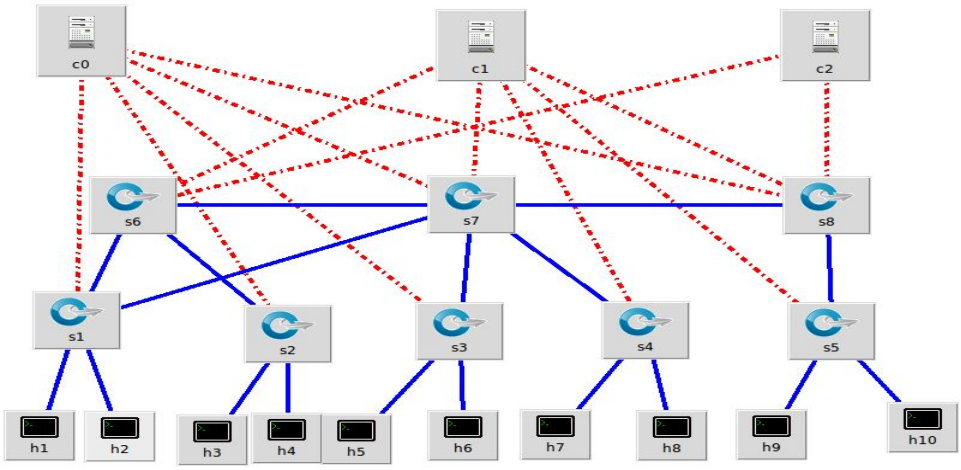


Figure 5.5 Topology

2) Run the file named as traffic_ctrl.py so as to control the flow of traffic using POX controller which is shown in figure 5.6

```
odldev@ODLDev:~/pox$ sudo python pox.py log.level --CRITICAL traffic_ctrl openfl
ow.discovery
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
Now: 2018 : 6 : 28
```

Figure 5.6 Running POX

3) Open the terminal screen(xterm) for h1,h2,h5 and h6 which is shown in figure 5.7

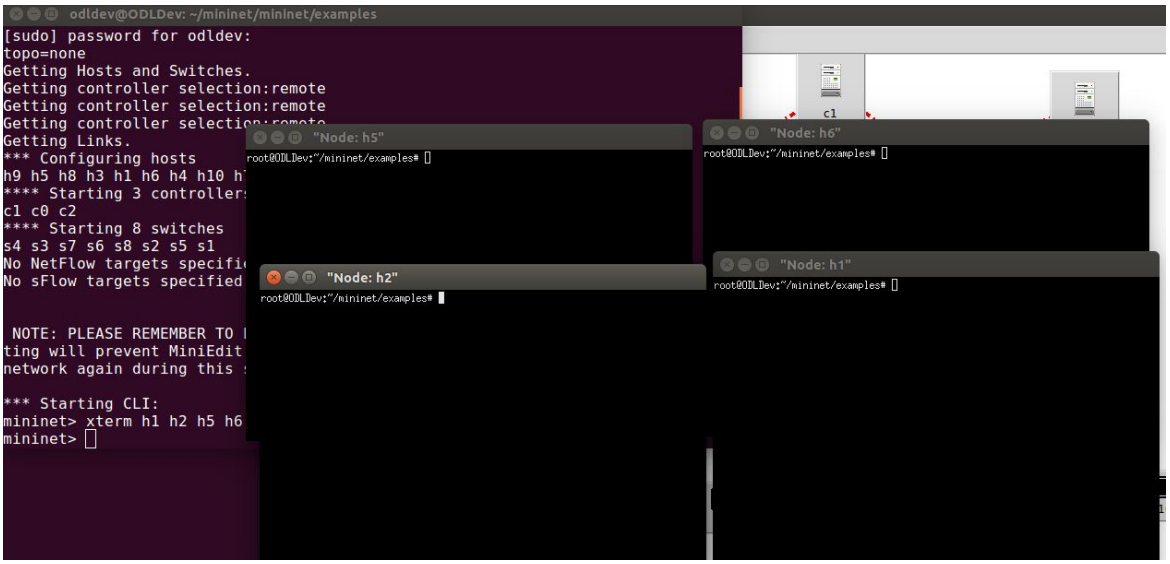


Figure 5.7 Terminal window of h1 ,h2 ,h5 and h6

4) Run the Iperf servers on h5 and h6 using commands which are shown in figure 5.8 and 5.9.

Figure 5.8 Iperf-h5 server

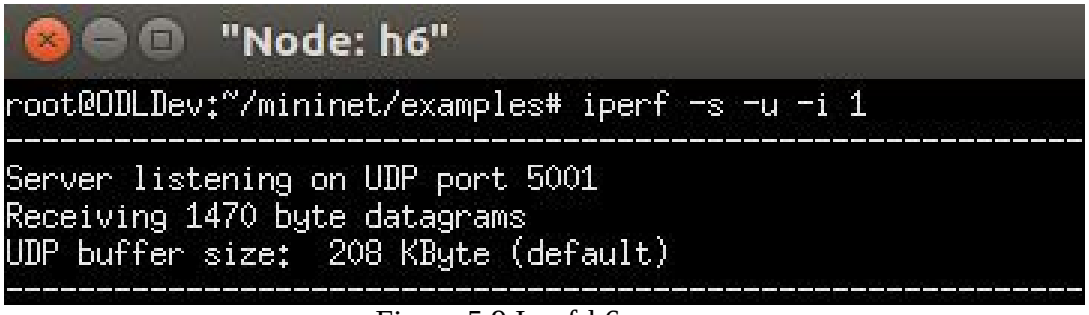


Figure 5.9 Iperf-h6 server

5) Run the Iperf UDP client on h1 firstly as shown in figure 5.10 and note the output of the pox controller as shown in figure 5.11.The shortest path using bellman ford

algorithm is also shown in figure 5.11 .From the output of the pox controller, when the used[00:00:00:00:00:01] h1 is greater than the pre-defined value, i.e.300000 bytes, h5 will not show any further information as described in figure 5.12. Because the h1->h5 is blocked.

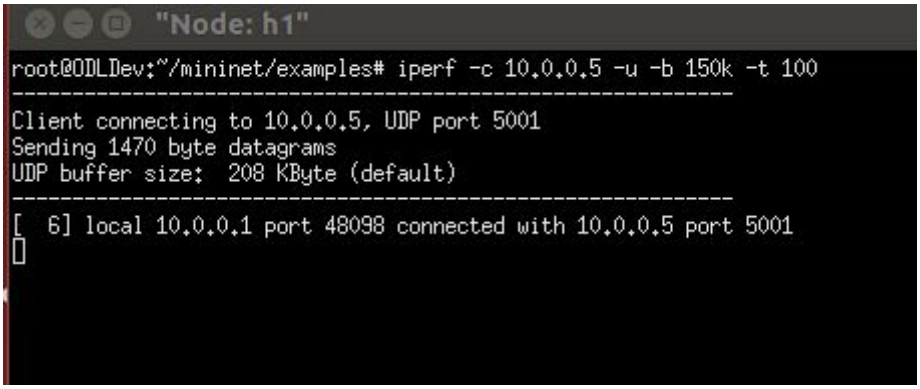


Figure 5.10 Client-h1

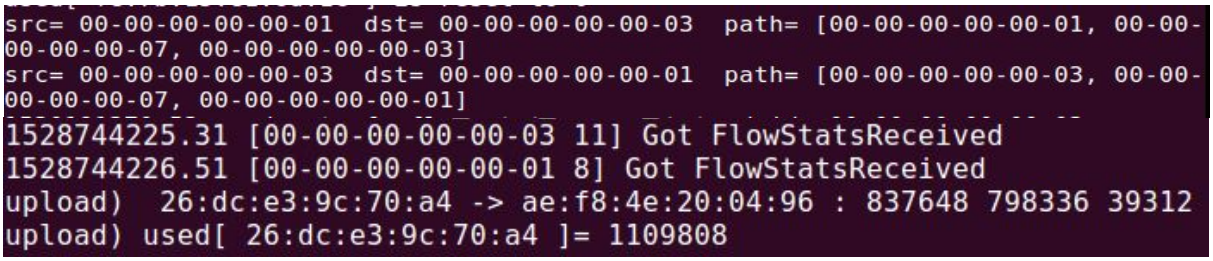


Figure 5.11 Shortest path and limit which is more than predefined limit

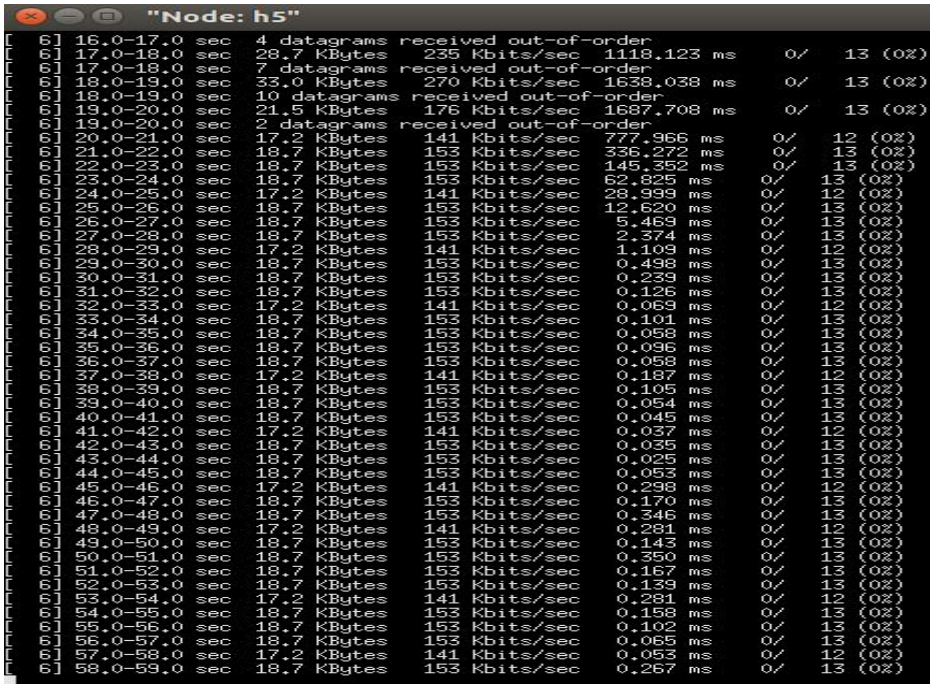


Figure 5.12 Result at h5

6) Run the Iperf UDP client on h2 as shown in figure 5.13 and result at h5 is shown in figure 5.14 as pre-defined value is not reached so h2-h6 link will not blocked and the obtained count is shown in figure 5.15.

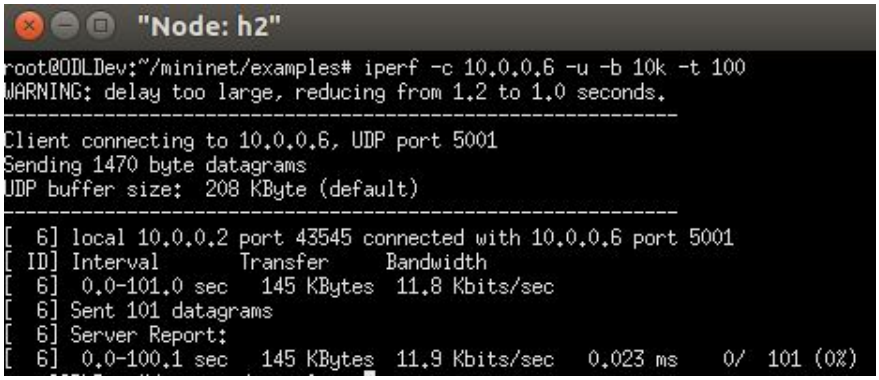


Figure 5.13 Result at h2

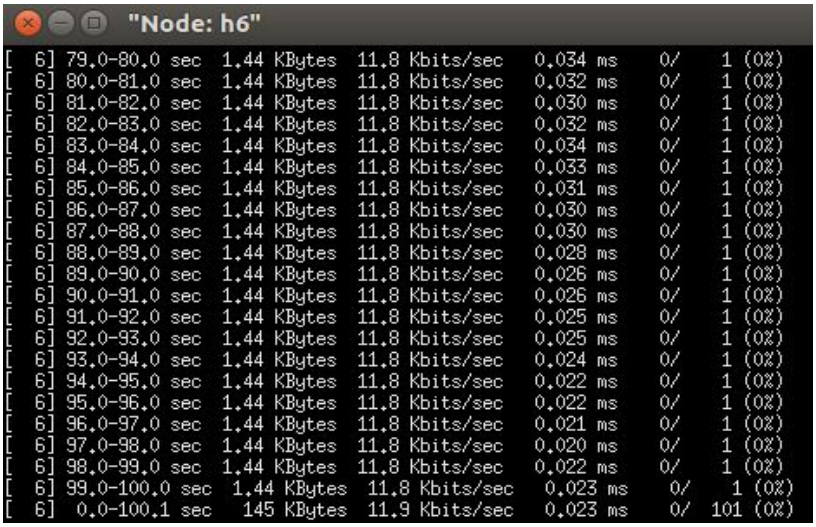


Figure 5.14 Result at h6

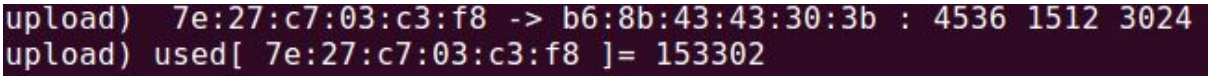


Figure 5.15 Limit is less than predefined value

7) Change the system time as shown in figure 5.16 and see if the h1 can send data again. The quota is reset and h1 can send the data packets again which is described in the figure 5.16 below.

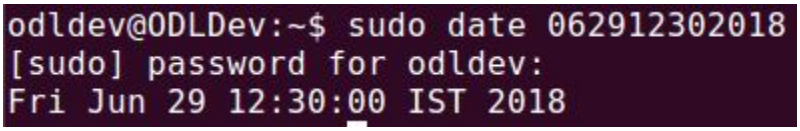


Figure 5.16 Change in system date

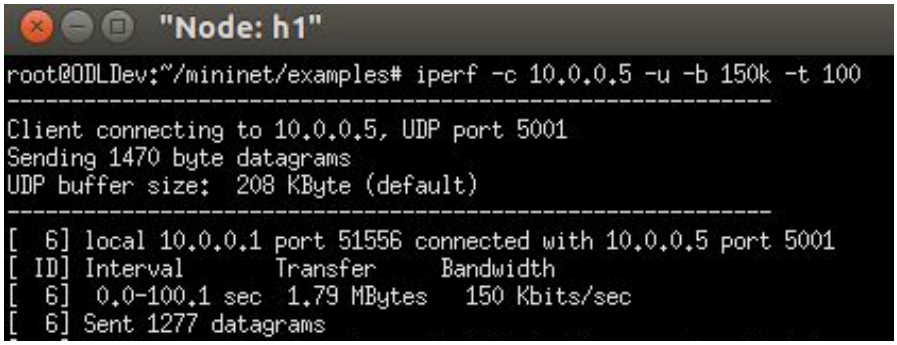


Figure 5.17 Result at h1

5.3.1 Comparison-graph

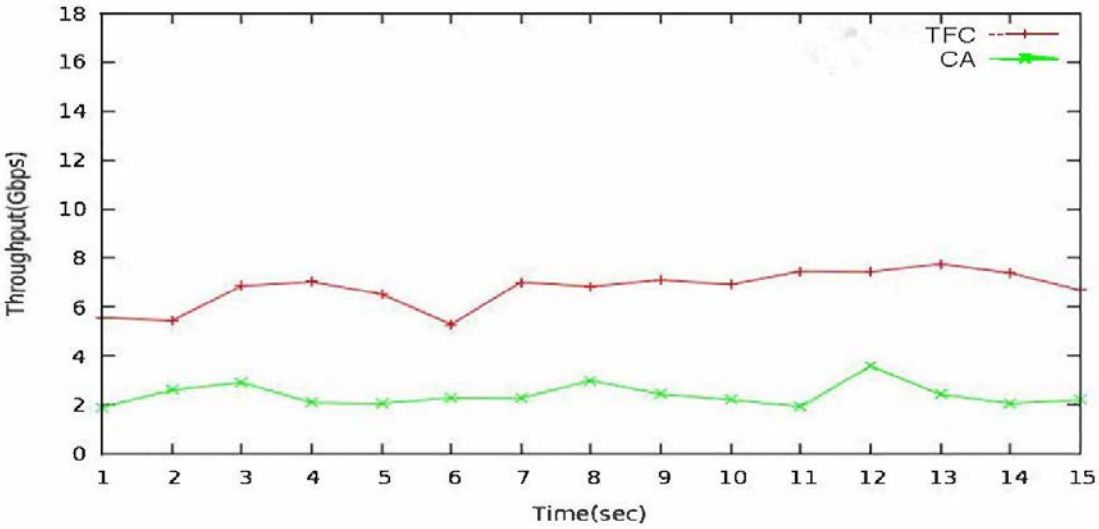


Figure 5.18 Throughput Comparison Graph of TFC and CA algorithm

From the above figure it can be concluded that TFC algorithm has higher throughput approx. 5.9 Gbps in comparison to the congestion avoidance having 3 Gbps of throughput, it means the more number of data a link can carry during the traffic flow from source to destination.

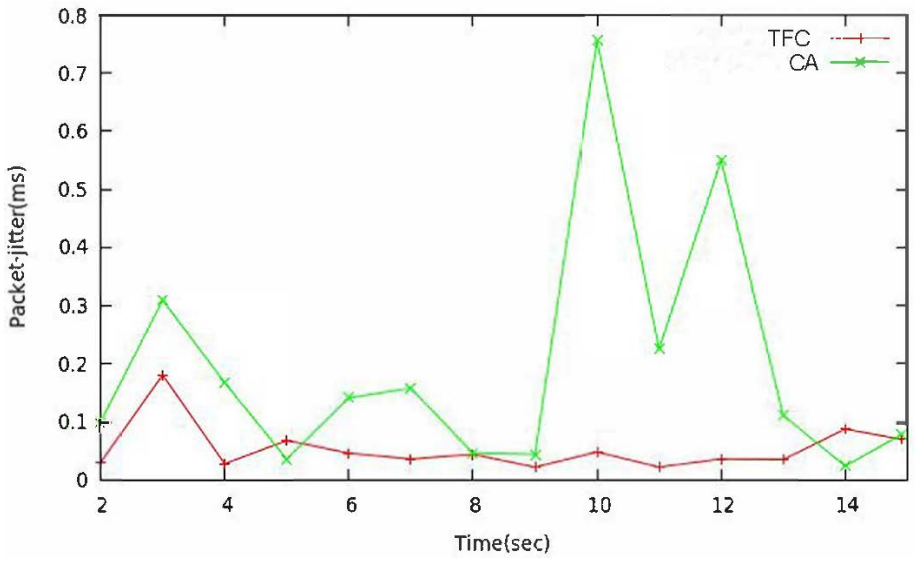


Figure 5.19 Jitter Comparison Graph of TFC and CA algorithm

From the above figure it can be concluded that Jitter is higher in case of congestion avoidance i.e. 0.79 ms (maximum) it means packets takes longer time in case of congestion avoidance algorithm.

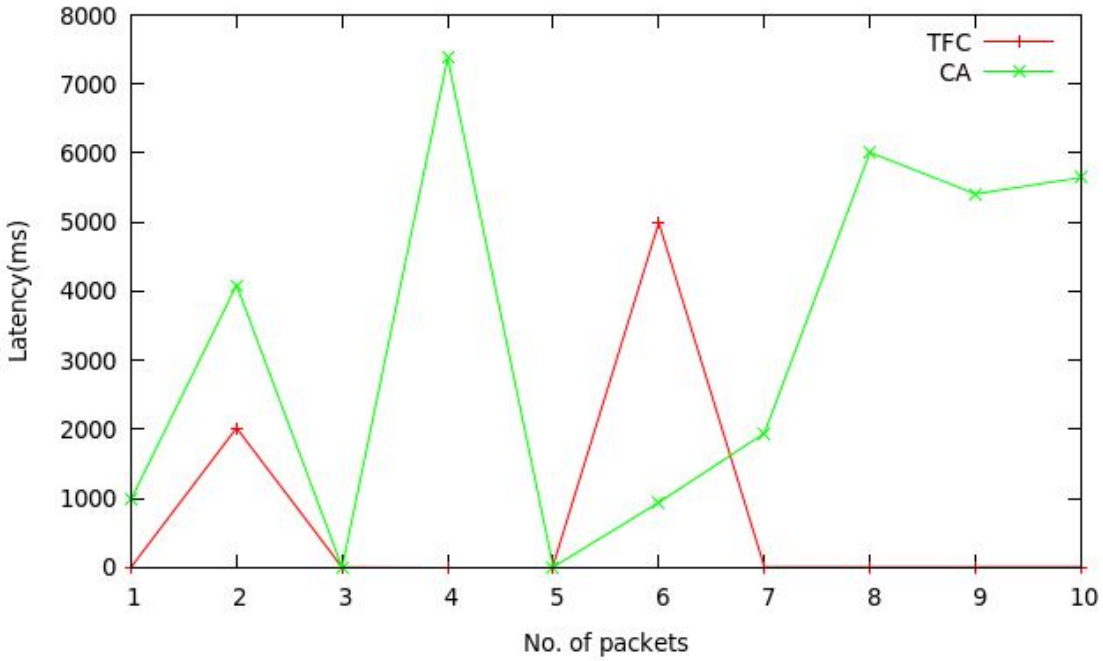


Figure 5.20 Latency Comparison Graph of TFC and CA algorithm

From the above figure it can be concluded that more number of packets can be travelled from source to destination in case of Traffic control algorithm with minimum latency i.e. 0.9 ms.

Chapter-6

Conclusion and Future Scope

This chapter concludes the thesis work with future scope

6.1 Conclusion

This thesis describes the already existing problem as well as solution above and successfully implements the problem by using Traffic Flow Control Algorithm.

The proposed framework named as traffic flow control algorithm , TFC is implemented with the use of shortest path algorithm bellman ford in the SDN network to control traffic and shown the comparison of TFC algorithm with the Congestion Avoidance algorithm in terms of throughput, latency and jitter and concluded that the link utilization of network is more in case of TFC algorithm because in this algorithm, there is a limit of 30000 bytes and if any source limit crossed then it will be blocked so as to prevent huge traffic in broadcast domain which was not in case of congestion avoidance algorithm.

6.2 Future Scope

In future, the network technology will be so smart with use of machine learning algorithms so that there will no need to define any quota or limit to restrict traffic.

The machine learning will automatically predict the traffic data at each switch and forward it so that no other traffic flow will affect the others.

References

[1] A. L. Valdivieso Caraguay, L. I. Barona Lopez and L. J. Garcia Villalba, "Evolution and Challenges of Software Defined Networking," in *IEEE Conference on SDN for Future Networks and Services (SDN4FNS)*, Trento, pp. 1-7, 2013.

[2] Y. Li and M. Chen, "Software-Defined Network Function Virtualization: A Survey," in *IEEE Access*, vol. 3, pp. 2542-2553, 2015.

[3] Nadeau, T. and Gray, K. "*SDN: software defined networks*." Beijing: O'Reilly, pp. 18-28, 2013.

[4] F. Hu, Q. Hao and K. Bao, "A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation," in *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2181-2206, Fourthquarter 2014.

[5] G. Tarnaras, E. Haleplidis and S. Denazis, "SDN and ForCES based optimal network topology discovery," in *1st IEEE Conference on Network Softwarization (NetSoft)*, London, pp. 1-6, 2015.

[6] Jacquenet, C., Bourdon, G. and BoucadaIr, M. "*Service automation and dynamic provisioning techniques in IP/MPLS environments*." Chichester, England: J. Wiley & Sons, p.346, 2008.

[7] H. Xu, X. Zong, J. Su and Y. Fu, "Formalization of SNMP messages using composite-elements based on extenics for software-defined networking," in *IEEE 9th International Conference on Communication Software and Networks (ICCSN)*, Guangzhou, pp. 989-992, 2017.

[8] S. U. Masruroh, A. Fiade, M. F. Iman and Amelia, "Performance evaluation of routing protocol RIPv2, OSPF, EIGRP with BGP," in *International Conference on Innovative and Creative Information Technology (ICITech)*, Salatiga , pp. 1-7, 2017.

[9] M. Jethanandani, “YANG, NETCONF, RESTCONF” in *Optical Fiber Communications Conference and Exhibition (OFC)*, Los Angeles, pp. 1-65, CA, 2017.

[10] Z. Shu *et al.*, “Traffic engineering in software-defined networking: Measurement and management,” *IEEE Access*, vol. 4, pp. 3246–3256, 2016.

[11] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, “Overview and Principles of Internet Traffic Engineering,” pp. 1–71, 2002.

[12] J. M. D. Awduche, J. Malcolm, J. Agogbua, M. O’Dell, I. R. “Requirements for traffic engineering over MPLS, and S. 1999. For Comments, RFC 2702, “Requirements for traffic engineering over MPLS, Internet Requests for Comments, RFC 2702, Sep. 1999.,” pp. 1–29, 1999.

[13] B. Fortz and M. Thorup, “Internet traffic engineering by optimizing OSPF weights,” in *IEEE Infocom*, vol. 2, pp. 519–528, 2000.

[14] G. Swallow, “MPLS advantages for traffic engineering,” in *IEEE Communications Magazine*, vol. 37, no. 12, pp. 54-57, Dec 1999.

[15] S. Song, J. Lee, K. Son, H. Jung and J. Lee, “A congestion avoidance algorithm in SDN environment,” in *International Conference on Information Networking (ICOIN)*, Kota Kinabalu, pp. 420-423, 2016.

[16] Seyeon Jeong, Doyoung Lee, Junemuk Choi, Jian Li and James Won-Ki Hong, “Application-aware Traffic Management for OpenFlow networks,” in *18th IEEE conference on Network Operations and Management Symposium* Kanazawa, pp. 1-5, 2016.

[17] C. Ren, S. Wang, J. Ren, X. Wang, T. Song and D. Zhang, “Enhancing Traffic Engineering Performance and Flow Manageability in Hybrid SDN,” *in IEEE Global Communications Conference (GLOBECOM)*, Washington, DC, pp. 1-7, 2016.

[18] C. P and D. B. S. Shylaja, “Intelligent Traffic Management in SDN,” in *International Journal of Engineering Research* , vol. 6, no. 05, pp. 1-4, 2017.

[19] M. Team, “Mininet Overview,” *Mininet Overview – Mininet*. [Online]. Available: http://mininet.org/overview/.

[20] V. GUEANT, “iPerf – The ultimate speed test tool for TCP, UDP and SCTPTest the limits of your network Internet neutrality test,” *iPerf.fr*. [Online]. Available: https://iperf.fr/.

[21] “Using the POX SDN controller,” *Open-Source Routing and Network Simulation*, 09-Jul-2015. [Online]. Available: http://www.brianlinkletter.com/using-the-pox-sdn-controller/.

[22] O.N.F., “Software-defined networking: The new norm for networks,” *ONF White Pap.*, vol. 2, pp. 2–6, 2012.

[23] J.W.Ki Hong, *SDN Evolution*..[Online], Available: https://slideplayer.com/slide/6204491/

Publications

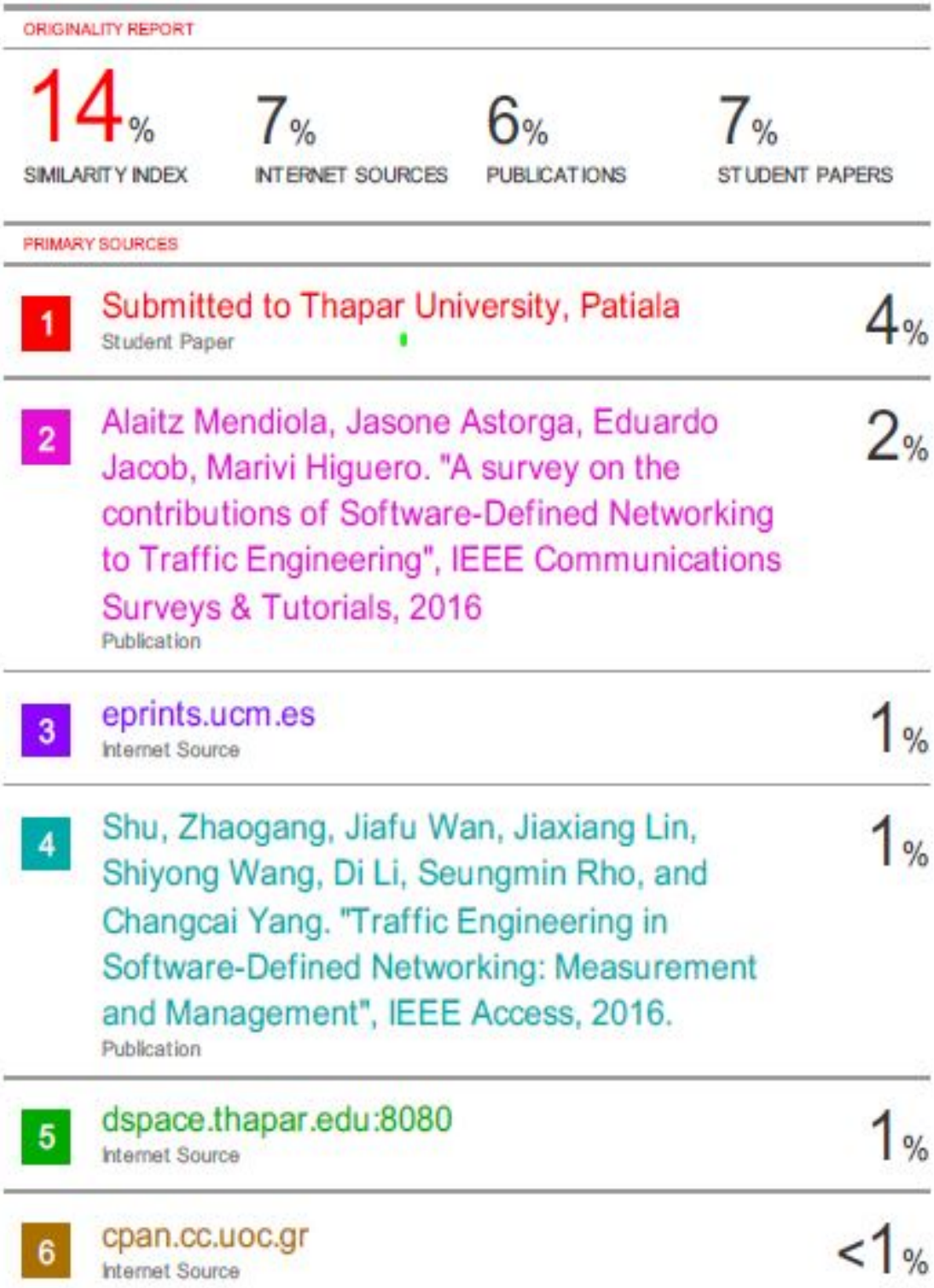
- Jatin Sharma and Dr. Rajesh Kumar, “Traffic Flow Control Algorithm in Software Defined Networking,” accepted in IJSRCSAMS journal, vol. 7, no. 4, pp. 1-5, 2018.

Video Presentation Link

- Presentation on Traffic Flow Control algorithm in SDN

  Link: https://youtu.be/4VDKvTRniO8

Plagrism Report



ORIGINALITY REPORT

14%	7%	6%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Thapar University, Patiala Student Paper	4%
2	Alaitz Mendiola, Jasone Astorga, Eduardo Jacob, Marivi Higuero. "A survey on the contributions of Software-Defined Networking to Traffic Engineering", IEEE Communications Surveys & Tutorials, 2016 Publication	2%
3	eprints.ucm.es Internet Source	1%
4	Shu, Zhaogang, Jiafu Wan, Jiaxiang Lin, Shiyong Wang, Di Li, Seungmin Rho, and Changcai Yang. "Traffic Engineering in Software-Defined Networking: Measurement and Management", IEEE Access, 2016. Publication	1%
5	dspace.thapar.edu:8080 Internet Source	1%
6	cpan.cc.uoc.gr Internet Source	<1%