An approach to Improve Scalability and Quality of Services in Software Define Networking

SUBMITTED BY
Raza Hussain
01-245171-022
MS (Telecommunication and Networks)

SUPERVISED BY
Dr. Kashif Naseer Qureshi

Department of Computer Science
Bahria University Islamabad Campus
Session 2017 -2018

Bahria University
Discovering Knowledge

Thesis Completion Certificate

Scholar's Name: Raza Hussain____________ Registration No, 01-245171-022_______

Program of Study: MS (Telecom & Networks)

____

Thesis Title: An approach to Improve Scalability and Quality of Services in Software Define Networking

It is to certify that the above student's thesis has been completed to my satisfaction and, to my belief, its standard is appropriate for submission of Evolution. I have also conducted plagiarism test of this thesis using HEC prescribed software and found similarity index at ___________ that is within the permissible limit set by HEC for the MS/MPhil degree thesis.

I have also found the thesis in format recognized by the BU for the M/MPhil thesis.

Principal Supervisor's Signature: [illegible]

Date: 12/02/2018 **Name:** Dr. Kashif Naseer Qureshi

Bahria University
Discovering Knowledge

MS-14A

Author's Declaration

I, Raza Hussain_ here by state that my MS thesis titled "An approach to Improve Scalability and Quality of Services in Software Define Networking" is my own work and has not been submitted previously by me for taking any degree from this university Bahria University.

At any time if my statement is found to be incorrect even after my Graduate the university has the right to withdraw/cancel my MS degree.

Name of scholar: Raza Hussain

Roll Num: 01-245171-022

Bahria University
Discovering Knowledge

MS-14B

Plagiarism Undertaking

I, solemnly declare that research work presented in the thesis titled "An approach to Improve Scalability and Quality of Services in Software Define Networking" is solely my research work with no significant contribution from any other person. Small contribution / help wherever taken has been duly acknowledged and that complete thesis has been written by me.

I understand the zero tolerance policy of the HEC and Bahria University towards plagiarism.
Therefore I as an Author of the above titled thesis declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred / cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even after award of MS degree, the university reserves the right to withdraw / revoke my MS degree and that HEC and the University has the right to publish my name on the HEC / University website on which names of students are placed who submitted plagiarized thesis.

Student / Author's Sign: [illegible]

Name of the Student: Raza Hussain

CERTIFICATE

We accept the work contained in this report as a confirmation to the required standard for the partial fulfillment of the degree of MS (TN).

	[illegible]
____________________	____________________
Head of Department	**Supervisor**
____________________	____________________
Internal Examiner	**External Examiner**

DECLARATION OF AUTHENTICATION

I, Raza Hussain (Enrollment No. 01-245171-022), solemnly declare that my research thesis report entitled “Methodology to Improve Scalability and Quality of Services in Software Define Networking” is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person. All the references and acquired help in this study have been acknowledged. I testify that the material produced in this research work, either in whole or part has not been obtained or used for any other degree at this or any other institution.

Signature……………………..

ACKNOWLEDGEMENT

Being thankful to ALLAH, to Whom belongs all the powers and grandeur, this research work is materialized in final shape.

I am extremely grateful to my supervisor, Dr. Kashif Naseer Qureshi, for the guidance, resolute support, valuable time, untiring efforts and confidence in me throughout the course of this thesis work.

I offer my best regards and prayers to all who supported me in any respect during the completion of this work. Last but not the least; I feel bound to pay homage to all those people who believed in me, gave me confidence and without whom I could never have achieved my goal; my family, especially my Parents. I thank them all for rendering their constant prayers and persistent support for me. At the end I would like to thank my friend who helped me throughout the completion of my thesis work.

May Allah bless them all with eternal happiness!

DEDICATIONS

To My Father, Mother, Family and Friends

.

Abstract

Software Defined Networking (SDN) is platform that support network application more efficiently. SDN is successful because it decoupled the control plane from its data plane. Detachment of these two-planes improved dynamic configuration, controller programmability, centralized and decentralized network. In this study, Distributed SDN approach is used to avoid scalability and robustness issue. The SDN controller logically centralized but it works on multiple nodes that physically distributed. Distributed nodes communicate with each other and improve the network performance to assist data flow. When flow arrives through end devices to any data plane in Distributed SDN, our approach will support the data plane to increase controller response time. In addition, controller will check, whether data needs to prioritize flow to send locally or globally with the help of Elephant or Mice flow. Proposed approach used to reduce the controller work load, response time and manage Quality of Services (QoS) between controllers.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

SDN	Software Define Network
TCP/IP	Transmission Control Protocol/Internet Protocol
CIDC	Communication Interface for Distributed control
SDNi	Software Define Network inter
BGP	Border Gateway Protocol
eBGP	External Border Gateway Protocol
NFV	Network Functions Virtualization
QoS	Quality of services
NFC	Near Field Communication
CPU	Central Processing Unit
MMPP	Markov Modulated Poisson Process
CSP	Controller selection problem
VNi	Visual Networking Index
EIGRP	Enhanced Interior Gateway Routing Protocol
OSPF	Open Shortest Path Firs
AS	Autonomous System

Chapter 1
Introduction

1.1 Overview

In network paradigm, Software Define Network (SDN) is immerging approach. Which have gained great interest in computer science domains. SDN has ability to drive network operations independently with the help of detaching control and forwarding plane. On one side, flow forwarding device turn out to be extremely well organized and programmable such as data plane. On the other hand, intelligent forwarding machine act like single entity to perform and control all network traffic such as control plane. SDN is more capable, easily deployable and provide maintenance to application as compared to previous network architectures. As a broad view, SDN is one of the architectures that strengthened the global view and consistency in network polices. Its infrastructure that brought evaluation in networking. Centralized SDN and Decentralized SDN are different with their own features. SDN infrastructure have great impact on networking applications. Most specifically load balancing, quality of services, strategies, requirement, transmittance, processing and storage [1, 2].

1.2 Introduction to Software Define Networking (SDN)

In network organizations management of data centers is main issue. System monitoring, and management of network become challenge. End devices are connected with each other and generate massive amount of data. Also, video conferencing, online streaming and online gaming make great impact on network capacity. Underlying network require flexible capabilities and parallel processing for mega data set handling [3]. Various devices generating massive amount of data. But handling massive amount of data is huge challenge for traditional network. Distributed computing requires for massive growing network base application. Controlling network devices and individual processing SDN is best approach for software base application.

Internet has combinations of data units and these units need their own processing to gather information. Because of need to process and acquire information is require to use data computing and storage process. To avoid delay or error in flow traffic in network, large data center requires suitable network configuration. Traditional network architecture is not efficient to provide all facilities or to avoid delay or mange error control in traffic [4]. So, the main reason for using SDN is more suitable to handle issues to control data configuration in data centers. SDN allows logically centralized performance to increased efficiency and reliability in network allocation and load managing across the switch to control selection methodology.

Management of forwarding traffic flow table, polices constraint related to any link discovery, monitoring the network statistics are main responsibilities of control plane in SDN. Combination between two planes can be done by southbound application program interface and controller to application layer interaction can be done by northbound application program interface as shown in Figure 1.1. Network state monitoring deals with two main aspects, real time traffic and abnormal node connection. Real time traffic involves with adjustment of network statistics, well-timed detection of network congestion. When the controller controls more than one network domains, it should notice abnormal statistics. These abnormal node helps to adjust network topology structure like abnormal node connection. The OpenFlow protocol controller helps to data plane to maintain forwarding table [5]. Main issue with single controller in network is sometime controller may not have all flow information to route in network. Because when topology change in SDN, link discovery protocol must be modified.

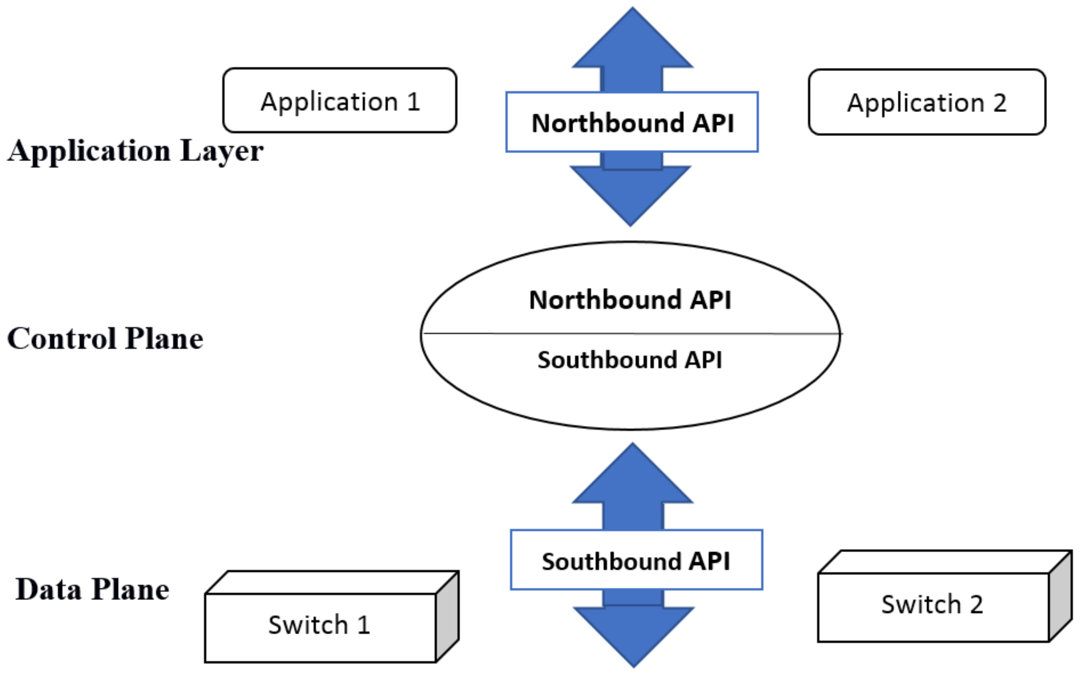


Figure 1.1: Basic diagram of software define networking

Underlying topology always dynamic and never be fixed. Method proposed to add some threshold [18] limit on switch to manage controller response time. Without threshold switch data plane can reduce controller outcome. But reactively coming flow rate is unpredictable. Data out rate of switch differs because reactively traffic patterns are hard to predict. Sometime large data flow increases by limit that can affect the controller performance. Topology of network can be change in case of large data failure or to provide high availability of resource. Fixed topology can be helpful but not viable completely.

In network organizations management of data centers is main issue. Network administrator main goal is how to achieve better approach to facilitate data centers. SDN is best approach to fulfill their demands. SDN manage large data flow with the help of its control and data plane. SDN can benefit end devices data to manage its structure and unstructured data like online streaming, online gaming and social media etc. All end devices data features can facilitate on demand under SDN. It has ability to perform intelligent networking to manage end devices data more customizable and scalable. Our propose method is topology independent framework to enhance the control layer with the objective of calculating the optimal number of controllers to reduce the workload and improve the quality of services.

Resource integrate according to the control plane because traffic flow and size behavior change regularly. Controller response time and delay is dynamically changed in load condition. Main reason behind this different traffic model has difficult to fulfill the quality of service requirement. To achieve minimization in traffic flow controller plane, response time should be minimum [19]. Controller main activity is always required flow resource and response time between traffic flow which should be minimum. Because of this, we need many controllers to facilitate network requirement. Many controllers will increase cast but they will facilitate flow more efficiently.

High traffic and their maintenance required addition and deletion of controller in network. Many works have been done related to controller capacity but the controller response time is more important to transfer flow end to end. [20, 21] as explained before in two main points that end to end delay is more important factor. We cannot compromise with performance but cast and resource etc., are affordable. When data flow arrives on controller its basic need is to traffic the route on its time. Controller response time should be minimizing to perform quality of service as focused on controller capacity.

1.3 Centralized SDN architecture

Looking first at a centralized SDN architecture -- the model that standards groups like the Open Networking Foundation (ONF) support -- the key element is the connecting technology that communicates central control decisions to devices. OpenFlow has become the official protocol to use in a centralized SDN model to make high-level routing decisions. As a result, the creation of central-control SDN must be based on selecting devices and control software that support the OpenFlow standard. With OpenFlow, there is good and bad news. The good news is that virtually all of the major switch/router vendors have announced OpenFlow support, even though there are several versions of the OpenFlow standard and vendors may not have released software for the latest version. A number of OpenFlow-compatible controllers can provide central control for a community of devices, and many OpenFlow-based SDN trials and deployments have been conducted using these tools.

1.4 Distributed SDN architecture

The distributed SDN model, evolution is the goal. Here the focus of SDN development is the control software that the centralized model is addressing in only a limited way. Distributed SDN presumes that switches and routers are already deployed throughout the network. It is also a given that these devices already support most or all of the connecting technology needed, in the form of protocols like MPLS, GRE and BGP. The goal is to expose the traffic and connectivity management capabilities of the current networks to a higher software layer, which would then frame these capabilities as "virtual network services" to the cloud or to applications. What matters is that northbound APIs or interfaces allow software, including cloud stack software, to control network services.

In the area of connecting technology, the distributed SDN model has a very different requirement set. Because it doesn't centralize routing decisions as its competing model does, the distributed model doesn't need OpenFlow, though it may be supported at some point. What it needs, however, is a practical way of gathering a considerable amount of status and performance information from the network, which means gathering it across all of the protocol layers, device types and vendors involved. Without this data, it is impossible to ensure that the virtual network services created in a distributed SDN model conform to software needs because network conditions can't be accurately determined. As a result, monitoring technology is the key to the success of the distributed SDN model.

SDN has been collaborated with different computer domains to facilitate the network domains. Main goal behind our work is, to provide QoS in network. When data arrive in any switch, threshold will use to provide limited flow rate to controller to minimize controller response time. Controller will prioritize the flow with help of Mice or Elephant algorithms. Controller will decide whether to deal with flow locally or peer controller. Our main purpose is to minimize the controller response time and achieve QoS in network as shown in Figure 1.2.

Single controller in network has limited scalability and computing power. In large network computing operations and traffic flow are very high. Single controller faces a number of concerns like scalability, robustness, bottleneck etc. Researchers have been presented many approaches about controller functionality with end devices data towards controller placement problem, but they are not satisfactory. In traditional based network processes were long and static for link cast just like ISIS routing protocol. Because of congested data delivery become biggest problem until the cost of link changed in the network. Whereas, casts of links dynamically changed to understand network changes in SDN. There should be intelligent machoism for routing that used to implement in SDN and on the other hand, routing technique should be improved to resolve these problems. Divide single SDN into distributed network domains to achieve scalability.

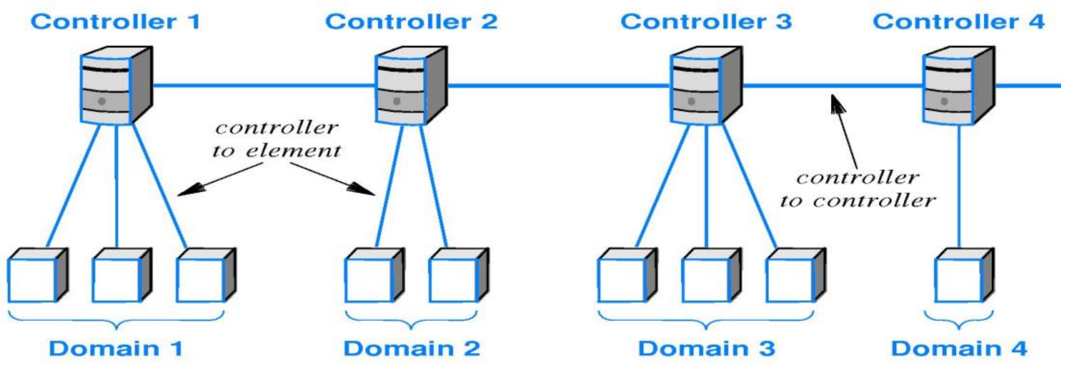


Figure 1.2: Diagram of Distributed software define networking

Our system used to demand resource in order to present the analysis of response time. Controller response should be improving with high rate of services rather than deploying to many controllers in network, that is the main part of our analysis. So we can minimize the response time and time flow setup. All controllers are connected to each other and responsible to manage their own domains. Moreover, for distributed architecture flow managed by a single controller to control flow these issues, two approaches have been presented, logically distributed and logically centralized. In logical centralized method controller used to collaborate with the view of network and database as well. On the other hand, multiple domains over high distributed network logically distributed method is used for network. In this approach, network topology whole view presents the global view of network and share specific information between controllers. To update global state. Neighbor controller should communicate with each other and threshold on switch to manage controller response time. In which we are proposing a model that don't have any tool to simulate properly. For simulation we are considering MATLAB, Mininet and GNS3 for each step. We are not discussing much about elephant or mice flow prioritization but presenting valid literature. We are not presenting any protocol but combining different methodology with the help of literature that support our Model.

1.5 Problem Background

Underlying topology always dynamic it never be fixed. Methodology proposed to add some threshold limit on switch to manage controller response time [5]. Without threshold switch data plane can reduce controller outcome. But reactively coming flow rate is unpredictable. Data out rate of switch differs because reactively traffic patterns are hard to predict. Sometime large data flow increases the limit of threshold that can affect the controller performance. Topology of network can be change in case of large data failure or to provide high availability of resource. Fixed topology can be helpful but not viable completely.

1.6 Problem Statement

Resource integrate accordingly control plane because traffic flow and size behavior change regularly. Controller response time and delay is dynamically change in load condition. Main reason is different traffic model has difficult to fulfil the quality of service requirement. To achieve minimization in traffic flow controller plane response time should be minimum [6]. Controller main activity is always providing required flow resource and response time between traffic flows should be minimum. Because of this we need many controllers to facilitate any network requirement.

1.7 Research Question

We have mention main points related to our problem domain after detailed and careful surveillance.

- How to handle network size and changes in topology?
- How to minimize controller response time to achieve quality of services in network?

1.8 Research Objective

Single controller in network has limited scalability and computing power. In large network computing operations and traffic flow are very high. Single controller faces a number of concerns like scalability, robustness, bottleneck etc. Researchers have been presented many approaches about controller functionality with end devices data towards controller placement problem, but they are not satisfactory.

- Divide single SDN into distributed network domains to achieve scalability.
- All controllers are connected to each other and responsible to manage their own domains. To update global state.
- Comparison between SDN controller with threshold.

Neighbor controller should communicate with each other and threshold on switch to manage controller response time.

Chapter 2
Related Work

2.1 Overview

In previous work controller selection in distributed SDN has not explained very well. Our aim is not to determine the optimal placement of controllers in the network, but to motivate the controller selection problem. Many literatures have been presented about controller placement problem.

2.2 Literature Review

They have presented heuristic base algorithm for shifting the load of controller dynamically, through switch to controller and between data plane for average flow request. But author didn’t explain about statistics, rules and flow between controllers [6]. They have proposed general approach for distributed controllers for achieving scalability. But they didn’t explain the challenges related to controller placement, number of optimal controllers and distribution of load between controller [7].

They showed a data collection system for vehicle networks. They focused on the collection and delivery, when there are a large number of data. Based on a mixture of cooperation between mobile networks and ad hoc networks. Bottleneck problem was main concerns in traditional networks. Because when data arrive in traditional network, it is difficult to address massive data transmission [8]. They highlighted some important points. These points have limited knowledge related to SDN and large volume. SDN can provide QoS when large volume of data flow arrives in network. They also have monitored the network packet. Batter way to achieve QoS for large data application and assert analytics [9].

Seer explained two major domains data analytics and SDN characteristics for introducing the central network system. They also provide the decisions base knowledge from third party. Analytical module combines with SDN control plane together data from network. Seer facilitates all network information with help of distributed message gathering system called Apache Kafta and distributed Database with Apache Cassandra. Knowledge base decision facilitates data with the help of data analytics [10]. Application layer centralized the logic for call interface provide between control networks. Resource allocation turned to provide by SDN in network. Network resource become more delegate capable and increase the network flexibility [11].

They proposed controller placement problem in SDN with the help of mathematical mode. Model used to determine type of controller, location of controller and optimal number of controllers for specific set of switches. They have also presented the method of minimization of elements cast for network [3]. More work has been done to optimize the model and used for bound algorithm as optimizer. But it causes the accuracy and consistency of model due to adding multiple factors.

They have proposed social TV for data analytics with the help of SDN platform. Main goal behind is to find TV program that accept the audience requirement with distributed domains. They used Hadoop and cluster statistics execution. They have presented the technique, in which the combined SDN and big data with social TV program. Microblog data-based program used to collect people thoughts, opinion and perception. But in this model, it is difficult to distinguish structure and unstructured data [12]. Detection algorithms in uncontrolled domain, in which one willing to provide anomaly explicit support. DNA methodology used to provide data network to controller and highlights the statistics generation. When communication takes place in controller, it required high bandwidth and it also case overhead controller for each process. Data resource configures to provide local network to support data analytics, in which they have used agent-based approach [13].

They proposed a scheme to reduce transmission delay for smart devices. Smart devices generate huge amount of data and they faced the reduction of dimensionality problem. Scheme used reconstruction error minimization with the help of Frobenius norm. But they didn’t explain how they will handle large data from single SDN controller. Data will cause, when single controller become overload with large flow. Bottleneck and scalability issues are not highlighting in their study [4]. They have presented approach in traffic engineering to facilitate SDN. Main goal behind their work for network optimization and configuration in SDN controllers to provide quality of services. They have commuted optical circuits to facilitate network dynamically and configurable. But they didn't explain data scheme to support SDN for network application management [14].

Big data in computing domain based on large scale is MapReduce. TCP/IP based approach is used to provide map phase for reducers. Main issue is, when large data arrived, high speed case problem to manage optimization and speed cannot accommodate in this condition [15]. Pythia model proposed for MapReduce to enable communication for prediction the system application. It also allocated the optimized bandwidth for network application and configuration. But application aware and MapReduce approach have been designed multiple times for SDN system [16]. For services management, they proposed AWESoME prototype to facilitate SDN. Originating services explained in detail for concept of DNS and blog domains for initial packet association. Grained control defined for marginal load and accurate flow enabling. So, the SDN data plane and control plane work practically [17].

TABLE 2.1: Literature review on SDN

Ref/year	**Scheme**	**SDN Architecture**	**Objective**	**Limitations**
[5]/2015	Ant Colony Optimization	Optimized control plane	Optimized control plane, Cluster security	Distributed SDN controller cluster scalability and security not explained
[6]/2016	Distributed controllers	Floodlight SDN controller	distributed controllers for achieving scalability	Didn't explain Challenges of Distributed controllers
[7]/2016	Preco	Software Defined Vehicular Networks	Predictive routing for Ad hoc relay mode multi-hop	Repeated flow traffic blocking when large data evolve
[8]/2016	Tensor Based Model	T-SDN	Network traffic prediction and QoS provisioning	Minimal work on exploiting the properties of SDN
[9]/2016	Seer	ONOS SDN	SDN & Big data framework provide intelligence in network	Holistic method to deploy every module with scale up

[10]/2017	AAN-SDN framework	Centralized SDN	(SDN) and (AAN) network underlying function, forwarding logics with M/R	Large scale deployment issue in test case in big data analytics
[4]/2015	Mathematical model	Controller Placement Problem	Determine type of controller, location of controller and optimal number of controllers	Causes the accuracy and consistency of model
[11]/2015	SDN Social TV analytics	SDN	Micro blog data extracts the knowledge and public perception in TV program	Challenges for unstructured or structured data
[12]/2015	DNA	SDN	Data source for dynamic and reconfigure analytics tasks	Not fully-distributed event collection, scalable feature collection and management
[13]/2017	Clustered distributed SDN	Distributed controller	To improve network performance, its scalability and its reliability	Throughput rate, for flow setup when No, of cluster are large
[14]/2012	Data Center Networking	Centralized SDN	Optimize network utilization	Flow-level traffic engineering
[15]/2015	BASS	Cluster centrally controlled SDN	Bandwidth-Aware Scheduling with SDN in Hadoop	Huge amount of data still takes a lot of time to transfer.
[16]/2014	Pythia	SDN	Optimize bandwidth allocation	lacks of clear and comprehensive SDN design with MapReduce

[17]/2017	AWESoME	SDN	To facilitate web services to important traffic, prioritize and identify	Security for application, service accounting

2.1.1 Traffic Engineering For SDN

Network status can be changed by dynamic behavior of controller in SDN. In traditional based network processes were long and static for link cast just like ISIS routing protocol. Because of congested data delivery become biggest problem until the cost of link changed in the network. Whereas, casts of links dynamically changed to understand network changes in SDN. There should be intelligent machoism for routing that used to implement in SDN and on the other hand, routing technique should be improved to resolve these problems. Protocol can be changed as per network requirement dynamically to improve the quality of surface, congestion, avoid pitch and resource utilization. SDN is modern network architecture in which various research community presented the many techniques for traffic engineering to resolve dynamic changing in the network.

They proposed the method that is used for Mice and elephant flow detection accurately. Also, for scheduling take arrangement by itself according to different condition. Whereas, mice flow used database for path updating to make flow decision and elephant flow compensated by k edge disjoint to compute path through k. Algorithm used to study and satisfy the different flow demand for heuristic information optimization and for this ant algorithm presented [18].

The proposed flow scheduling traffic scheme, when congestion occur for redistribute SDN flow in network. They presented the first-time flow scheduling detection method for elephant flow aggregation. Their technique used to improve QoS overall and utilization of links for variance minimization with help of new optimization model in TE [19] as shown in Figure 2.1. They proposed a mutual methodology for flow that can live long for promoting its packets. They used both architecture traditional and SDN as well. These techniques also called heavy hitters or elephant flow and mange maintenance short lived aggregation. Elephant flow used for the SDN flow detection and each control plane individually take care its rule only to maintain elephant flow [20].

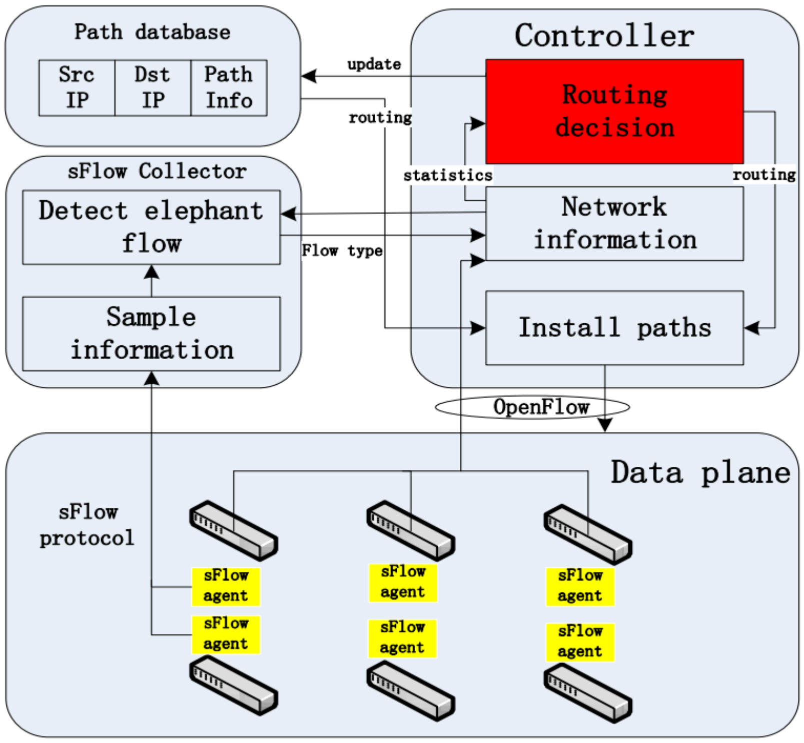


Figure 2.1: Framework of the flow scheduling system

They presented the Deveflow method that is used to enhance the performance and scalability of network. Main goal is to obtain centralized view and record the all flow in forward plane. This method is used to minimize the data plane and controller interaction. So switch have access to choose flow without requesting decision from controller again (e.g. elephant flow) [21]. But it need decision from controller plane on higher level application. Mahout [22] proposed the scheme in which data can be large amount but with help of elephant flow detection it is easy to control flow timely and significantly. There are lots approach for elephant detection like traffic statistics parodic polling. While, this technique is better for lower processing method to reduce overhead and also detect faster elephant flow. But it needs end hot notification.

Hedera [23] presented the technique for data center for better utilization of bandwidth. It's used to detect the elephant flow in the edge of data plane. They used periodic polling for implementation. For large flow detection that gather statistics to check more easy flow in five second. Because this method used for bandwidth improvement and with periodic polling over heading and resource utilization became higher.

TABLE 2.2: Traffic Engineering for SDN Scheduling algorithm

Ref/year	**Scheme**	**Scheduling algorithm**	**Objective**	**Limitations**
[18]/ 2017	Congestion-aware traffic scheduling algorithm (CATS)	Aggregated elephant flow sharing congested links and its detection algorithm	Traffic scheduling algorithm use to eliminate network congestion and to improve QoS	link utilization needs more improvement
[19]/2017	Ant colony optimization algorithm (ACO)	Elephant and mice flow algorithm	Eliminate the conflicts between elephant and mice flows and schedule traffic effectively	Cannot satisfy the demands for different flows
[20]/2016	Parametric Minimum Cross Entropy (PMCE) algorithm	Elephant and mice flow algorithm	Mechanism reduce the overhead and improve scalability of control plane	Controller only installs individual rules for elephant flows
[21]/ 2011	DevoFlow	Elephant algorithm	Detects the elephant-flows at the edge switches, if threshold is met, i.e. 1–10 MB, it marks the flow as elephant flow.	Need higher level decisions from the control plane
[22]/2011	Mahout	Elephant flow detection algorithm elephant flow	Detect elephant flows and signals the network controller using an in-band mechanism	Requires modification of the end-hosts

		detection algorithm		
[23]/2010	Hedera	Global First Fit and Simulated Annealing	Dynamic flow scheduling system for multi-stage switch	High resource utilization and overhead

2.1.2 Controller-to-Controller Communication

The design of controller is based on controlling the traffic of network. Logical distributed SDN was presented to increase the performance of network that are multi domain like WAN. Moreover, for distributed architecture flow managed by a single controller to control flow these issues, two approaches have been presented, logically distributed and logically centralized. In logical centralized method controller used to collaborate with the view of network and database as well. On the other hand, multiple domains over high distributed network logically distributed method is used for network. In this approach, network topology whole view presents the global view of network and share specific information between controllers. Furthermore, every controller is responsible for its own domain in distributed SDN.

They have presented the network task variation with performance. Their model consists of distributed controllers SDN. Moreover, to support network intelligence for each controller flow is managed for information and statistics. While, they also presented the method that support latest functionality and features in SDN controller module [24] as shown in Figure 2.2. The proposed scheme used to divide the SDN controller to exchange and synchronize multiple services with notification called Communication Interface for Distributed control plane (CIDC) [25]. But they didn’t present the approach for traffic engineering in the CIDC.

They have presented the algorithm for placement of controller that is based on reliability and different domains for controller. It’s a method used for multiple SDN controller domains which is used to divide the large network. Main purpose is that each controller reduces its partitioning for reliability for SDN [26]. On the other hand, more information is required to handle security risk as compared to traditional network related to centralized

SDN. They proposed WB-Bridge method for peer and cooperate mechanism for administrator domain of SDN specifically. Its method to share information between different network domain but not related to routing protocol. Their main task is evolved information view for announcing domain to provide inter domain technique for communication with help of various network information [27].

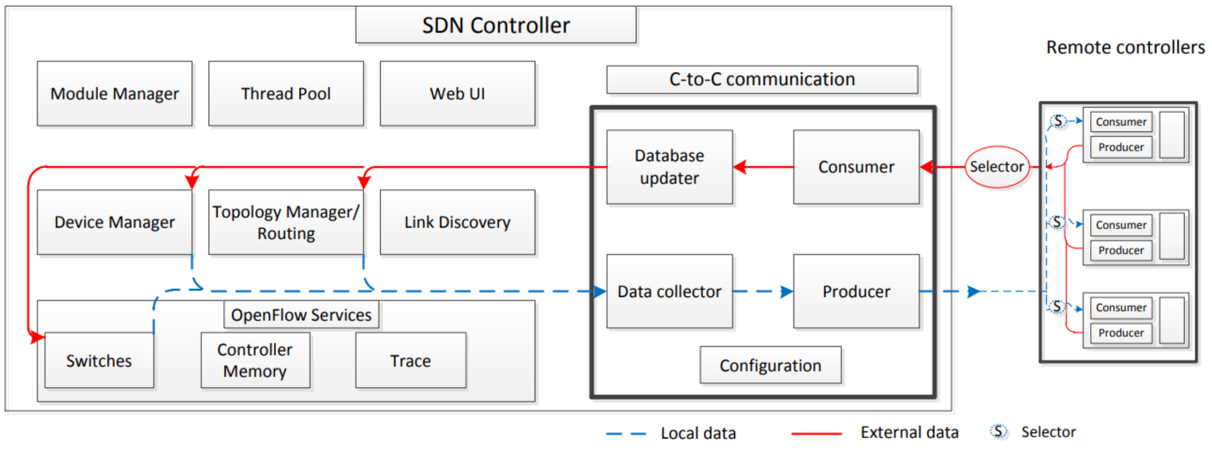


Figure 2.2: Architecture of controller-to-controller communication interface

Phemius et al. [28] presented the framework that SDN controller maintain network traffic of different domains. Whereas, proposed scheme is different from others because it maintains and differentiate intra and inter domain of flow information. In which inter domain helps heterogenous inter domain like SATCOM to introduce controller to the switch form another controller to its neighbor if any controller become down. SDNi [29] presented the technique that provide reliable information to coordinate each flow to setup in multiple domain. Furthermore, it is used to create more dependable and scalable distributed platform for distributed SDN. So, this protocol managed to interoperable and orchestrated network flow. Its control the diversity of platform to increase the leverage to interoperability to reduce the fault tolerance in software. On the other hand, system robustness increase diversity by reducing the common fault of probability. BGP is used to route in network and its job is share routing news from one autonomous system to other. That is reason behind its popularity that's varieties BGP protocol and it can easily modify for inter-SDN controller to controller message transformation, which is the best way for SDN network domains that have control and access of policies, QoS and other parameters concluded the SDN control plane. BGP has subsequent structures that needed for east-west interface of SDN. Messages from BGP can carry reachability and capability information as of in message presentation. BGP is feasible and standard protocol for peer to peer data exchanged. BGP meets the situations in current legacy-based resolutions such standard eBGP [30].

TABLE 2.3: Controller-to-Controller Communication for SDN

Ref/year	**Scheme**	**SDN Controller**	**Objective**	**Limitations**
[24]/ 2017	C-to-C	Floodlight	Dedicated interface that provides several modes to exchange information between distributed controllers	Real testbed required
[25]/2016	CIDC	Onedayight	Allows synchronization, exchange of notifications, services between multiple distributed SDN Controllers	Limited knowledge about traffic engineering
[26]/2015	Zebra	Floodlight, ryu and pox	Goal is to resolve communication problems between different controllers	Still at an primary stage
[27]/ 2015	WE-Bridge	Floodlight	Exchange basic network information between different domains	Features of inter-domain cannot be achieved inter-domain routing and end-to-end QoS routing
[28]/ 2014	DISCO	Floodlight	Manages its own network domain and communicates with other controllers to provide end-to-end	Resilient and recovery mechanisms not mention

[29]/2012	SDNi	OpenDaylight	Coordinate flow setup and exchange reachability information across multiple domains	Difficult to implement
[30]/2012	BGP	North-bound APIs	Inter-SDN controller communication	limitation to the number of switches and hosts that an SDN controller can manage

2.1.3 Queuing theory

We explain system modeling to improve the performance of SDN controller. Our system used to demand resource in order to present the analysis of response time. Controller response should be improving with high rate of services rather than deploying to many controllers in network, that is the main part of our analysis. So, we can minimize the response time and time flow setup. To improve the quality of services controller response time should be minimize this is our main goal. Moreover, resource and capabilities are also wasted if workload is very less and Qos also effected by high workload. Whereas, if we try to use system below its capacity, there should be some penalty.

They proposed the scheme for SDN controller for distributed decision. They presented the solution for flow balancing to investigate the benefit and feasibility. Also, dependences should be considered for each network like cast simultaneously, performance and resource [31]. However, priority flow does not consider in this scheme. They proposed frequency change of network traffic that is dynamic nature. They presented that due to change of network pattern, controller setup time for reactive flow also change. That is why controller selection strategy should be mapped application rather than controller placement strategy. So that is bounded for requirement of QoS process can be application in it [32]. But there is no model for controller selection to apply multiple incorporating parameters for example, number of services, delay and CPU utilization etc.

They have proposed the model that combined the NFV and SDN architecture. M/M/1 model used to drive NFV packet delay for analysis. In which we have presented the combination of SDN architecture and NFV. In real environment M/M/1 model drive to use

NFV packet delay. NFC is for more flexible to serve NFV to define instance loaded lighted [33]. Miao et al. [34] used multimedia network traffic for realistic nature and also consider Markov Modulated Poisson Process (MMPP) for packet arrivals to burst the network model. Furthermore, they also proposed two types of queues, low priority and high priority type queues in data plane. This technique used to solve high priority type queue portable with the help of MMPP/M/I/K and MMPP/M/I and also low priority type respectively.

Beigi-Mohammadi et al. [35] proposed a methodology about infrastructure application aware model for scalability and efficiency. Whereas, for validity to his work author presented the measurement of its testbed to verify his model, while, for cloud testbed author identify the scalability with bottleneck issues. But author didn't explain the tablespace for flow scalability. They presented a model [36] for switch capability that represent finite capacity and priority queues. In this model when high priority type queue has no packet than lower priority queue used to performed. It is based on non-preemptive priority queue. Moreover, for openflow switch these priority queues structure are more valuable. Furthermore, switch used queue model of finite capacity but queuing model is limited to produce specific queuing solution that are difficult to analyses significant as shown in Figure 2.3.

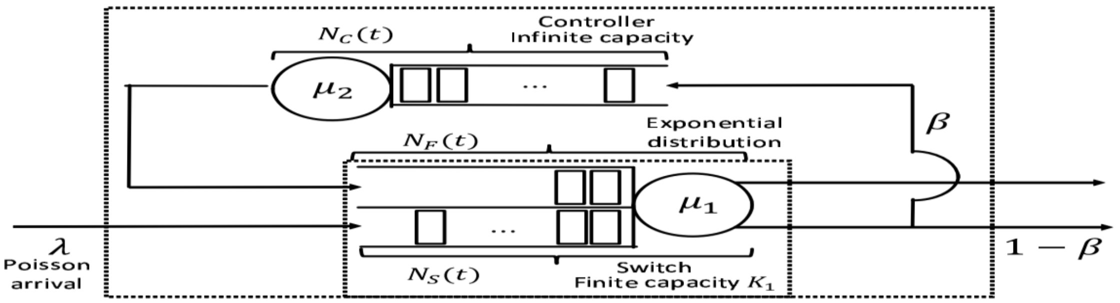


Figure 2.3: Queueing model of simple network with an OpenFlow switch connected to a single controller

TABLE 2.4: System modeling with Queuing theory for SDN

Ref/year	**Methodology**	**SDN and Technique**	**Objective**	**Limitations**

[31]/ 2016	Queuing theory, M/M/1 and M/M/m	distributed	Delivers flow balancing (with definite QoS) of SDN controllers	Implementation of Controller and switch not define
[32]/2017	M/M/1 and M/M/m	Control plane performance	Primary switches limited to certain bound so QoS can be guaranteed	Prototype model not executed and demonstrated
[33]/2017	M/M/1 queuing model	SDN+NFV	Objective to carry about analytical modeling of NFV C and NFV AC	NFV packets to reduce packet delay in VNF
[34]/ 2016	MMPP/M/1	SDN and (MMPP)	Capturing the traffic characteristics of multimedia applications	Ave. packet delay
[35]/2016	M/M/1 and M/M/1/S queue systems	SDN and SDI	Model captures details while maintaining tractability and extendibility	Model and evaluation not define for SDI
[36]/2016	Continuous-time Markov chai	OpenFlow–based SDN	A network comprising a single controller with multiple switches	Not includes modelling the performance of various controller architectures

2.2 Discussion

In network organizations management of data centers are main concern. Network administrator has main goal to achieve better approach to facilitate data centers. SDN is one of the best approach to fulfill their demands. SDN can benefit end devices data to manage its structure and unstructured data like online streaming, online gaming and social media etc. All end devices data features can facilitate on demand under SDN [6]. It has ability to perform intelligent networking to manage devices data more customizable and scalable. Our method is to introduce topology independent framework to enhance the control plane with the objective of manipulative the best number of controllers to decrease the capacity and improve the quality of services.

SDN is the extensive concentration in networks, fewer schemes have available and popular to the existing domains. This work is initial effort to travel the cravings between control plane load (or flow organization), many resources, and functioning expense. We talked QoS properties provisioning expenses minimization problematic. For avoiding important blockage at the centralized SDN controller [7], to improve scalability, placement of distributed control plane has been planned. That's the best and general method for attain network scalability in data center. Whereas, their placement, the optimal controller and distribution workload issues are remain there in network domain. So, to provide guarantee related to QoS challenging. In many network, QoS is importance for many network workers to carry a provide services [10]. Present results, order to keep QoS, attempted to reductions the controller-switch interruption with the help of K−center method, or K−median method. Whereas, every current explanation are either dependent to topology or not calculat the load of resources or controller. This makes our work more motivate, this study providing the solution at control layer that maintain the QoS and reduce the resources costs, furthermore the main concern is to provide the explanation of the application-wise QoS that is topology self-governing.

In which main apprehension is to deals with SDN and Controller selection problem (CSP). Controller selection problem is more important than controller placement problem [27]. When flow arrives to any data plane from end devices in Distributed SDN, there should be specific limit on data plane flow to increase controller response time. Data plane will check different IP address in table to select controller on the bases of trust level, distance, bandwidth etc. Flow will arrive in controller and it will check, whether data need to prioritize to send locally or globally. On base of flow and QoS requirement engine prioritize each flow [38]. Different algorithms have been proposed for this like Mice and Elephant.

When prioritize process assign ranked for flow in controller. It will decide what will be batter for flow to serve to fulfill the quality of services requirement. For inter connectivity and exchange massage between peer controllers can be done by Software Define Networking inter massage through BGP. Underlying connected data plane and peer controller used inter SDN module for communication. Peer controller update and collects with help of state collection control function. When flow arrives on controller its basic need is to traffic the flow on time. Controller response time should be minimum to preform quality of services requirement [30].

Chapter 3
Research Methodology

3.1 Overview

The purpose of this chapter is to introduce the research strategies and research framework. Moreover, this chapter also define the simulation setup, simulation tool, research environment, assumption and limitations. This chapter is divided into three sections. In section 3.2, research strategy is defined. Section 3.3 is about simulation setup in which simulation metrics and performance metrics are discussed. Section 3.4 describe the physical model of research environment in which the metrics that discussed in section 3.3 are used to analyze and validate in physical model. Limitation and assumptions of the proposed scheme are presented in section 3.5.

3.2 Research Strategy

Research methodology flow consist of three phases as shown in Figure 3.1.

3.2.1 Phase 1

Single controller in network has limited scalability and computing power. In large network computing operations and traffic flow are very high. Single controller faces a number of concerns like scalability, robustness, bottleneck etc. Researchers have presented many approaches about controller capacity with large data towards controller placement problem but they are not satisfactory.

3.2.2 Phase 2

In this research our main goal is controller selection process when flow arrives any switch. In this study we are using distributed controller approach to avoid scalability and robustness Issues. The proposed technique reduces the workload, response time and manages quality of services between control layers, also investigating the location of controller in independent network topology.

3.2.3 Phase 3

MATLAB and GNS3 are flexible software simulation tool to provide easy and quick method to drive SDN approach and network prototype. Best feature of this tool is, it provides virtual environment to implement software define network protocols. Virtualization is absolutely same as physical hardware. It provides virtual image of data plane and control plane as well. We can configure both planes according to our requirement.

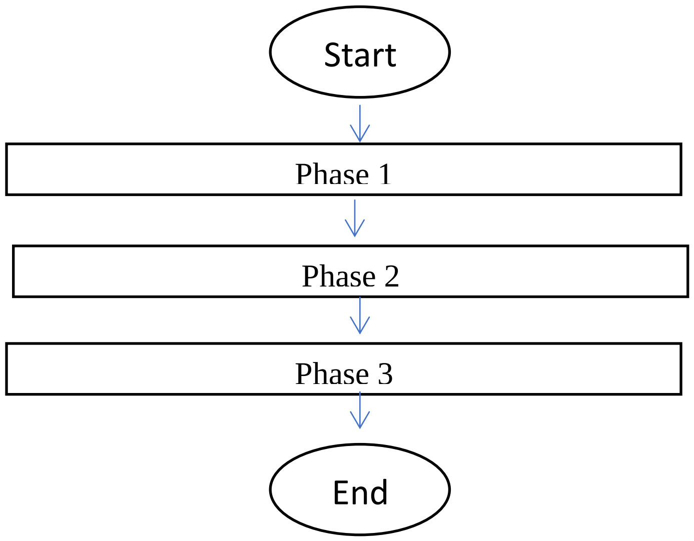


Figure 3.1: Research Methodology Flow

3.3 Simulation Setup

In this step we discussed the simulation setup of our methodology. First step is implemented on Matlab with respect to m/m/1 theorem. Single controller response time effected by maximum switches. To overcome this issue we are using Distributed SDN controller strategy to scale the network. For this scenario we are using M/M/1 and M/M/m theorem to explain the real environment simulation with the help of graph. Our second step is utilizing elephant or mice flow to prioritize the flow. Because of elephant flow we can detect our network topology. After that when flow will arrive in controller and it will check, whether data need to prioritize to send locally or globally. Also, it will reduce controller response time as well. In our third step we are using SDNi messages between controller to improve the performance with the help of border gateway protocol (BGP). In which we will use our desire topology to simulate or particular situation.

3.4 Physical Model

In this section, our study has contemporary the physical scenario of model. This study has conducted by MATLAB simulations of scheme at each controller. This study main concern is real-datasets and estimate the analytical theory to confirm our findings. It provides virtual image of data plane and control plane as well. We can configure both planes according to our requirement. We can implement network topology with help of MATLAB. We can configure many devices as we want to emulate SDN network with the help of MATLAB.

For suitability, to provide assistances of multiple controllers are summarized as follows:

• Scalability enabled.

• Enhance fault tolerance of controllers.

• Realize load balancing of controllers.

• Reduce the latency from switch to its closest controller.

Furthermore, this study analyze the output of multiple control plane. To provide modeling the flow set-up requirements through data plane to control plane for the group of coming process M/M/m with m control planes, the transition diagram is showed at Figure 3.2.

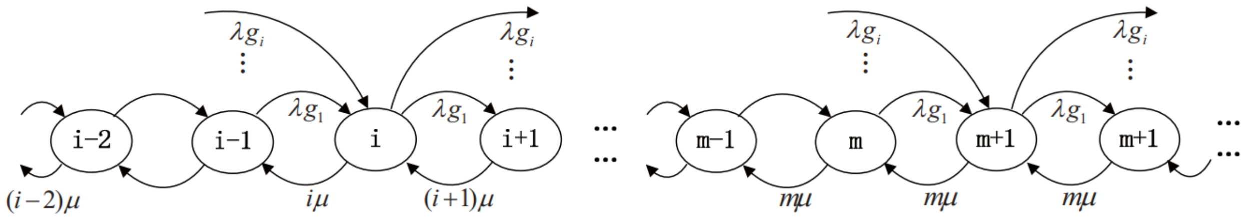


Figure 3.2: State diagram of M/M/m

3.4.1 SDN Controllers Communication

Distributed SDN can achieved by using the vertical or the horizontal method. In which we are focusing on horizontal approach [30]. Distributed SDN controllers can be stablish with the help of peer-to-peer controller communication, as it can be seen in Figure 3.3. Any controller wants to request for connections or information through neighbors, in its domain controllers from other domains in the network. It can be achieved east-west interface SDNi with using Border Gateway protocol that is simulated in Packet Tracer. The horizontal method is more feasible and preferable for geographies ascending across network, as control planes in this model can communicate with help of BGP to each other by standard and friendly protocol. This method retains the SDN controller independent in network, with path setup and distinct policies to relate to network elements in its control, while preserving a confederation with the adjacent networks domain.

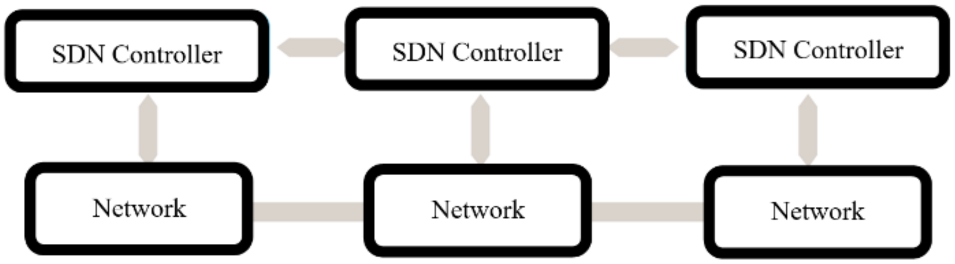


Figure 3.3: Inter-SDN Horizontal Approach for Controller Communication

BGP is famous routing protocol that required to share the routing material through two autonomous systems in the network. BGP is the model protocol and it can used to adapted for controller communication in inter-SDN environment. Whereas, many SDN network domains will have control and access of policies, QoS, and parameters that are useful for the SDN control planes. BGP one of the features that are used for controller negotiation through east-west interface for SDN. It can transmit reachability capability and information as message portion to their format. It is one of the most feasible and standard protocol for any peer to peer information to be replaced.

3.4.2 Flow scheduling System Framework

Traffic engineering and traditional flow are based on IP. SDN/OpenFlow [39] describes flows with the numerous forms of granularity like source and destination data plane, application types and VLAN. Hedera introduced, elephant flow in each single and large flow completed a certain (10% of the link) duration and size. That also identified with the help of same 10-tuple in network. whole account for internet video flows as for the most Cisco VNI network traffic as report predicts. Whereas, all flow is much fewer in video than 10% in flow to capacity of link. Elephant flow as different to individual, this study used the SDN/OpenFlow to aggregate big flow into a labelled aggregated for elephant flow: the flows are aggregated into one aggregated elephant flow when 1) flows share the congested link and share part of their path; 2) flows are video flows or big enough flows (for example: each flow is bigger than 1% of the congested link).

3.5 Assumption and Limitations

Following assumptions are taken into account while considering the entire scenario of the proposed scheme.

I. Our work does not consider any real application traffic.

II. In random topologies no optimization of the controller placement has been performed.

III. The results are based on abstract topologies and no real internet topology has been considered.

IV. The work is theoretical in nature with no emulation or simulation of large

Chapter 4
Proposed Solution

4.1 Overview

This chapter present our proposed Model in Distributed SDN. High traffic and its maintenance require addition and deletion of controllers in network. Many works have been done related to controller capacity, but the controller response time is more important to transfer flow. Main concern is end to end delay is more important factor. We cannot compromise with performance but cast and resource etc., are affordable. When data flow arrives on controller its basic need is to traffic the route on its time. Controller response time should be minimizing to perform quality of service as focused on controller capacity.

4.2 Mathematical Method

This study discusses the system modeling with the help of mathematical model. In which our work analysis the response time in order to present the resource requirement that are approximately utilized by our model. Our work simulates with the help of Opendaylight, Mininet, BGP and MATLAB. Many previous works explained in [40 ,41 ,42] that also focus on simulating our idea on mathematically. They have presented controller response through M/M/1 network model that can also extend through M/M/m discipline that will best way to evaluate our Distributed SDN performance. All SDN communication deployed in hierarchical architecture hierarchical architecture. Many literatures have been presented that M/M/m is the optimal way to show distributed results of SDN process including memoryless and additive properties [43, 41]. So, with the help of background mainstream we can use mathematical equation for our analysis to prove our strategy for Distributed SDN. In which we are considering incoming flow packets fallow the distributed Poisson constraint that is two processes given justification for different time scales [43]. Further, we are also considering exponential distribution for each controller for service rate, that is normal behavior in analysis of queueing model and existing researches is in-line also.[44, 45].

$$T(t) = k + 1 /2(\mu - \lambda\, k) \qquad (4.1) \text{ M/M/1}$$

$$T(t)= p0 *((m*\rho)\hat{}m/ m!(1 - \rho)) \qquad (4.2) \text{ M/M/m}$$

$$p0 = (1 + (m-1 \sum k=1) * (m\rho)\hat{}k/k! + (\infty) \sum (k=m) * (m\rho)\hat{}k / k! * 1/(m\hat{}k)-m)\hat{}-1 \quad (4.3)$$

TABLE 4.1: Useful Symbols

Symbols	Definitions
k	Total access switches
μ	Processing average flow rate of controller
λ	Average new flow arrival rate in each switch
S	Processing average flow time of controller
T	Service time average flow
N	Queue average size

4.3 Algorithm

We make use of the passive detection of congestion in SDN/OpenFlow network. The SDN controller defines the congestion level for OpenFlow switches. For example, when the link utilization is higher than a threshold of 90%, then switches will send a congestion notification via the Packet in message to the SDN controller [46]. The controller receives the message, and the message is parsed to obtain the following information:

1) The location of congestion by “switch ID” (data path identification or DPID);
2) Big flows on the congested link by “byte counters”.

The SDN controller calculates the counter of the flow meter of the congested switch to find flows that occupies more than certain amount of link utilization or video flows. In the following, we summarize our algorithm to detect the aggregated elephant flow for traffic engineering.

Algorithm 1: Aggregated elephant flow detection algorithm

Input: The flow Table entry set (OF Flow Stats Entry) of the congested switch which is collected through the Statistics Collector module of the SDN controller.

Output: An aggregated elephant flow:

ret=1 when a single flow as an elephant flow;

ret=2 when aggregated flows as an aggregated elephant flow.

The flow set flow set F in the aggregated elephant flow.

Algorithm 1 Aggregated elephant flow detection algorithm

```
1:  ret=0;
        F=null;
        flowentryset=StatisticsCollector.OF FlowStatsEntry;
        flowentryset.sortbybytesize;
2:  FOR $flow_j \in$ flowentryset do
3:      IF ($flow_j$.bytes $\geq$ THRESHOLD) THEN
4:          RETURN ret=1;
5:  FOR $flow_i \in$ flow entryset and $flow_j \in$ flowentryset do
6:      IF (flowi.srcDPID=$flow_j$.srcDPID)
            or $flow_i$.dstDPID=$flow_j$.dstDPID THEN
7:          F.add($flow_i$) and F.add($flow_j$);
8:  IF(F.bytes$\geq$THRESHOLD)
        THEN
9:          RETURN ret=2;
```

Whenever congestion is detected, the switch sends a congestion notification to the controller via the Packet_in messages in the OpenFlow protocol. As described above in the aggregated elephant flow detection algorithm, it is necessary to sort firstly all flows in the flow set of the switch flow table. Flows are sorted in terms of flow size or importance. Only important video flows may be considered for simplicity or in case of limit network resource. Then the algorithm detects a single flow or flows as an aggregated elephant flow when their flow.bytes ≥ THRESHOLD.

4.4 Network Method

Our network model consist of three main steps. First step is implemented on MATLAB with respect to m/m/1 theorem that has 30 switches and one controller. In whish we can understand the working of real scenario of network. Single controller response time effected by maximum switches. To overcome this issue we are using Distributed SDN controller strategy to scale the network. For this scenario we are using M/M/m theorem to explain the real environment simulation with the help of graph [30]. Our second step is utilize elephant or mice flow to prioritize the flow. Because of elephant flow we can detect our network topology. After that when flow will arrive in controller and it will check, whether data need to prioritize to send locally or globally. Also, it will reduce controller response time as well. In our third step we are using SDNi messages between controller to improve the performance with the help of border gateway protocol. In which we will use our desire topology to simulate or particular situation as show in Figure 4.1.

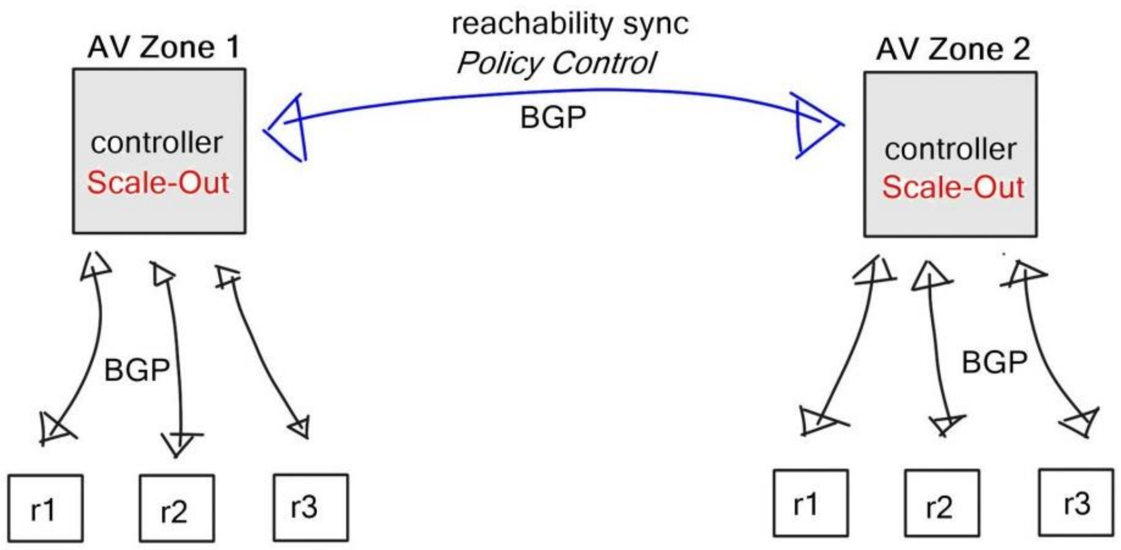


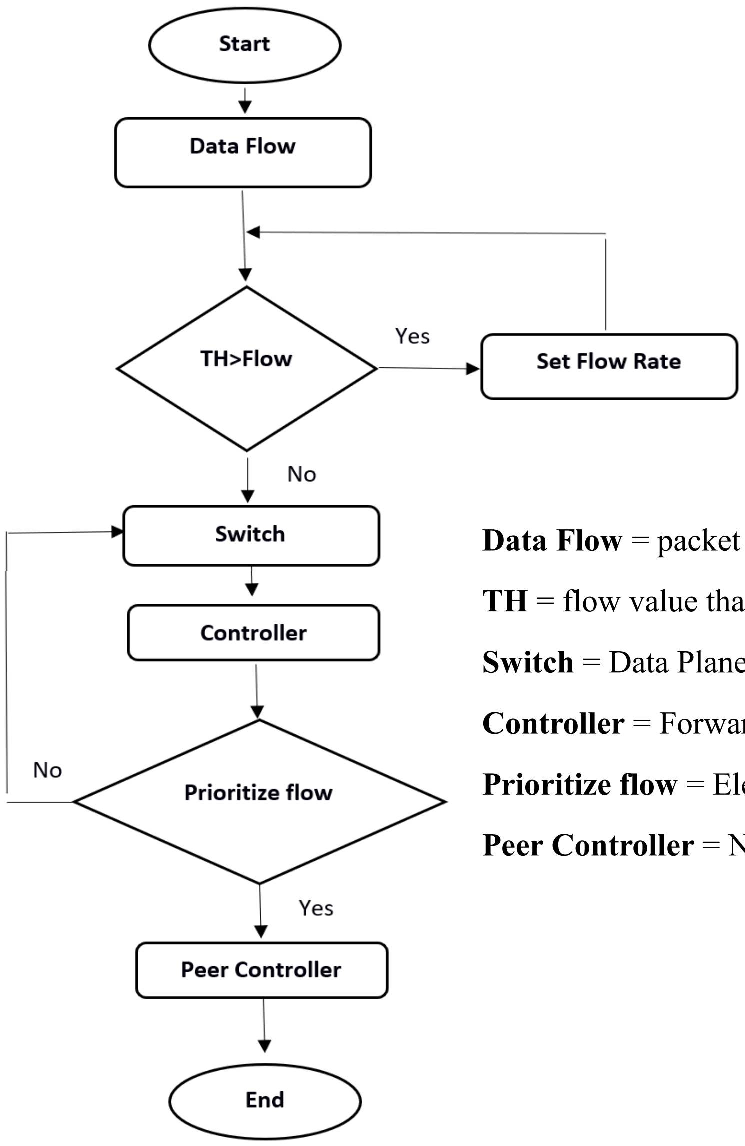
Figure 4.1: Network Topology

4.5 Proposed Model

Our propose method deals with SDN and Controller selection problem (CSP). Controller selection problem is more important than controller placement problem [47]. When data flow arrives to any data plane from end devices in Distributed SDN, limited data plane that should be set to increase controller response time. Data plane will check different IP address in table to select controller on the bases of trust level, distance, bandwidth etc. Flow will arrive in controller and it will check, whether data need to prioritize to send locally or globally. On base of flow and QoS requirement engine prioritize each flow [48]. Different algorithms have been proposed for this like Mice and Elephant.

When prioritize process assign ranked for flow in controller. It will decide what will be batter for flow to serve to fulfill the quality of services requirement. For inter connectivity and exchange massage between peer controllers can be done by Software Define Networking inter (SDNi) massage by using BGP. Underlying connected data plane and peer controller used inter SDN module for communication. Peer controller update and collects with help of state collection control function as shown in Figure 4.2.

When flow arrives on controller its basic need is to traffic the route on time. Controller response time should be minimum to preform quality of services requirement. Quality of services and dynamic workload requirement needs for logically adding and deleting controllers in network. Our methodology is supportive for invoke and revoke number of logical controllers.



Data Flow = packet form sender to receiver

TH = flow value that has some boundary

Switch = Data Plane

Controller = Forwarding Plane

Prioritize flow = Elephant Flow

Peer Controller = Neighbor Controller

Figure 4.2: The block diagram of proposed solution

Chapter 5
Experimental Result

5.1 Overview

High traffic and their maintenance required addition and deletion of controller in network. Many works have been done related to controller capacity but the controller response time is more important to transfer flow end to end. [31, 37] as explained before in two main points that end to end delay is more important factor. We cannot compromise with performance but cast and resource etc., are affordable. When large data flow arrives on controller its basic need is to traffic the route on its time. When flow arrives through end devices to any data plane in Distributed SDN, our model will support the data plane to increase controller response time. Also, controller will check, whether data needs to prioritize flow to send locally or globally with the help of Elephant or Mice flow. This model is used to reduces the controller work load, response time and manage quality of services (QoS) between controllers.

5.2 Physical and Process Model

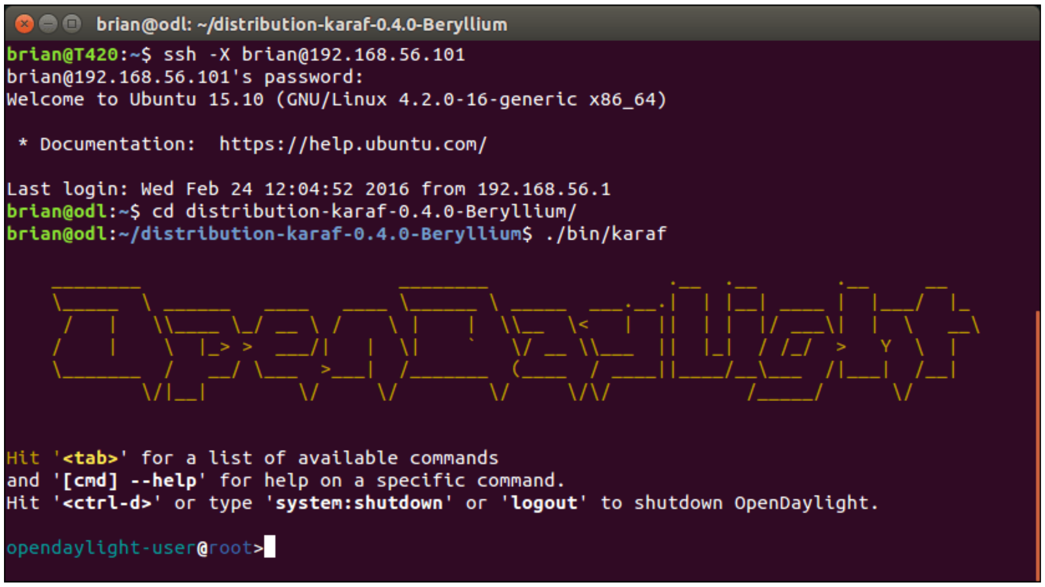
Our first step is evolving around network topology. In which we have centralized SDN that is implemented on Opendaylight platform with coordination of Mininet. We are assuming the topology that is independent to network. Our main purpose is to explain how elephant flow evolve in any SDN topology. In this phase, we have re-implemented the work of Jing Liu et al [49] to test their proposed Load Balancing technique to manage the elephant flows in data centers using SDN technology. We have compared their approach of effectively assigning the link weights to adjust the data flow amongst the links with Single Shortest Path Routing technique. We used the OpendayLight Controller in OpenFlow testbed and observed that Weighted Multipath Routing outperforms the Single Path Routing, especially in the case of Elephant Flows. We observed that their approach performs in case of heavy flows, effectively overcoming the network congestion and improving the network utilization. Thus, it will provide better Quality of Service to the customers.

Install OpenDaylight and Mininet Next, install the minimum set of features required to test OpenDaylight and the OpenDaylight

GUI: opendaylight-user@root> feature:install odl-restconf odl-l2switch-switch odl-mdsal-apidocs odl-dlux-all

The above is an example of installing optional modules in a karaf container. You only need to install an optional feature once. Once installed, these features are permanently added to the controller and will run every time it starts.

Now on we have presented our network simulation with the help pf Opendaylight and Mininet



```
brian@odl: ~/distribution-karaf-0.4.0-Beryllium
brian@T420:~$ ssh -X brian@192.168.56.101
brian@192.168.56.101's password:
Welcome to Ubuntu 15.10 (GNU/Linux 4.2.0-16-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

Last login: Wed Feb 24 12:04:52 2016 from 192.168.56.1
brian@odl:~$ cd distribution-karaf-0.4.0-Beryllium/
brian@odl:~/distribution-karaf-0.4.0-Beryllium$ ./bin/karaf

    ________                       ________                .__  .__       .__     __
    \_____  \ ______   ____   ____ \______ \ _____  ___.__.|  | |__| ____ |  |___/  |_
     /   |   \\____ \_/ __ \ /    \ |    |  \\__  \<   |  ||  | |  |/ ___\|  |  \   __\
    /    |    \  |_> >  ___/|   |  \|    `   \/ __ \\___  ||  |_|  / /_/  >   Y  \  |
    \_______  /   __/ \___  >___|  /_______  (____  / ____||____/__\___  /|___|  /__|
            \/|__|        \/     \/        \/     \/\/            /_____/      \/


Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.

opendaylight-user@root>
```

Figure 5.1: We install Opendaylight and Mininet.

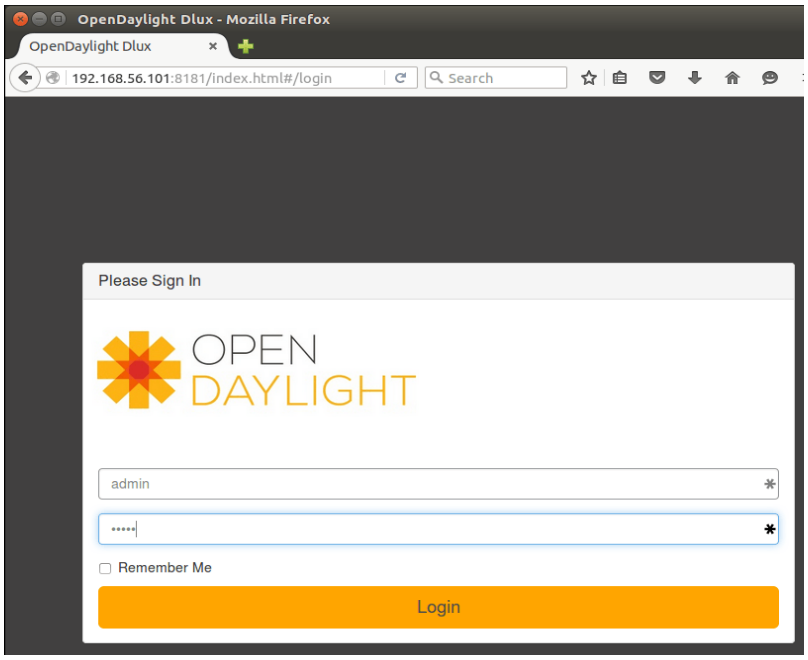


Figure 5.2: Collaborate both and assign ip to SDN mininet controller that will call in Opendalight

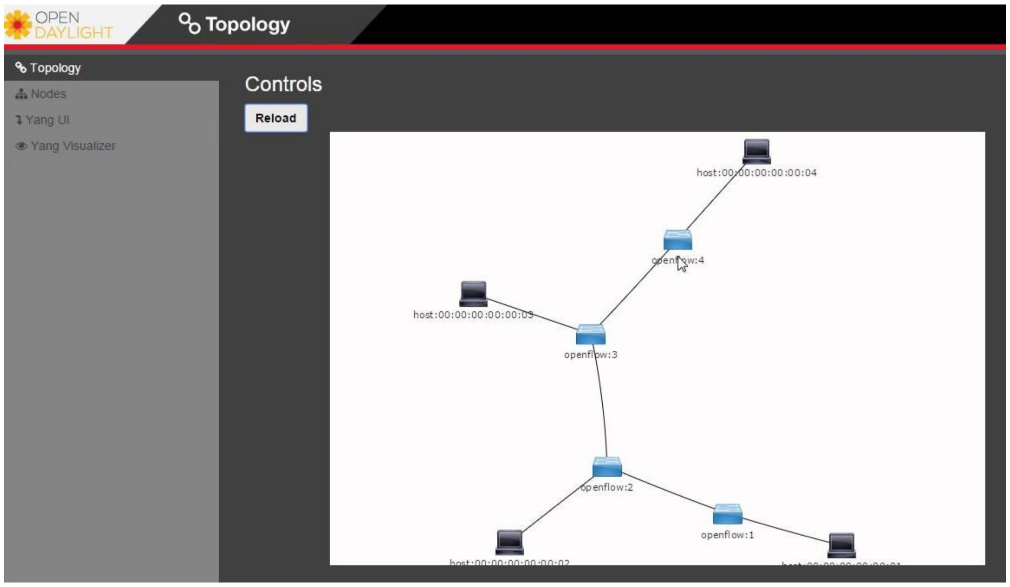
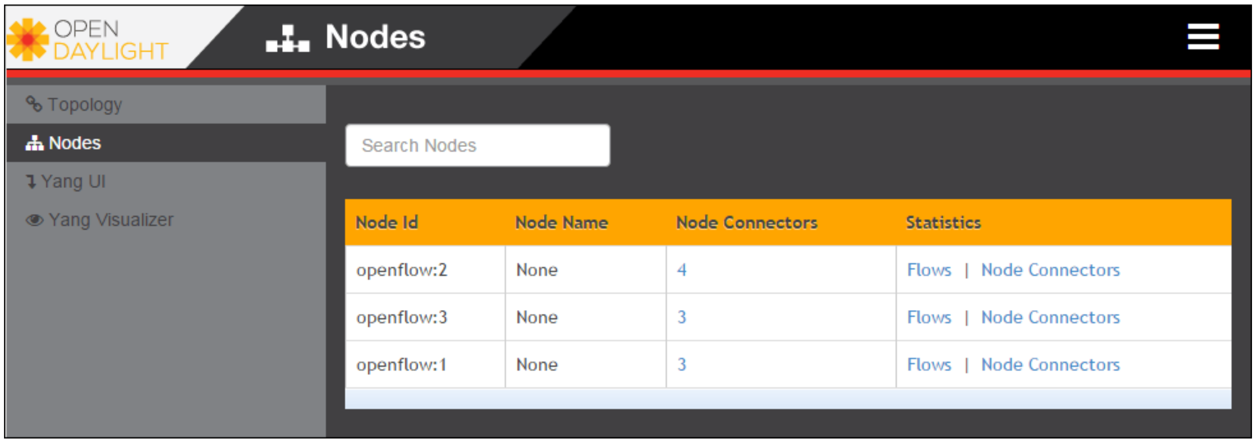
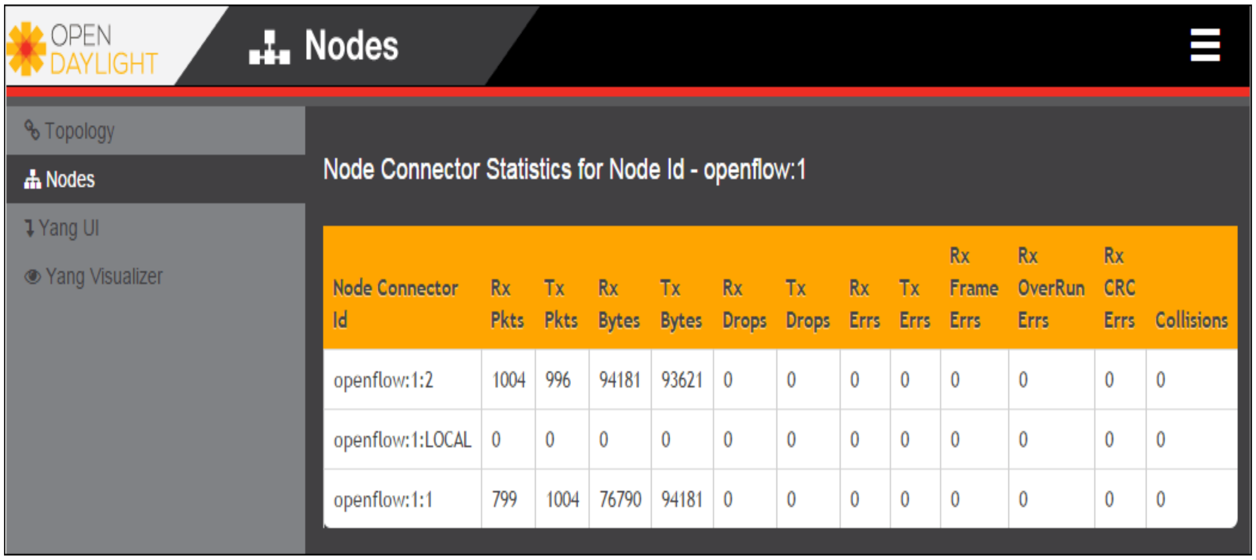


Figure 5.3: After installing features of opendaylight and mininet topology we can see our topology in our web browser



Node Id	Node Name	Node Connectors	Statistics
openflow:2	None	4	Flows | Node Connectors
openflow:3	None	3	Flows | Node Connectors
openflow:1	None	3	Flows | Node Connectors

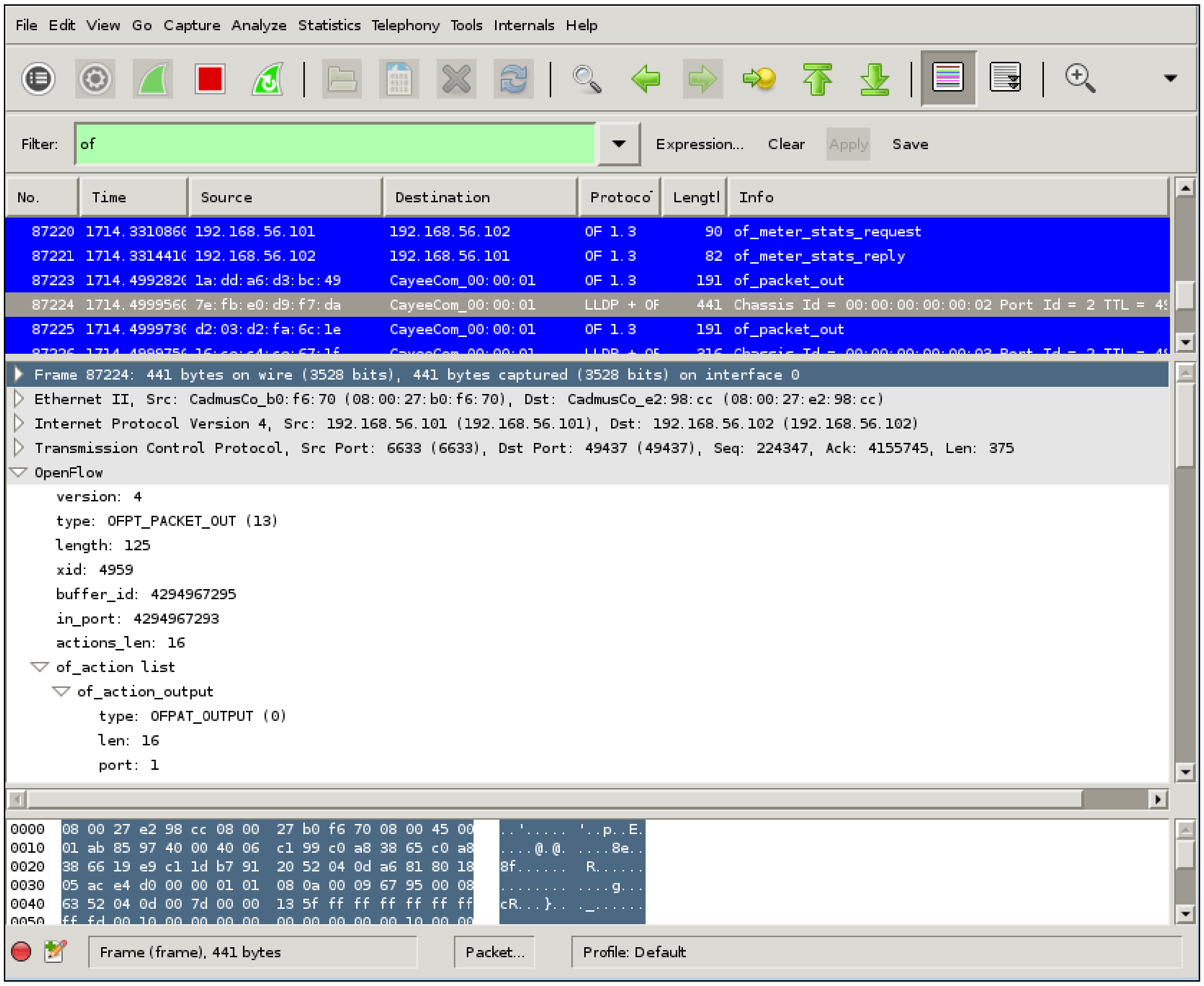
Figure 5.4: When we have opendaylight platform we can easily overlook our topology and its node information and statistics



Node Connector Statistics for Node Id - openflow:1

Node Connector Id	Rx Pkts	Tx Pkts	Rx Bytes	Tx Bytes	Rx Drops	Tx Drops	Rx Errs	Tx Errs	Rx Frame Errs	Rx OverRun Errs	Rx CRC Errs	Collisions
openflow:1:2	1004	996	94181	93621	0	0	0	0	0	0	0	0
openflow:1:LOCAL	0	0	0	0	0	0	0	0	0	0	0	0
openflow:1:1	799	1004	76790	94181	0	0	0	0	0	0	0	0

Figure 5.5: After running our topology and getting its all statistics we can Capturing OpenFlow Messages with the help of Wireshark


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: of
Expression... Clear Apply Save

```
No.     Time          Source              Destination        Protocol  Length  Info
 87220  1714.3310860  192.168.56.101      192.168.56.102     OF 1.3        90  of_meter_stats_request
 87221  1714.3314410  192.168.56.102      192.168.56.101     OF 1.3        82  of_meter_stats_reply
 87223  1714.4992820  1a:dd:a6:d3:bc:49   CayeeCom_00:00:01  OF 1.3       191  of_packet_out
 87224  1714.4999560  7e:fb:e0:d9:f7:da   CayeeCom_00:00:01  LLDP + OF    441  Chassis Id = 00:00:00:00:00:02 Port Id = 2 TTL = 4
 87225  1714.4999730  d2:03:d2:fa:6c:1e   CayeeCom_00:00:01  OF 1.3       191  of_packet_out
```

```
▷ Frame 87224: 441 bytes on wire (3528 bits), 441 bytes captured (3528 bits) on interface 0
▷ Ethernet II, Src: CadmusCo_b0:f6:70 (08:00:27:b0:f6:70), Dst: CadmusCo_e2:98:cc (08:00:27:e2:98:cc)
▷ Internet Protocol Version 4, Src: 192.168.56.101 (192.168.56.101), Dst: 192.168.56.102 (192.168.56.102)
▷ Transmission Control Protocol, Src Port: 6633 (6633), Dst Port: 49437 (49437), Seq: 224347, Ack: 4155745, Len: 375
▽ OpenFlow
    version: 4
    type: OFPT_PACKET_OUT (13)
    length: 125
    xid: 4959
    buffer_id: 4294967295
    in_port: 4294967293
    actions_len: 16
  ▽ of_action list
    ▽ of_action_output
        type: OFPAT_OUTPUT (0)
        len: 16
        port: 1
```

```
0000  08 00 27 e2 98 cc 08 00  27 b0 f6 70 08 00 45 00   ..'.....  '..p..E.
0010  01 ab 85 97 40 00 40 06  c1 99 c0 a8 38 65 c0 a8   ....@.@.  ....8e..
0020  38 66 19 e9 c1 1d b7 91  20 52 04 0d a6 81 80 18   8f......   R......
0030  05 ac e4 d0 00 00 01 01  08 0a 00 09 67 95 00 08   ........  ....g...
0040  63 52 04 0d 00 7d 00 00  13 5f ff ff ff ff ff ff   cR...}..  ._......
```

Figure 5.6: In our last step we have Matlab graph to demonstrate behavior of response time of controller to multiple switches

5.3 SDNi Distributed Approach

OpenDaylight is a consortium created with the help of major industry players to achieve the same objective. Cisco has already contributed an open source SDN controller to the ODL community known as OpenDaylight Controller as shown in Figure 5.7. OpenDaylight SDN Controller (ODL) presents a new SDN controller architecture based on Services Abstraction Layer (SAL) concept. ODL also supports protocols other than OpenFlow. The work presented is differs from all previous studies since no previous study has considered OpenDaylight SDN Controller.

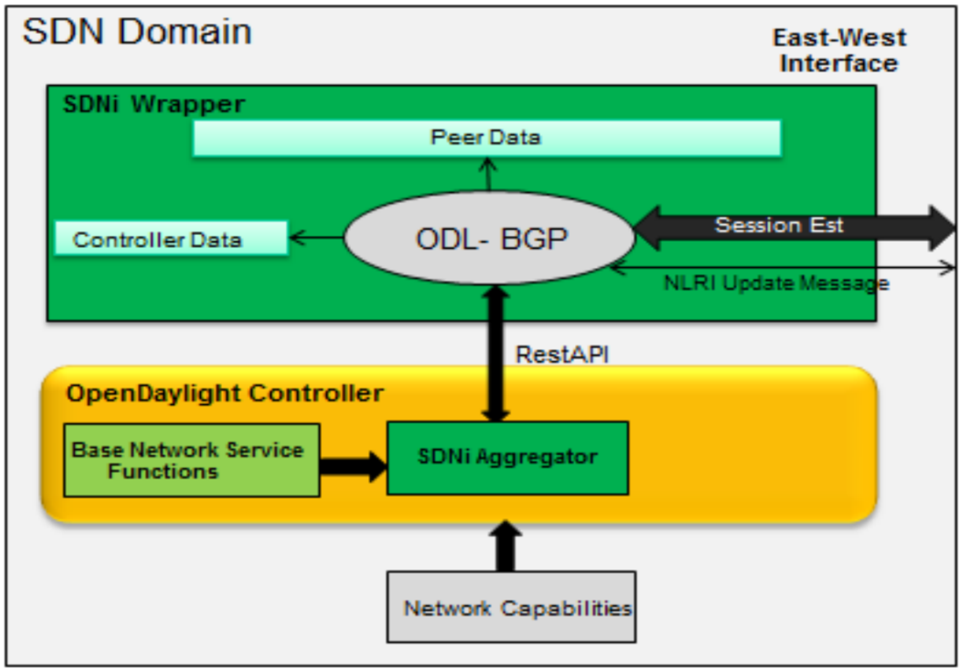


Figure 5.7: ODL-SDNi working

Wrapper Features:

• SDNi Wrapper utilizes the existing ODL-BGP Plugin.

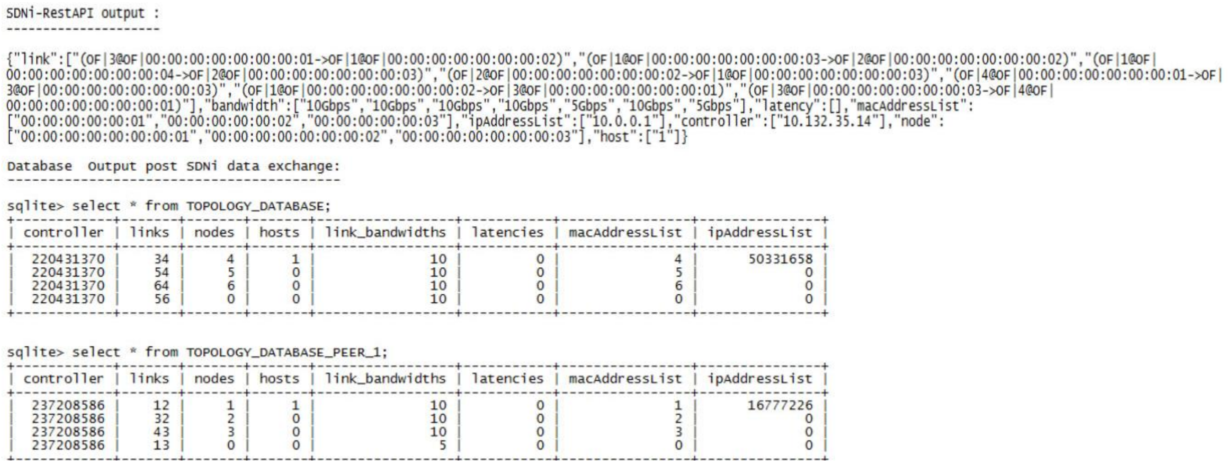
• Enhanced the NLRI update message (of BGP) for capability data

• This data to be exchanged available through the RestAPIs that are developed.

• Wrapper to read and store this data in a database (SQLite).

• Each controller to have peer data for the controllers in a session over real-time.

• The data exchanged can be restricted (based on security)



```
SDNi-RestAPI output :
---------------------

{"link":["(OF|3@OF|00:00:00:00:00:00:00:01->OF|1@OF|00:00:00:00:00:00:00:02)","(OF|1@OF|00:00:00:00:00:00:00:03->OF|2@OF|00:00:00:00:00:00:00:02)","(OF|1@OF|
00:00:00:00:00:00:00:04->OF|2@OF|00:00:00:00:00:00:00:03)","(OF|2@OF|00:00:00:00:00:00:00:02->OF|1@OF|00:00:00:00:00:00:00:03)","(OF|4@OF|00:00:00:00:00:00:00:01->OF|
3@OF|00:00:00:00:00:00:00:03)","(OF|1@OF|00:00:00:00:00:00:00:02->OF|3@OF|00:00:00:00:00:00:00:01)","(OF|3@OF|00:00:00:00:00:00:00:03->OF|4@OF|
00:00:00:00:00:00:00:01)"],"bandwidth":["10Gbps","10Gbps","10Gbps","10Gbps","5Gbps","10Gbps","5Gbps"],"latency":[],"macAddressList":
["00:00:00:00:00:01","00:00:00:00:00:02","00:00:00:00:00:03"],"ipAddressList":["10.0.0.1"],"controller":["10.132.35.14"],"node":
["00:00:00:00:00:00:00:01","00:00:00:00:00:00:00:02","00:00:00:00:00:00:00:03"],"host":["1"]}

Database  Output post SDNi data exchange:
----------------------------------------

sqlite> select * from TOPOLOGY_DATABASE;
+------------+-------+-------+-------+-----------------+-----------+----------------+--------------+
| controller | links | nodes | hosts | link_bandwidths | latencies | macAddressList | ipAddressList |
+------------+-------+-------+-------+-----------------+-----------+----------------+--------------+
|  220431370 |    34 |     4 |     1 |              10 |         0 |              4 |     50331658 |
|  220431370 |    54 |     5 |     0 |              10 |         0 |              5 |            0 |
|  220431370 |    64 |     6 |     0 |              10 |         0 |              6 |            0 |
|  220431370 |    56 |     0 |     0 |              10 |         0 |              0 |            0 |
+------------+-------+-------+-------+-----------------+-----------+----------------+--------------+

sqlite> select * from TOPOLOGY_DATABASE_PEER_1;
+------------+-------+-------+-------+-----------------+-----------+----------------+--------------+
| controller | links | nodes | hosts | link_bandwidths | latencies | macAddressList | ipAddressList |
+------------+-------+-------+-------+-----------------+-----------+----------------+--------------+
|  237208586 |    12 |     1 |     1 |              10 |         0 |              1 |     16777226 |
|  237208586 |    32 |     2 |     0 |              10 |         0 |              2 |            0 |
|  237208586 |    43 |     3 |     0 |              10 |         0 |              3 |            0 |
|  237208586 |    13 |     0 |     0 |               5 |         0 |              0 |            0 |
+------------+-------+-------+-------+-----------------+-----------+----------------+--------------+
```

Figure 5.8: Sample SDNi Rest API Output

5.3.1 BGP Protocol

After implementing centralized SDN topology we can implement our Distributed SDN with the help of Packet Tracer. Due to limitation of Mininet tool we cannot implement communication between two controllers. So, we are assuming controller to controller communication with the help of two router that are connected with each other with BGP protocol to share information. Also, there are multiple switches they are acting like data plane. Furthermore, we also have end dives they can communicate with each other as shown in Figure 9.

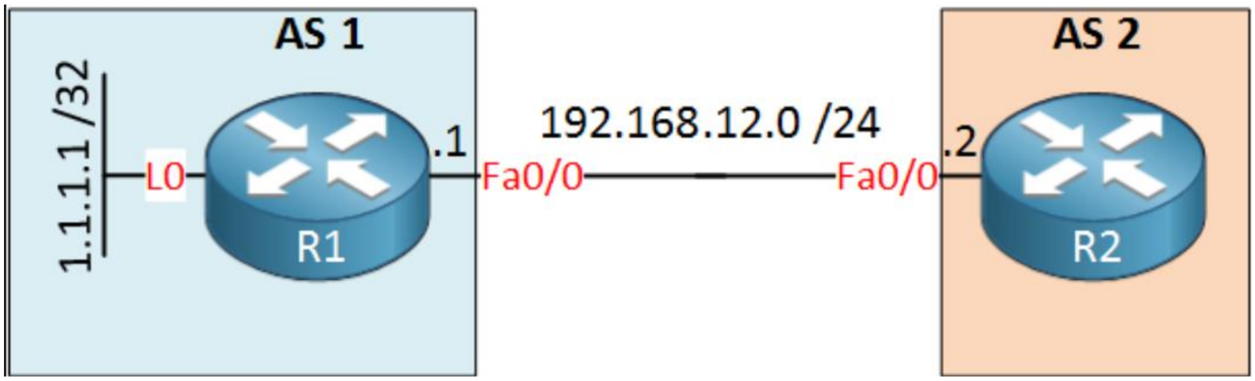


Figure 5.9: BGP working

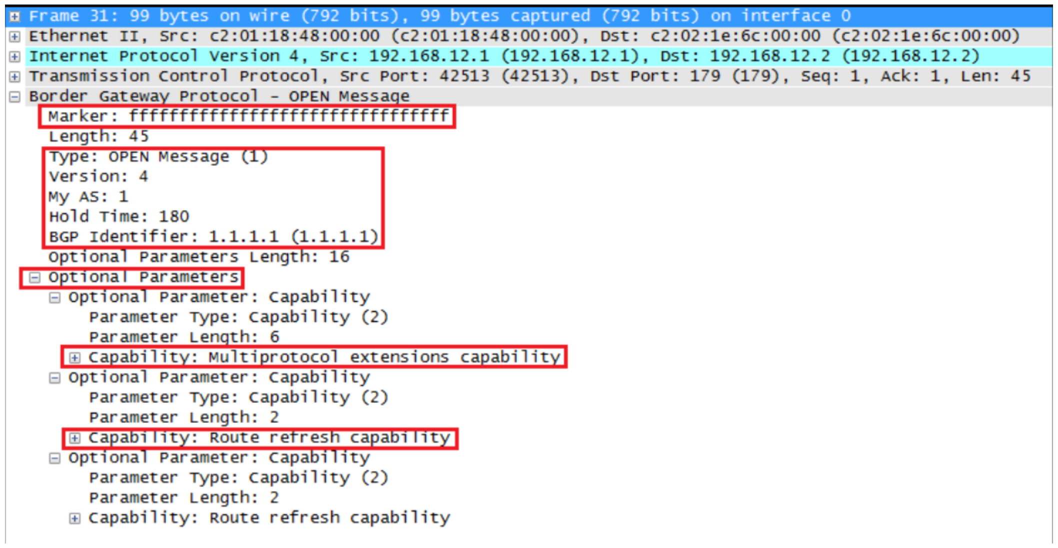
BGP is the core routing protocol of the Internet. It is described as a path vector protocol and does not use traditional IGP (OSPF, EIGRP, RIP) metrics, but makes routing decisions based on path, network policies and/or rule sets. It maintains a table of IP networks or 'prefixes' which designate network reachability among autonomous systems (AS). In this Free Cisco Lab Packet Tracer activity, we will learn to use the limited BGP functionality of Cisco’s Packet Tracer to configure a complex BGP network. We will also refresh some of our other knowledge such as configuring EIGRP and OSPF in multiple areas. We will also learn to redistribute learned routes from the Static, EIGRP and OSPF network in to BGP. Lastly, we will verify and test the connectivity of our network. BGP uses a variety of messages for establishing the connection, exchanging routing information, checking if the remote BGP neighbor is still there and/or notifying the remote side if any errors occur.

To do all of this, BGP uses 4 messages:

- Open Message
- Update Message
- Keepalive Message
- Notification Message

5.3.2 Open Message

You can see the open message from R1 to R2. You can see the things that we discussed, the BGP version, AS number, hold time, BGP ID and the optional parameters (MP-BGP and route refresh). The marker field on top is used to indicate if we use MD5 authentication. When it’s filled with 1’s then we are not using authentication.

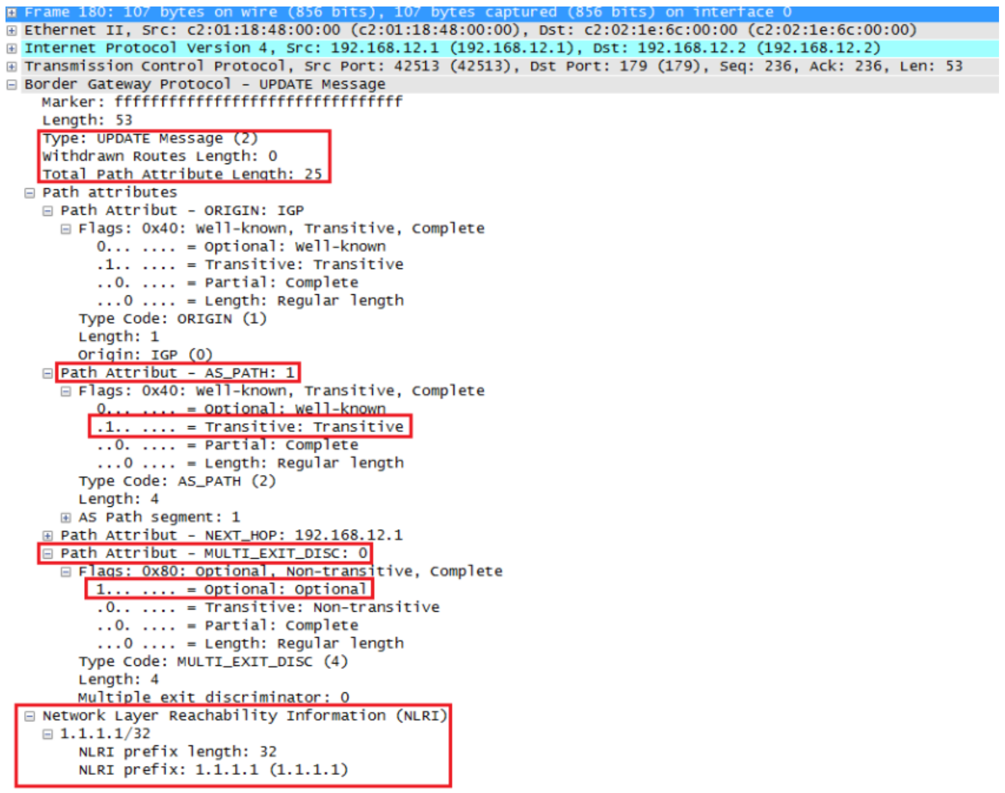


```
⊞ Frame 31: 99 bytes on wire (792 bits), 99 bytes captured (792 bits) on interface 0
⊞ Ethernet II, Src: c2:01:18:48:00:00 (c2:01:18:48:00:00), Dst: c2:02:1e:6c:00:00 (c2:02:1e:6c:00:00)
⊞ Internet Protocol Version 4, Src: 192.168.12.1 (192.168.12.1), Dst: 192.168.12.2 (192.168.12.2)
⊞ Transmission Control Protocol, Src Port: 42513 (42513), Dst Port: 179 (179), Seq: 1, Ack: 1, Len: 45
⊟ Border Gateway Protocol - OPEN Message
    Marker: ffffffffffffffffffffffffffffffff
    Length: 45
    Type: OPEN Message (1)
    Version: 4
    My AS: 1
    Hold Time: 180
    BGP Identifier: 1.1.1.1 (1.1.1.1)
    Optional Parameters Length: 16
  ⊟ Optional Parameters
    ⊟ Optional Parameter: Capability
        Parameter Type: Capability (2)
        Parameter Length: 6
      ⊞ Capability: Multiprotocol extensions capability
    ⊟ Optional Parameter: Capability
        Parameter Type: Capability (2)
        Parameter Length: 2
      ⊞ Capability: Route refresh capability
    ⊟ Optional Parameter: Capability
        Parameter Type: Capability (2)
        Parameter Length: 2
      ⊞ Capability: Route refresh capability
```

Figure 5.10: Here’s an example of a Wireshark capture of an open message between R1 and R2

5.3.3 Update Message

You can see a update message from R1. No routes are withdrawn and there are a couple of BGP attributes. You can see the ORIGIN, AS_PATH and MULTI_EXIT_DISC (MED). I also highlighted some of the flags. The AS_PATH attribute is transitive while MULTI_EXIT_DISC is optional. At the bottom you can find the NLRI information with our prefix.



```
⊞ Frame 180: 107 bytes on wire (856 bits), 107 bytes captured (856 bits) on interface 0
⊞ Ethernet II, Src: c2:01:18:48:00:00 (c2:01:18:48:00:00), Dst: c2:02:1e:6c:00:00 (c2:02:1e:6c:00:00)
⊞ Internet Protocol Version 4, Src: 192.168.12.1 (192.168.12.1), Dst: 192.168.12.2 (192.168.12.2)
⊞ Transmission Control Protocol, Src Port: 42513 (42513), Dst Port: 179 (179), Seq: 236, Ack: 236, Len: 53
⊟ Border Gateway Protocol - UPDATE Message
    Marker: ffffffffffffffffffffffffffffffff
    Length: 53
    Type: UPDATE Message (2)
    Withdrawn Routes Length: 0
    Total Path Attribute Length: 25
  ⊟ Path attributes
    ⊟ Path Attribut - ORIGIN: IGP
      ⊟ Flags: 0x40: Well-known, Transitive, Complete
          0... .... = Optional: Well-known
          .1.. .... = Transitive: Transitive
          ..0. .... = Partial: Complete
          ...0 .... = Length: Regular length
        Type Code: ORIGIN (1)
        Length: 1
        Origin: IGP (0)
    ⊟ Path Attribut - AS_PATH: 1
      ⊟ Flags: 0x40: Well-known, Transitive, Complete
          0... .... = Optional: Well-known
          .1.. .... = Transitive: Transitive
          ..0. .... = Partial: Complete
          ...0 .... = Length: Regular length
        Type Code: AS_PATH (2)
        Length: 4
      ⊞ AS Path segment: 1
    ⊞ Path Attribut - NEXT_HOP: 192.168.12.1
    ⊟ Path Attribut - MULTI_EXIT_DISC: 0
      ⊟ Flags: 0x80: Optional, Non-transitive, Complete
          1... .... = Optional: Optional
          .0.. .... = Transitive: Non-transitive
          ..0. .... = Partial: Complete
          ...0 .... = Length: Regular length
        Type Code: MULTI_EXIT_DISC (4)
        Length: 4
        Multiple exit discriminator: 0
  ⊟ Network Layer Reachability Information (NLRI)
    ⊟ 1.1.1.1/32
        NLRI prefix length: 32
        NLRI prefix: 1.1.1.1 (1.1.1.1)
```

Figure 5.11: Capture a update message from R1

5.4 Simulation Result

To provides virtual image of data plane and control plane. We can configure both planes according to our requirements. We can implement network topology with help of our MATLAB. We can configure many devices as we want to emulate SDN network.

$$T(t) = k + 1 /2(\mu - \lambda k) \qquad (5.1)$$

In our first analysis we are comparing our two graphs to obtain comparison between SDN controller response towards without any threshold and with threshold. With help of our graphs we can clearly concluded that controller response time effected by switch flow request from SDN controller. In our first graph we have set controller response time μ =10000 and 10 switches. Their flow rate towards controller is λ =100, λ =200 and λ =300. Our first graph shows that controller response time increase when there is no threshold. In our second graph we set the threshold value 150. We analyses that controller response time is more better against our first graph.

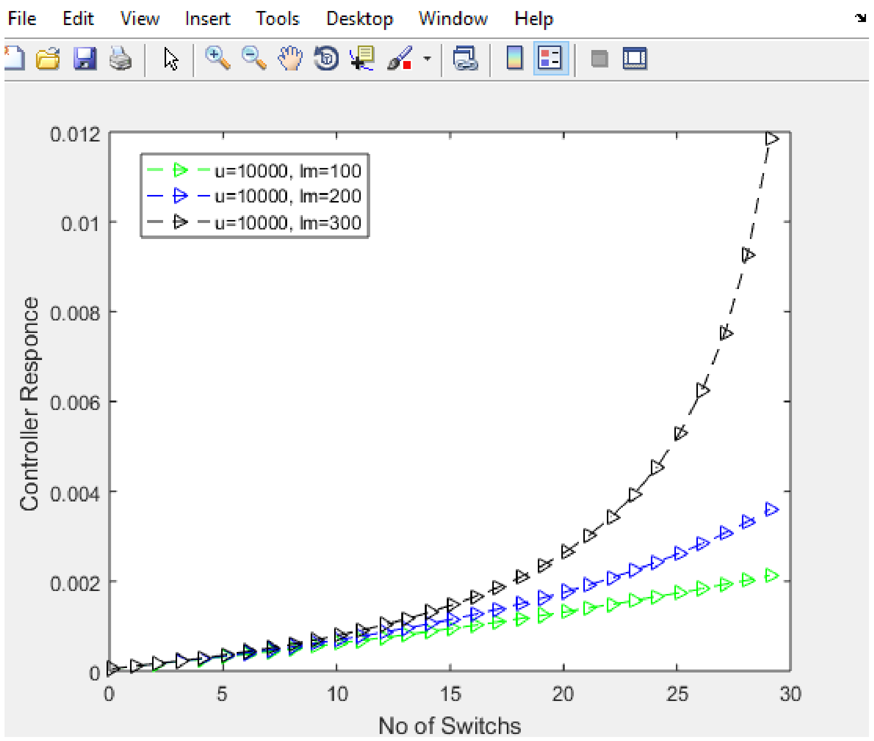


Figure 5.12: μ and the number of switches without Threshold

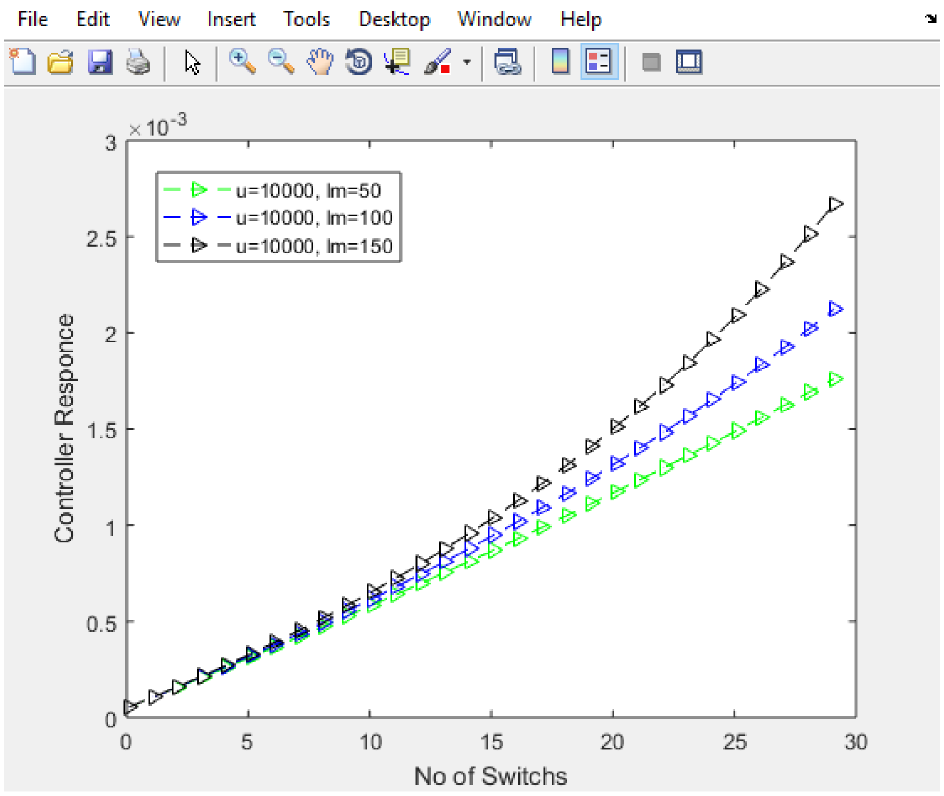


Figure 5.13: μ and the number of switches with Threshold

Our two analysis are performed in MATLAB. As Figure 5.12 explained with help of Equation (1), that described the coordination between in one controller to S switch and flow setup time to controller. Whereas, with the help of ref. [24] our results are presenting the statistics that are very close to real-world example, such as arrival of packet rate (λ pps) and rate of service (μ). As the Figure 5.13 explains, there is a relationship between controller and switches. While, 10 switches are connected with one controller and time for each flow required 1ms setup is at fixed λ = 100 and μ = 10,000. On the other hand, to manage at μ = 30,000 of according to execute 10 switches and also take care of changing of μ time for flow setup time.

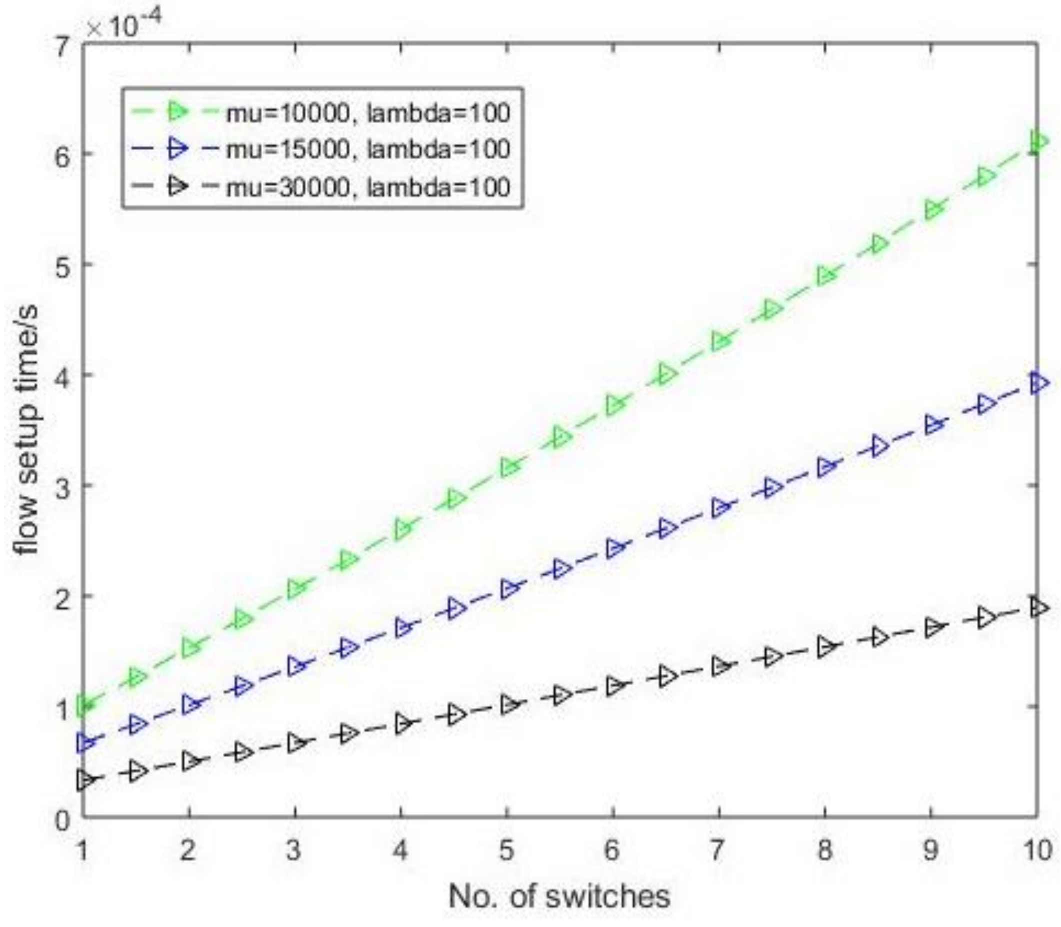


Figure 5.14: Relationship between μ and the number of switches with regards to flow-setup time

In our analysis with respect to Figure 5.13, for observing the difference between the time for flow setup μ of the switches data rate (200 packets) changed. It is clearly seen that in both figures, at μ = 10,000 to manage switches that controller capacity changed as well.

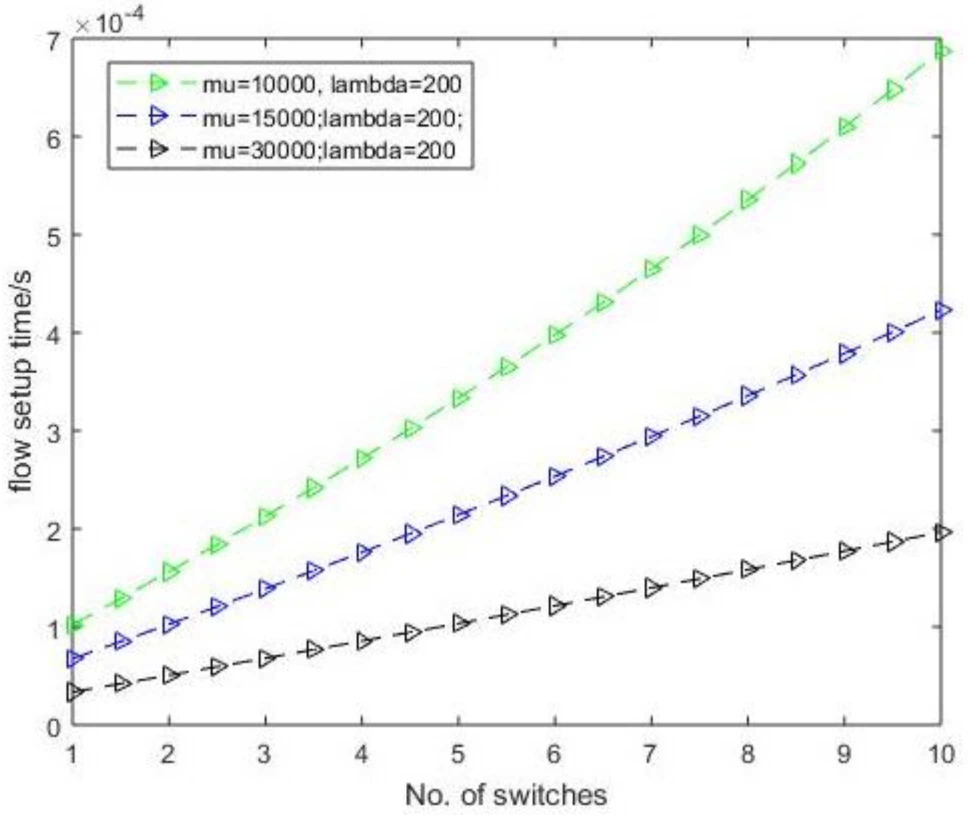


Figure 5.15: Flow-setup time relationship with μ and the number of switches

In results our analysis showed that capacity of controllers significantly affects the QoS in context of applications requirement. Whereas, to guaranteed the application of QoS requirement should has some limits or certain boundary for underlying switches or need distributed approach. Also, it required dynamic demands for flow balancing. A glance at the graphs reveals, that capacity of controller rate change when underlying switches rate increased. So, because of this we need to add more resource in network that will increased our cast but that will increase our work efficiency as well.

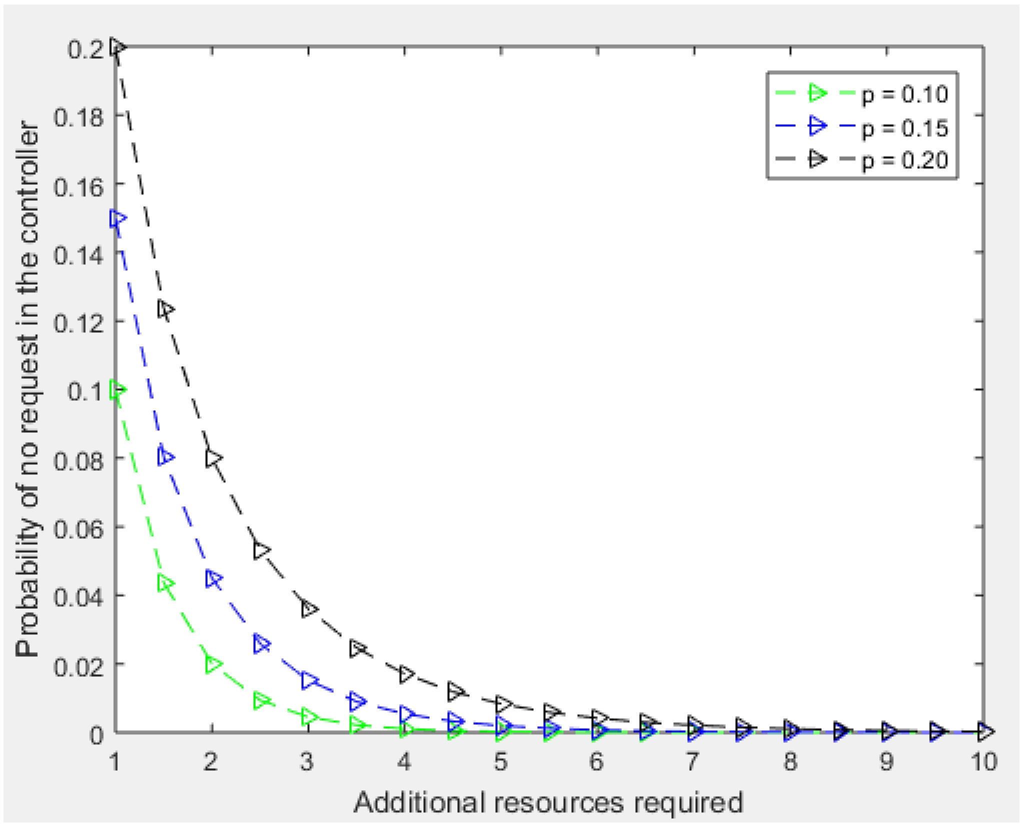


Figure 5.16: The relationship between p0 and extra required resources (m), by ρ

This experiment explained, the control plane capacity expressively affects the QoS application requirements. As a significance, to maintain its QoS requirements specific application, we have to put some boundary on switches that should be under limited or certain bound, and dynamic flow is necessary for flow balancing. by equation (5.2) where M/M/1 model, Figure 5.14 exposes that number of flows in the queue fluctuates as traffic intensity ($\rho = \lambda/\mu$) varies with probability.

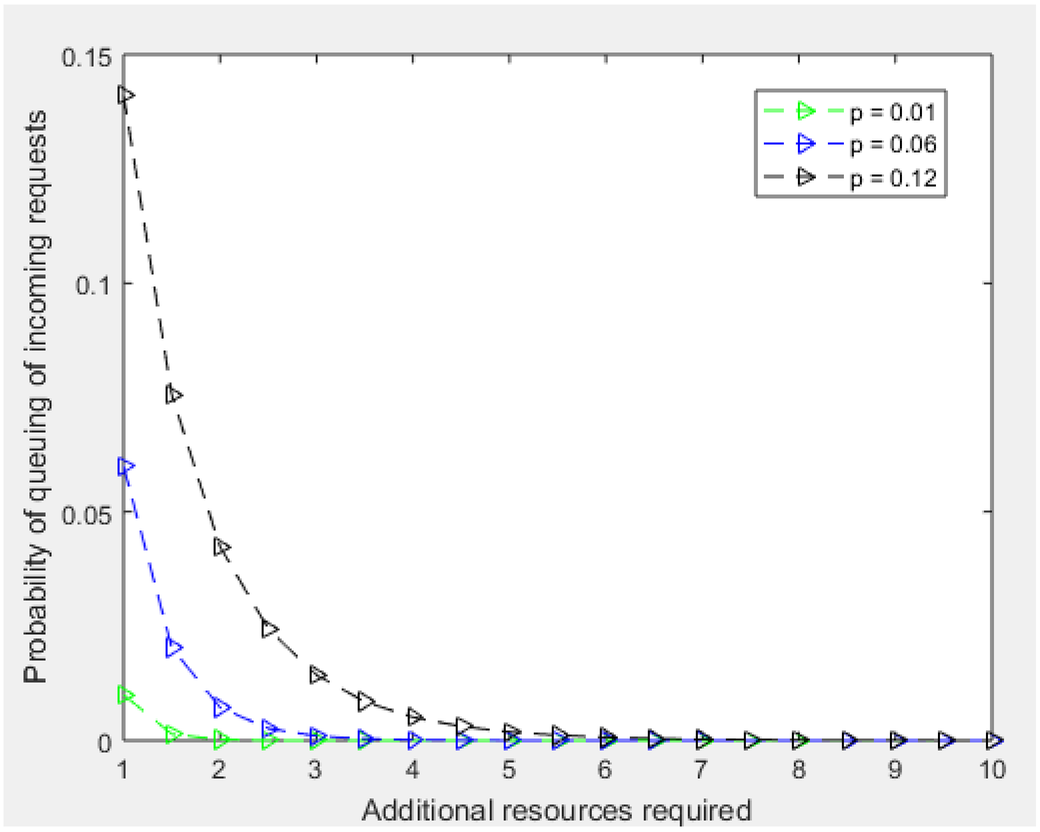


Figure 5.17: The relation of p queueing and more required resources (m), by ρ

$$T(t)= p0 *((m*\rho)^m/ m!(1 - \rho)) \quad (5.3)\ M/M/m$$

Now we have equation (5.3) representation of M/M/m, that indicated by advancing additional resources in the network domain, queue decreases the probability of data flows considerably, as shown in Figure 5.15 Relating Figure 5.14 and 5.15 at given ρ = 0.7, we can analyze that queue decreases with probability streams of flow that when we deploy the extra resource.

Chapter 6
Conclusion and Future work

6.1 Conclusion

SDN has been collaborated with different computer domains to facilitate the network. Our Research work is specifically focus on distributed SDN and Controller Selection Problem (CSP). Main goal behind our model is, to provide QoS and reduce controller response time, when flow arrive in switches, there should be some boundary that specify how many switches controller can manage and that will use to provide limited flow rate to controller to minimize controller response time. Controller will prioritize the flow with help of Mice or Elephant algorithms to facilitate network globally or locally. Controller will decide whether to deal with flow locally or peer controller. In order to improve the scalability, we have to use Distributed SDN to avoid bottleneck significant problem, centralized SDN have robustness, bottleneck and scalability problem. Also, it is the main and general approach to provide scalability. Whereas, load distribution, their placement and optimal controller such as problems are always there. So, to provide guarantee in SDN for QoS in not easy task. But to provide service on time and consist that is the main concern for network designer to provide QoS.

The emergence of SDN is imposing novel requirements due to diverse infrastructural entities and architectures. In this work, firstly we discussed that the complex, heterogeneous, and hierarchical SDN deployments affect the application performance (QoS) and end-user experience. After-that, we analytically studied that the flow arrival rate, number of required resources and associated cost have mutual dependencies that affects controller’s response time. We showed that effective flow-balancing strategies resulting in resources minimization, cost savings, and QoS improvements. We revealed that controller’s high service capability is always better than deploying multiple controllers with low service rate.

6.2 Future Work

Furthermore, this study fascinates many other network directions. Firstly, we have, a different distribution for analyzing the model and analysis of further general theoretical statistics. Our next consideration is, this model can capture multiple data plane hopes significantly in network domain, so, it will become easy to deploy Jackson-feedback concept that facilitate extended model. On third, for more rigorous real time prototype need to compare the next model that should be established. In next direction for future, study will be proposed to analysis M/M/1/c and M/M/m/c methodology that can be more representative in Distributed domain. Furthermore, we are focusing to deploy our work in SDN network in real time strategy, later it can modify and utilize in cloud based SDN network. Our work will attract many areas to focus SDN in their research work and try deploying it in network environment.

References

[1] G. Li, M. Dong, K. Ota, J. Wu, J. Li and T. Ye, "Deep Packet Inspection Based Application-Aware Traffic Control for Software Defined Networks", *2016 IEEE Global Communications Conference (GLOBECOM)*, 2016.

[2] W. Han, M. Dong, K. Ota, J. Wu, J. Li and G. Li, "SD-OPTS: Software-Defined On-Path Time Synchronization for Information-Centric Smart Grid", GLOBECOM 2017 - 2017 IEEE Global Communications Conference, 2017.

[3] A. Sallahi and M. St-Hilaire, "Optimal Model for the Controller Placement Problem in Software Defined Networks", *IEEE Communications Letters*, vol. 19, no. 1, pp. 30-33, 2015.

[4]D. Kaur, G. Aujla, N. Kumar, A. Zomaya, c. Perera and R. Ranjan, "Tensor-based Big Data Management Scheme for Dimensionality Reduction Problem in Smart Grid Systems: SDN Perspective", *IEEE Transactions on Knowledge and Data Engineering*, pp. 1-1, 2018.

[5] J. Wu, M. Dong, K. Ota, J. Li and Z. Guan, "Big Data Analysis-Based Secure Cluster Management for Optimized Control Plane in Software-Defined Networks", *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 27-38, 2018.

[6] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel and M. Hoffmann, "Heuristic Approaches to the Controller Placement Problem in Large Scale SDN Networks", *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 4-17, 2015.

[7] K. Sood, S. Yu, Y. Xiang and S. Peng, "Control layer resource management in SDN-IoT networks using multi-objective constraint", *2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)*, 2016.

[8] Z. Jiao, H. Ding, M. Dang, R. Tian, and B. Zhang, “Predictive big data collection in vehicular networks: A software defined networking-based approach,” in Proc. IEEE Glob. Commun. Conf. (GLOBECOM), Washington, DC, USA, 2016, pp. 1–6.

[9] L. Kuang, L. Yang, X. Wang, P. Wang and Y. Zhao, "A tensor-based big data model for QoS improvement in software defined networks", *IEEE Network*, vol. 30, no. 1, pp. 30-35, 2016.

[10] K. Sideris, R. Nejabati and D. Simeonidou, "Seer: Empowering Software Defined Networking with Data Analytics", *2016 15th International Conference on Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS)*, 2016.

[11] S. Zhao and D. Medhi, "Application-Aware Network Design for Hadoop MapReduce Optimization Using Software-Defined Networking", IEEE Transactions on Network and Service Management, vol. 14, no. 4, pp. 804-816, 2017.

[12] H. Hu, Y. Wen, Y. Gao, T. Chua and X. Li, "Toward an SDN-enabled big data platform for social TV analytics", IEEE Network, vol. 29, no. 5, pp. 43-49, 2015.

[13] Clemm, A., et al." DNA: An SDN framework for distributed network analytics," Integrated Network Management (IM), IFIP/IEEE International Symposium on. IEEE, 2015.

[14] G. Wang, T. E. Ng, and A. Shaikh, ''Programming your network at run-time for big data pplications,'' in Proc. 1st Workshop Hot Topics Softw. Defined Netw., 2012, pp. 103–108.

[15] P. Qin, B. Dai, B. Huang and G. Xu, "Bandwidth-Aware Scheduling with SDN in Hadoop: A New Trend for Big Data", IEEE Systems Journal, vol. 11, no. 4, pp. 2337-2344, 2017.

[16] M. Neves, C. Rose, K. Katrinis and H. Franke, "Pythia: Faster Big Data in Motion through Predictive Software-Defined Network Optimization at Runtime", *2014 IEEE 28th International Parallel and Distributed Processing Symposium*, 2014.

[17]"AWESoME: Big Data for Automatic Web Service Management in SDN", IEEE Transactions on Network and Service Management, vol. 15, no. 1, pp. 13-26, 2018

[18]X. Li, J. Yan and H. Ren, "Software defined traffic engineering for improving Quality of Service", *China Communications*, vol. 14, no. 10, pp. 12-25, 2017.

[19]C. Wang, G. Zhang, H. Chen and H. Xu, "An ACO-based elephant and mice flow scheduling system in SDN", *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)(*, 2017.

[20]Z. Liu, D. Gao, Y. Liu and H. Zhang, "An Enhanced Scheduling Mechanism for Elephant Flows in SDN-Based Data Center", *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, 2016.

[21]A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: scaling flow management for highperformance networks", ACM SIGCOMM Comp. Commun. Rev., vol. 41, no. 4, pp. 254–265, 2011.

[22]A. Curtis, W. Kim and P. Yalagandula, "Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection", *2011 Proceedings IEEE INFOCOM*, 2011.

[23]M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks", *Proc. 7th USENIX Conf. Netw. Syst. Design Implement.*, pp. 19, 2010

[24]F. Benamrane, M. Mamoun and R. Benaini, "New method for controller-to-controller communication in distributed SDN architecture", *International Journal of Communication Networks and Distributed Systems*, vol. 19, no. 3, p. 357, 2017.

[25]F. Benamrane, M. Ben mamoun and R. Benaini, "An East-West interface for distributed SDN control plane: Implementation and evaluation", *Computers & Electrical Engineering*, vol. 57, pp. 162-175, 2017.

[26]H. Yu, K. Li, H. Qi, W. Li and X. Tao, "Zebra: An East-West Control Framework for SDN Controllers", *2015 44th International Conference on Parallel Processing*, 2015.

[27]P. Lin, J. Bi, S. Wolff, Y. Wang, A. Xu, Z. Chen, H. Hu and Y. Lin, "A west-east bridge based SDN inter-domain testbed", *IEEE Communications Magazine*, vol. 53, no. 2, pp. 190-197, 2015.

[28]K. Phemius, M. Bouet and J. Leguay, "DISCO: Distributed multi-domain SDN controllers", *2014 IEEE Network Operations and Management Symposium (NOMS)*, 2014.

[29]H. Yin et al., ‘‘SDNi: A message exchange protocol for software defined networks (SDNS) across multiple domains,’’ Internet Engineering Task Force, Internet Draft, Jun. 2012. [Online]. Available: http://tools. ietf.org/id/draft-yin-sdn-sdni-00.txt.

[30]D. Gupta ,R. Jahan, Inter-sdn controller communication: Using border gateway protocol. White Paper by Tata Consultancy Services (TCS), 2014.

[31]K. Sood, S. Yu, Y. Xiang and H. Cheng, "A General QoS Aware Flow-Balancing and Resource Management Scheme in Distributed Software-Defined Networks", *IEEE Access*, vol. 4, pp. 7176-7185, 2016.

[32] K. Sood and Y. Xiang, "The controller placement problem or the controller selection problem?", *Journal of Communications and Information Networks*, vol. 2, no. 3, pp. 1-9, 2017.

[33]A. Fahmin, Y. Lai, M. Hossain, Y. Lin and D. Saha, "Performance Modeling of SDN with NFV under or aside the Controller", *2017 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, 2017.

[34] W. Miao, G. Min, Y. Wu, H. Wang, and J. Hu, “Performance modelling and analysis of software-defined networking under bursty multimedia traffic,” ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), vol. 12, no. 5s, pp. 77:1–19, 2016

[35]N. Beigi-Mohammadi, H. Khazaei, M. Shtern, C. Barna and M. Litoiu, "On Efficiency and Scalability of Software-Defined Infrastructure for Adaptive Applications", *2016 IEEE International Conference on Autonomic Computing (ICAC)*, 2016.

[36]Y. Goto, H. Masuyama, B. Ng, W. Seah and Y. Takahashi, "Queueing Analysis of Software Defined Network with Realistic OpenFlow–Based Switch Model", *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2016.

[37]K. Sood and Y. Xiang, "The controller placement problem or the controller selection problem?", Journal of Communications and Information Networks, vol. 2, no. 3, pp. 1-9, 2017.

[38]B. Lantz, B. Heller, and N. McKeown, “A network in a laptop: rapid prototyping for software-defined networks”, In Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks, 2010.

[39] “The Openflow switch,” 2014. http://www.open-networking.org.

[40] M. Liu, W. Dou, S. Yu, and Z. Zhang, “A Decentralized Cloud Firewall Framework with Resources Provisioning Cost Optimization,” in IEEE Trans. on Parallel and Dist. Syst., vol. 26, no. 3, pp. 621-631, 2015.

[41] S. Yu, Y. Tian, S. Guo and D. O. Wu, “Can We Beat DDoS Attacks in Clouds?,” in IEEE Transactions on Parallel and Distributed Systems”, vol. 25, no. 9, pp. 2245-2254, Sept. 2014.

[42] L. Yao, P. Hong, W. Zhou, “Evaluating the Controller Capacity in Software Defined Networking,” in Computer Communication and Networks (ICCCN), 23rd International Conference on, 2014.

[43] L. Kleinrock, Queueing System Wiley Interscience, 1975, Vol. I: Theory.

[44] J. Hu, C. Lin, X. Li, J. Huang, “Scalability of Control Planes for Software Defined Networks: Modeling and Evaluation,” Quality of Service (IWQoS), IEEE 22nd International Sym. of, pp. 147-152, 2014.

[45] K. Mahmood, A. Chilwan, O. Osterbo and M. Jarschel, “Modelling of OpenFlow-based software-defined networks: the multiple node case,” in IET Networks, vol. 4, no. 5, pp. 278-284, 2015.

[46] L. Long, F. Binzhang, and C. mingyu ., “Nimble: A fast flow scheduling strategy for openflow networks.,” Journal of Com- puter Science, vol. 38, pp. 1056–1068, 2015.

[47] A. Sallahi and M. St-Hilaire, "Optimal Model for the Controller Placement Problem in Software Defined Networks", IEEE Communications Letters, vol. 19, no. 1, pp. 30-33, 2015.

[48] K. Sood and Y. Xiang, "The controller placement problem or the controller selection problem?", Journal of Communications and Information Networks, vol. 2, no. 3, pp. 1-9, 2017.

[49] J. Liu, J. Li, G. Shou, Y. Hu, Z. Guo and W. Dai, "SDN based load balancing mechanism for elephant flow in data center networks," 2014 International Symposium on Wireless Personal Multimedia Communications (WPMC), Sydney, NSW, 2014, pp. 486-490.