SERVER AND NETWORK LOAD BALANCING

IN SDN CONTENT DELIVERY DATACENTER NETWORK

By

Gaurav

Bachelor of Engineering in Computer Science, M.D.U. Rohtak India, June 2010

A thesis presented to Ryerson University

in partial fulfillment of the requirements for the degree of

Master of Applied Science

in the department of Computer Networks

Faculty of Electrical and Applied Science

Toronto, Ontario, Canada 2015

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

Abstract

Server and Network Load Balancing

Gaurav
Master of Applied Science
in Dept. of Computer Networks
Faculty of Electrical and Architectural Science
Ryerson University, Toronto, 2015

Software Defined Networking (SDN), is an emerging networking technology. This thesis aims to develop a new Server and Network Load balancing scheme in content delivery datacenters using SDN-based architecture. The scheme, called Server and Network Load Balancing (SNLB), tends to distribute the traffic load more evenly across the network. The SNLB achieves even distribution of flows on the links and servers by utilizing real-time network statistics. Furthermore, SNLB classifies the network flows into mice (flows with small bandwidth) and elephant (flows with large bandwidth) flows and performs load balancing on these two classes of flows separately. A detailed comparison of SNLB with Global first fit, Round robin and Load based balancing is presented. Other objectives achieved in this thesis are the designs of overload traffic handling technique and Fault tolerance method. The overload traffic handling technique activates and de-activates servers according to the traffic load; the fault tolerance method can reduce the impact on network performance during the network fault.

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor Dr. Ngok-Wa Ma for the continuous support of my Master's study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

I thank my fellow mate researchers: Akinniyi Ojo, Liyi Huang, Maryam Kia, for the stimulating discussions.

I would also like to thank the Computer Networks department and the Yeats School of Graduate Studies at Ryerson University for providing the opportunity to complete my Master's degree and support of Ryerson graduate scholarship and Graduate assistantship.

A special thanks to my family, who constantly supported and encouraged me throughout the course of my studies.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

SDN	Software Design Network
SNLB	Server and Network Load Balancing
CAPEX	Capital Expenditure
OPEX	Operating Expenditure
PPS	Packet Per Second
API	Application Programming Interface
DCN	Datacenter Network
ECMP	Equal Cost Multi Path
GFF	Global First Fit
OEM	Original Equipment Manufacturer
ONF	Open Networking Foundation
API	Application Programming Interface
NFV	Network Function Virtualization
SNMP	Simple Network Management Protocol
SLA	Service Level Agreement
VAS	Value Added Services
IT	Information Technology
VA	Virtual Appliance
OF	OpenFlow
NOF	Non OpenFlow
DPI	Deep Packet Inspection
HA	High Availability
QOS	Quality Of Service
DSCP	Differentiated Service Code Point
SSL	Secure Socket Layer
VM	Virtual Machine
KVM	Kernel-based Virtual Machine
SPAN	Switch Port Analyzer

RSPAN	Remote SPAN
CLI	Command Line Interface
LACP	Local Control and Accounting Plan
VETH	Virtual Ethernet cables
ISP	Internet Service Provider
ECMP	Equal Cost Multi Path
CRC	Cyclic Redundancy Check
TCP	Transport Control Protocol
UDP	User Datagram Protocol
RTT	Round Trip Time
DHT	Distributed Hash Table
MAC	Media Access Control
NIC	Network Interface card
LLDP	Link Layer Discovery Protocol
L2	Layer 2
RAM	Random Access Memory
OS	Operating System
MVIP	Mice Virtual Internet Protocol
EVIP	Elephant Virtual Internet Protocol
BGP	Border Gateway Protocol
DNS	Domain Name Server

Chapter 1

Introduction

1.1. Overview

Software Defined Networking (SDN) has brought a new paradigm in networking. It decouples the distributed control plane from the data plane and moves the control plane to the centralized controller. Thus, the controller has a complete view of the network topology plus the full control of network resources. Together with the controller's programmability, SDN offers efficient and flexible ways to deliver networking functions.

Network operators of large organizations and cloud providers are facing big challenges of building enormous datacenters to previously unimaginable size that support thousands of switches and servers. As datacenters and their applications continue to grow, utilizing network resources in order to improve datacenter performance presents a particular challenge. Datacenter workload varies over both time and space. As a result, static resource allocation is inefficient. In recent datacenter, designs rely on the path multiplicity to achieve horizontal scaling of hosts [24, 25, 27, 28]. For these reasons, datacenter topologies are very different from typical enterprise networks. In datacenter networks, switches are interconnected with many paths to provide redundancy. In order to utilize the bandwidth efficiently datacenter networks use TRILL, the IETF data forwarding protocol standard, or Fabric Path, the Cisco proprietary protocol, to implement Equal Cost Multiple paths (ECMP). In ECMP, traffic is load balanced across multiple equal cost paths by assign flows to different paths. The path selection is usually

based on hashing. The problem with hashing or its variation, round robin scheme, does not guarantee true load balancing. This results in undesirable situation that some links are oversubscribed while other links are undersubscribed.

In the recent years, SDN has been increasingly adopted in datacenter networks. SDN provides new solution to the old networking issues. Because of the centralized view of the network, SDN provides a more efficient platform to implement ECMP load balancing. In this thesis, we propose a load balancing scheme by utilizing the characteristic of SDN. The implementation is lightweight but introduces more advanced features than the existing protocols.

In the content delivery datacenter, there are many replicate servers that can deliver the same content. The load balancing issue in such a case will not be completely addressed without addressing server load balancing. In non-SDN networks, server load balancing usually uses either round robin or random selection. Again, by taking advantages of the SDN characteristics, we propose a server load balancing scheme that is simple but efficient.

1.2. Brief Introduction of the Proposed Schemes

In this thesis, a Server and Network Load balancing (SNLB) scheme for content delivery datacenter network is proposed based on the SDN architecture. The SNLB scheme will run at the controller. To better load balance the content request we categorized the flows as Mice[19] and Elephant[14]. Mice flow is a flow that requires less than 10% of the link bandwidth and Elephant flow require 10% or more. The SNLB scheme in content delivery datacenter network will provide efficient utilization of the

links by distributing the flows among all available servers and load balancing the flows evenly on the paths connecting to the selected server.

The main feature of the proposed scheme for the content delivery datacenter network are as follows:

- The use of Server and Path selection algorithm results in efficient utilization of the bandwidth and server resources.
- Flow and port statistics of the traffic, and the equal cost paths to all the active servers paths are stored in the controller's database to reduce complexity.
- Overload conditions are well addressed by using available resources smartly.
- Fault tolerance is improved by keeping the impact in network to the minimum.

1.3. Solution Validation

Simulation results show that the proposed scheme is more efficient than the traditional global first fit, load based and round robin based ECMP schemes. Major performance parameters such as network throughput, latency, jitter and network throughput for server with even spreading are used for the evaluation of the proposed scheme. The simulation results also show the effectiveness of the proposed overload handling and fault tolerance schemes.

1.4. Thesis Outline

The rest of the thesis is organized as follows:

Chapter 2 gives a brief introduction of SDN and its architecture, SDN as framework and OpenFlow protocol concepts including Open Switch architecture. Some of the current load balancing algorithms are investigated.

In Chapter 3, A Server and Network Load balancing scheme is proposed. A thorough description of the scheme is presented. This chapter also introduces the concepts of Server activation for the overload handling and Fault tolerance scheme.

Chapter 4 describes the technologies and architecture used in the implementation phase. It also provides details on performance measurements, simulation setup, results and observations.

Chapter 5 draws the final remarks and conclusion. Possible future works also discussed.

Chapter 2

Background

In this chapter, we will introduce the Software defined networking architecture. SDN framework and built-in components are discussed. This chapter will also cover OpenFlow protocol. Software Defined Networking in content delivery datacenter network is discussed. Finally, the chapter will review some of the current load balancing schemes used in datacenter networks.

2.1. Software Defined Networking

Software Defined Networking (SDN) is an emerging network architecture where network control is decoupled from forwarding plane and it is programmable. By centralizing the control plane, it expanded the possibility of network intelligence by having complete network visibility. The network infrastructure can be smartly utilized and performance of network have great opportunity for optimization.

The OpenFlow protocol is a popular southbound protocol used for the communications between the controllers and the network elements. In SDN, the network devices only implement the data plane. They accept instructions from the SDN controller[12] through the OpenFlow and other southbound protocols for data forwarding. This reduces complexity of the network devices as forwarding devices no longer required to understand and implement the control plane. The indirect benefit of SDN is that network devices will be correspondingly cheaper.

2.1.1. SDN Architecture

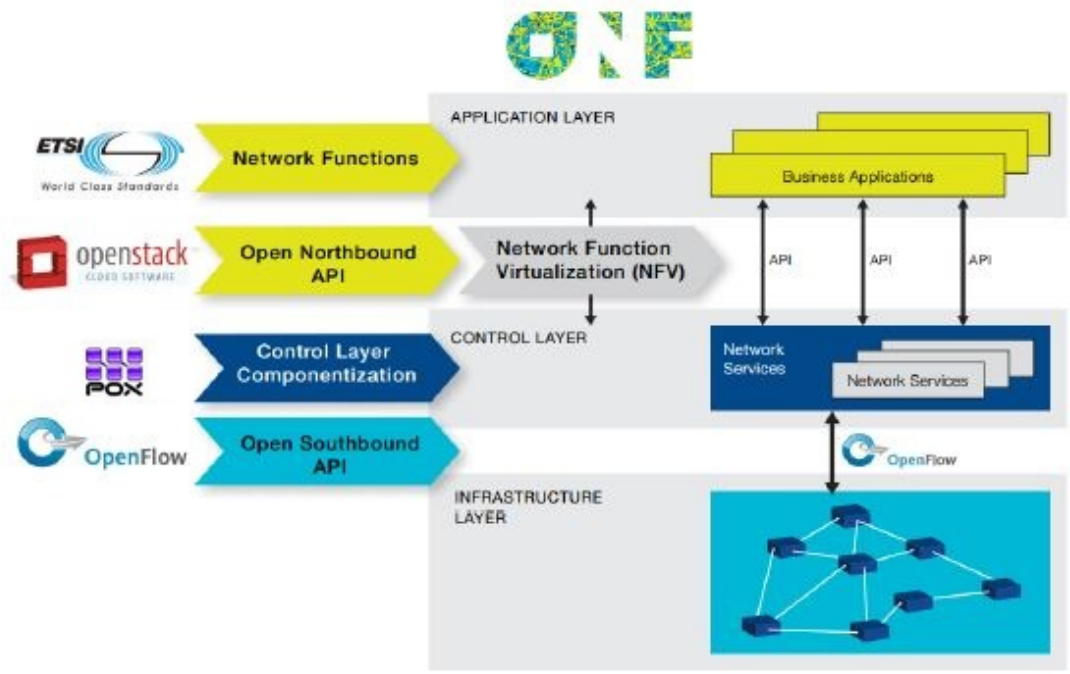

Figure 2.1: The Open SDN architecture, source [13]

Figure 2.1 depicts a logical view of the SDN architecture which consists of three main layers: infrastructure, control and application layer. Network intelligence is (logically) centralized in software-based SDN controller, which maintains a global view of the network.

2.2. SDN Framework

SDN Framework comprises 3 major components:

- SDN compliant Forwarding Hardware – Fast Path
- SDN Controller – Control Plane (Middleware)
- SDN Applications

Controller will connect with one or more SDN compliant switches to manage the data forwarding. Based on the information carried by the new flow, flow forwarding decisions are made by the controller and these instruction are sent to the switch using southbound interface protocol which is OpenFlow in our thesis. These instructions are installed as actions for the flow in switching infrastructure. Virtual appliances(VAs) can send commands to the controller through the Northbound API to instruct the controller on how to implement specific network functions which could be utilized for the inputs to make forwarding decisions for the flow.

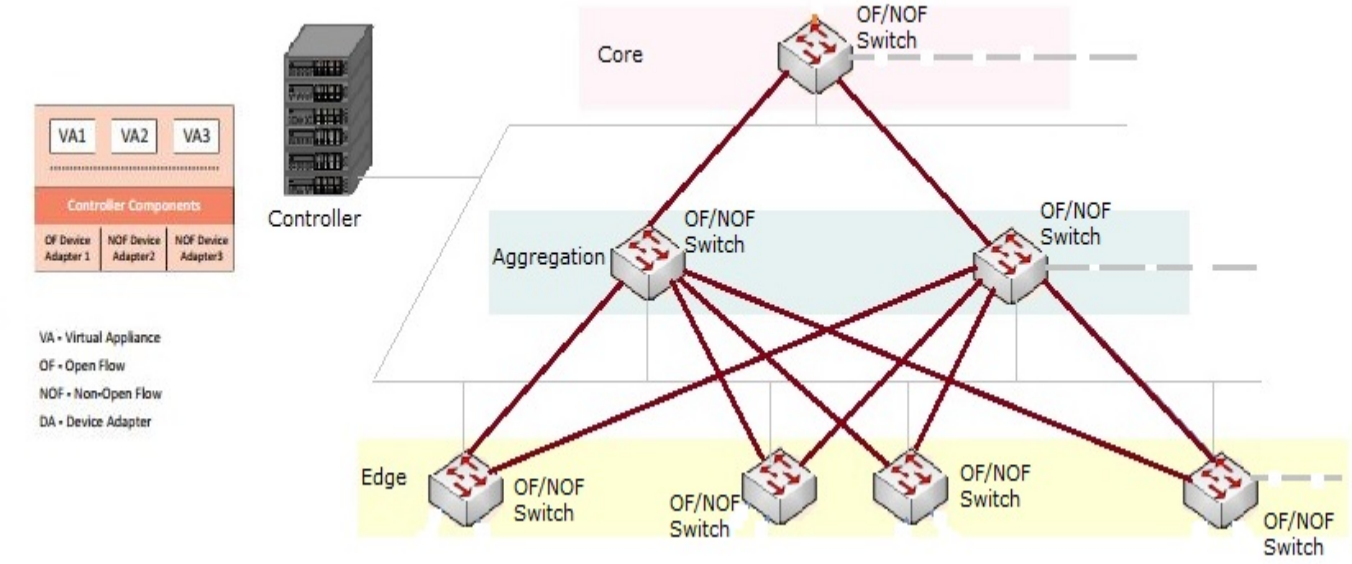

Figure 2.2: SDN Framework

SDN controller can communicate with the switches from any vendor by using device specific adaptors as shown in figure 2.2. These adaptors are used to enable SDN on different devices. This makes it possible for the network operators and enterprises to orchestrate the resources in heterogeneous network. This is one of the many potential benefits of SDN.

2.3. Components in SDN Controller

Controller development platforms in SDN offers library of built-in components which can be used to design new features. The following description is based on the architecture of the POX controller which is used in this thesis.

2.3.1. Network Discovery Component

After starting the controller and system components, discovery component attempts to discover network topology. To discover the connectivity between OpenFlow switches in the network topology, discovery component sends LLDP packets. The response is received as openflow_PacketIn event. Using the event handler for this event, the LLDP packet is received and then processed. After discovering all the switches and links, the network topology is generated as graph with nodes as switches and edges as vertices.

2.3.2. L 2 Multi Component

The graph of the network generated by discovery component is used by l2_multi component. It runs the multi path Floyd Warshalls algorithm which picks multiple least cost paths to store in database. After the computation, the information of all the equal and the shortest paths are stored in database to be accessed by the path selection scheme.

2.3.3. Spanning Tree Component

Spanning tree component available to work with the learning component. For the

purpose of redundancy, the network topology usually have loops. This component overlay a spanning tree on the physical topology. The flood traffic will follow the spanning tree, like the Ethernet. The known unicast traffic, however, follows the shortest path, unlike the Ethernet.

2.4. OpenFlow Protocol

In SDN standards, OpenFlow Protocol[21] is defined as one of the communication protocols which enables the SDN/OpenFlow controller to interact with the switching infrastructure from multi vendor. Network can better adapt to the changing business requirements by making adjustments to the network using OpenFlow protocol. It is utilized by SDN controller to install flows to the switch/router for implementing network functions such as data forwarding partition traffic, control flows, etc.

2.4.1. OpenFlow Structure

OpenFlow provides the programming interface for better management of the forwarding plane. It interacts with the switching infrastructure using secure channel to provide taking efficient forwarding decisions. These decisions are given in the form of instruction to the switching infrastructure to add/modify flow tables/entries.

2.4.2. OpenFlow Switch

OpenFlow switch is a switch that can communicate with the SDN controller using OpenFlow protocol. Forwarding in OpenFlow switch is done using Flow tables, Group

table and a Metering table. Every flow table in a switch is designed to have a set of flow entries where each flow have match fields, counters, and a set of instructions that are applied on matching packets. Figure 2.3 illustrates matching fields in OpenFlow Packet.

Figure 2.3: OpenFlow Packet Match Fields

Figure 2.4 illustrates the OpenFlow Switch[15]. The switch consists of the four main modules:

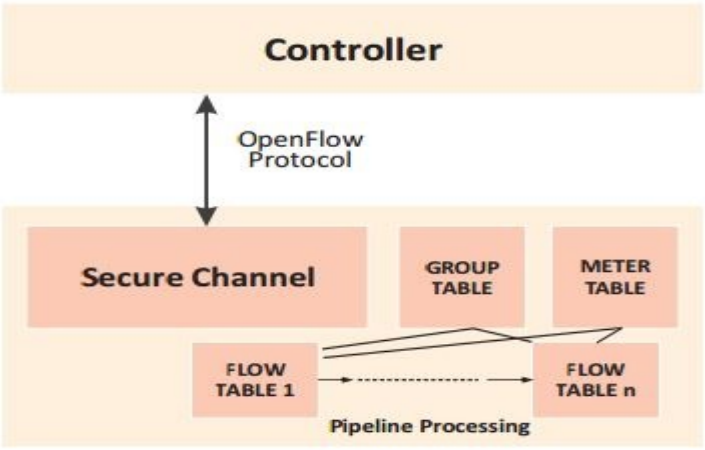

Figure 2.4: OpenFlow Switch Modules

1. Secure Channel:To send commands and packets between controller and the Open Switch, OpenFlow Channel running over Secure Sockets Layer (SSL) is utilized. Due to this reason, OpenFlow Channel is also referred as Secure Channel;

2. A Flow Table which contains match fields, counters and a set of instruction to apply on the matching flow. To find the corresponding flow rule, tables are looked up from Table0 to TableN. If no flow rule exists then the packet is sent to the controller for decision on the action.

3. Group Table contains group entries and each entry has a list of actions. These actions are applied to packets which are sent to the group entries [15].

4. A Meter Table consists of meter entries which defines per-flow meters. Various simple QoS can be enabled in OpenFlow by utilizing Per-flow meters, for example rate-limiting, and can be combined with per-port queues to implement complex QoS frameworks, such as DiffServ.

Communication between the controller and switch when a new flow arrives is illustrated in Figure 2.5.

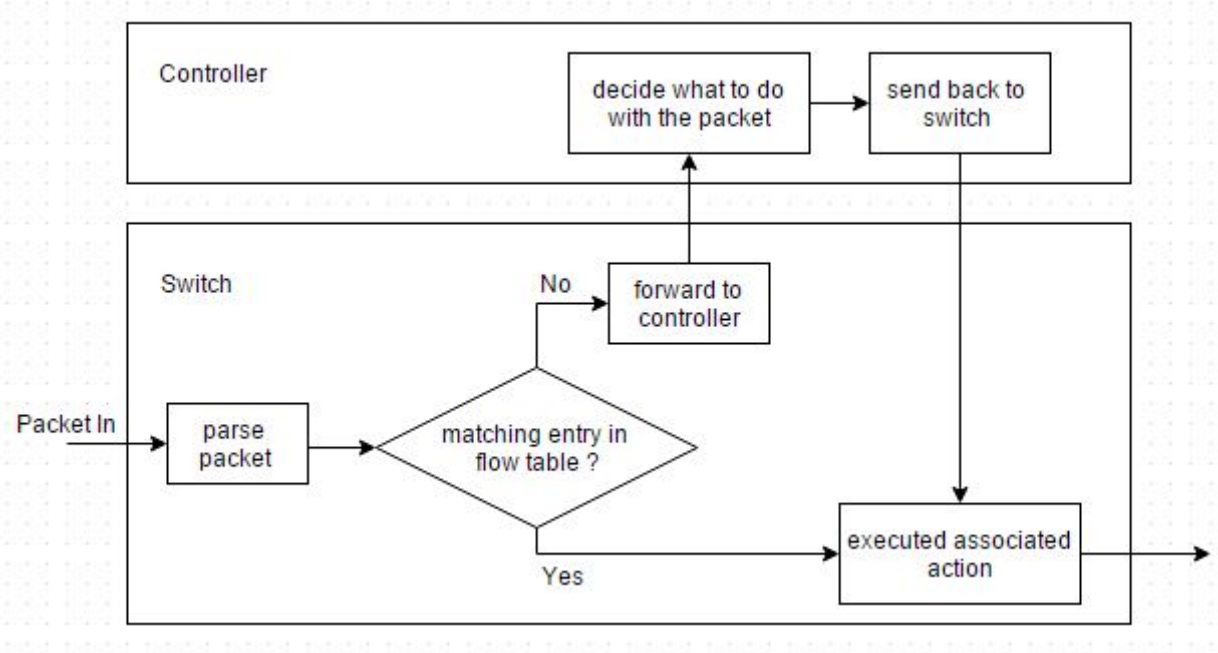

Figure 2.5: OpenFlow Controller and Switch

2.5. SDN in Content Delivery Datacenter Network

Content delivery datacenters are serving a large fraction of the internet content which includes media files, live streaming, on-demand steaming, social networks, to mention a few. Providing these services with high availability and high performance requires multiple replicated servers to service the request for the same content. Using SDN, these

datacenter can have improved page load times, better availability, increased network bandwidth to service the content requested.

In this thesis, the content request is categorized in two types of flows: Mice and Elephant. The difference between them is based on bandwidth requirement. The mice flows request less bandwidth than that of the elephant flows. We will give a formal definitions of them in the next chapter. The content requests using virtual IP address of the content server. This virtual IP address represents many replicated servers which are available to serve the content request. The virtual IP address will be translated to the IP address of one of the replicated servers. This server selection is based on several performance criteria. Present load balancing techniques in content delivery datacenter networks facing limitations of uneven path selection and poor utilization of the available content delivery datacenter resources. In this thesis, we proposed a scheme that will provide both server and bandwidth load balancing.

2.6. Load Balancing Techniques in Content Delivery DCN

The performance and the scalability of the datacenter can be improved by distributing the load evenly across network links and servers, or other resources. With efficient load balancing we can improve the throughput[3], redundant connectivity [4] or congestion [5].

Different forms of content delivery load balancing are deployed at various layers of the protocol stack. At the datalink layer, frames can be distributed over parallel links between two devices[6]. At the application layer, requests can be spread on a pool of servers.

To the best of our knowledge no report is found in the literature combining Server and Network Load balancing in DCNs resulting in efficient link and server utilization.

2.6.1. Equal Cost Multi Path (ECMP)

At present, the mostly deployed scheme is Equal-Cost Multi-Path (ECMP) [3]. ECMP is both a path selection scheme and a server load distribution mechanism. For the path diversity, it uses loop-free paths for forwarding. ECMP may both increase the network capacity and improve the reaction of the control plane to failures [4].

In Round-robin scheme[24] based ECMP, distribution of the flows are in sequential order among the available equal cost paths. This scheme does not guarantee even flow distribution.

In Hash based ECMP scheme which is the most deployed next-hop selection method, the path selection is based on the hash value derived from the fields of the packet header[8,3]. These fields are usually the source and destination IP addresses, the protocol number and the source and destination ports, i.e. the 5-tuple. The computed hash can then be used in various ways to select a next hop. The simplest and most deployed method is called Modulo-N. If there are N available next hops, then the hash is divided by N and the remainder is used as an identifier of the next hop of the selected. Implementing a hash function ensures a somewhat even distribution of the next-hop selection [5].

2.6.2. Global First Fit(GFF)

In GFF scheme, after receiving a new flow request, the scheduler searches linearly all the available paths in order to find the one which can accommodate the bandwidth

requirement of this new flow. After finding such a path, the flow is placed directly to it. This is a first fit algorithm, flow is greedily assigned to the first path which is fulfilling the requirement. Global First Fit does not distribute flows evenly across all equal-cost paths.

2.6.3. Related Work on Load Balancing

The possibility of improvement in the performance of content delivery datacenter by utilizing the network resources intelligently have attracted the interest of many researchers. There are number of schemes presented, some are trying to utilize network links and servers, others are scaling the servers for network over subscription. But the load balancing techniques have utilized mostly round robin or hash based ECMP scheme or Global first fit scheme. Below we have discussed some of them.

In Hedera[1], the algorithm designed to load balance large flows (require more than 10% of the link bandwidth) outperforms ECMP and Global first fit in the efficient bandwidth utilization of the links. But it has limitations such as the short flows are load balanced using hash function which results in hash collision that leads to uneven resource allocation. Also it does not load balance the flows at the time of its arrival, during the service of flow if it crosses threshold of the bandwidth requirement which is 10% of the least bandwidth link in the path, it then load balance it as per algorithm. This result in underutilized network resources. And its implementation is expensive in the datacenter networks.

In Mahout[7], the flows originating from the end host is monitored by shim layer in OS. When the elephant flow is detected, it marks the subsequent packets of that flow

using an in-band signaling mechanism. The switches in the network are configured to forward these marked packets to the Mahout controller. Then the controller manages these elephant flows using first fit algorithm and load balance them among multiple equal paths. The key limitations are the elephant flow detection time which is long after the server is allocated and servicing that flow, no mice flow load balancing and the use of first fit algorithm for elephant flow load balancing. As a result, this scheme is not very effective to improve utilization of network resources.

In "OpenFlow based load balancing gone wild"[17], the load balancing scheme is proposed which maps blocks of source IP addresses to servers in order to forward client requests directly with minimal intervention by the controller. To implement this, changing wildcard rules are installed and rules are updated timely. Some limitation of this scheme are traffic is not well spread between servers and paths as it relies completely on wildcard forwarding, the scheme tries to divide client traffic based on client ip addresses but is not quite successful and some of the links and servers remain underutilized.

In Plug-n-serve[27], the algorithm for load balancing among the paths and servers is proposed. The statistics from the switches has been utilized for the decision of load balancing. But no clear methodology is given on the algorithm used to select the path and the server. No clear vision is provided which statistics are used and the improvement as a result.

2.7. Chapter Summary

In this chapter, we have introduced Software Defined Networking and its

architecture. We discussed how SDN can bring improvement in content delivery datacenter networks. Some limitation of presently implemented load balancing schemes has been discussed. The uneven server and path assignments to service the request has been seen as critical limitation in the content delivery datacenter network. Performance of the network can be improved by using SDN by eliminating the limitation of uneven server and path load balancing.

Chapter 3

Server and Network Load Balancing Design

The objective of this thesis is to design a Server and Network Load balancing scheme (SNLB) running at a Controller which can evenly distribute the incoming traffic over Servers and Paths in order to maximize the performance of fat-tree Datacenter network. SNLB adopts the dynamic flow scheduling technique in the SDN environment. In this chapter, we first address the challenges of the new architecture and presents the proposed solutions for fat-tree datacenter network. We then investigate possible enhancements in terms of flexibility, complexity as well as the performances.

3.1. Proposed SNLB Scheme

Figure 3.1 describes the SNLB structure at the system level. The figure shows different components and modules of SNLB system. These components and modules are introduced below. Note that the Network discovery and l2-multi components were introduced in the last chapter. For the sake of completeness, they are also included in the following description.

3.1.1. Network Discovery Component

When the system starts, this is used to discover the nodes and links in network topology. Any change in network is discovered by this module and network convergence to update topology information is initialized.

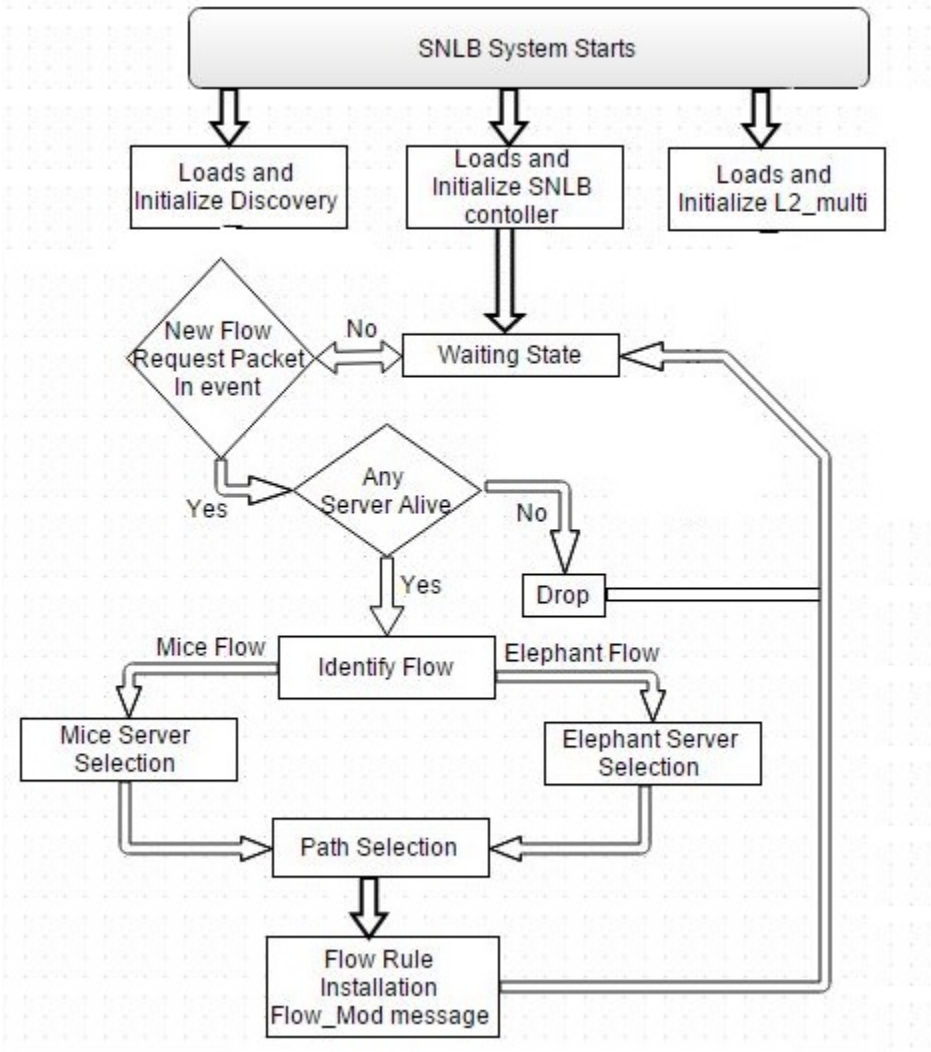

Figure 3.1: SNLB Structure

3.1.2. L2 Multi Component

This component is used for learning Ethernet addresses and computing the multiple least and equal cost paths between any pair of nodes. This information is afterwards utilized by Server and Path selection algorithm.

3.1.3. Flow Based Load Balancing

In SNLB scheme, we have categorized the external client flows coming to content delivery datacenter network into Mice flows and Elephant flows. Mice flow is the flow that requires less than 10% of the bandwidth of the link with the least capacity. This is usually the link between the access switch and the aggregation switch. Elephant flow, on the other hand, is the flow that requires 10% or more of the bandwidth of the link with the least capacity.

3.1.4. Switch Statistics Module

The statistics module plays an important role for decision making in server selection scheme and path selection scheme. Aggregation switches are queried periodically for the statistics. The module gathers port statistics and flow statistics of switches which provides information such as number of flows, destination address, source address, port number, transmitted bytes, received bytes, match fields etc. To access the statistics with

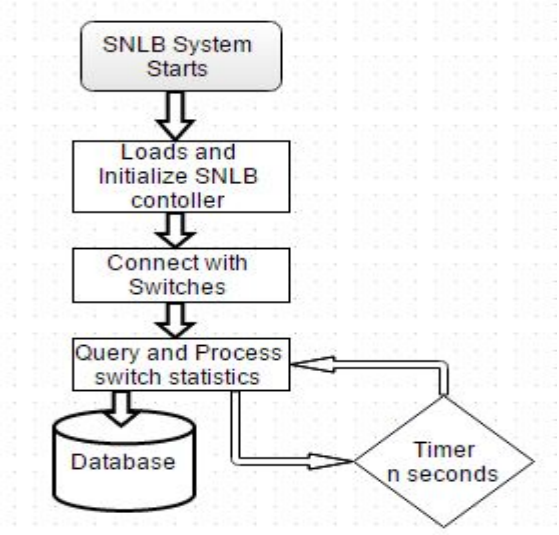

Figure 3.2:Switch Statistics Module Flowchart

least complexity this information is received from the switches and stored in the local database at the controller. The structure of this module is illustrated in Figure 3.2.

To maintain the accuracy of required statistics, statistics are updated periodically. Since our scheme classifies the flows as either Mice or Elephant flows hence the statistics of mice and elephant flows are stored separately.

3.1.5. Server Selection Module

The server selection module is used to select server and update the server flow status. For every server, the module maintains two counters, elephant flow counter and mice flow counter, to keep track of, respectively, the number of elephant and mice flows that are currently served by the server. When a new flow request is received, the controller first determines if the new flow is elephant or mice based on the virtual IP destination address. Then the module will select the server with the smallest elephant flow counter value or mice flow counter value depending on whether the new flow is elephant or mice, respectively. The following equation is used to select the server:

$$F(S_k) = \min_{i \in m} F(P_i) \tag{3.1}$$

where $F(S_k)$ is the number of flow on server k selected for the new flow and m is the number of the number of available servers.

In addition, if the new flow is elephant, then

$$F_{E,new} = F_{E,old} + 1 \tag{3.2}$$

where F_E is the elephant flow counter value of the selected server; similarly, if the new flow is mice, then

$$F_{M,new} = F_{M,old} + 1 \tag{3.3}$$

where F_M is the mice flow counter value of the selected server.

When a flow is terminated, caused by the idle time-out, the switch will inform the controller by sending an OpenFlow packet to the controller. The packet contains the statistics of the terminated flow such as flow duration, D_f, idle time-out duration, T_I, and the total number of transmitted bytes, B_F. The packet also gives enough information (The port number where the server attached) for the controller to identify the server. To determine if the terminated flow is elephant or mice, the server selection module calculates the data rate, R_D, of the terminated flow using the following equation based on the flow statistics:

$$R_D = B_F * 8 \,/\, (D_F - T_I) \tag{3.4}$$

If $R_D \geq 0.1$ * link capacity

then

$$F_{E,new} = F_{E,old} - 1 \tag{3.5}$$

Otherwise,

$$F_{M,new} = F_{M,old} - 1 \tag{3.6}$$

3.1.6. Path Selection Module

First, we define the traffic load of link l, $TL(l)$, as the number of flows traversing the link divided by the bandwidth of the link:

$$TL(l) = \text{number of flows} \,/\, B_l \tag{3.7}$$

Where B_l is the bandwidth of link l. Since we separate the flows into elephant and mice flows, there are two types of traffic loads associated with a link: elephant and mice traffic loads. Next we define the traffic load (elephant or mice) of the path as the maximum traffic load among all the links that constitute the path. Mathematically, it can be formulated as follows.

Let P_i be one of the equal-cost paths. P_i is described by a ordered set of links:

$$P_i = \{l_{i1}, l_{i2}, \ldots , l_{in}\} \tag{3.8}$$

where l_{ij} is the j^{th} link of path P_i and n is the number of links that constitute the path. For the typical datacenter fat-tree topology, n=3. Then traffic load of P_i, $TL(P_i)$, is defined as:

$$TL(P_i) = \max_{j \in n} TL(l_{ij}) \tag{3.9}$$

Thus, if a server is selected to serve the flow request and there are m equal cost paths to select to reach the server, P_i, $1 \leq i \leq m$, the path selection module will choose path k if path k has the minimum traffic load among all the equal cost paths:

$$TL(P_k) = \min_{i \in m} TL(P_i) \tag{3.10}$$

Once the path is chosen and the new flow is installed on the path, the traffic loads of the links of the path will be updated. Let path k be the selected path and

$$P_k = \{l_{k1}, l_{k2}, \ldots , l_{kn}\} \tag{3.11}$$

then

$$TL(l_{kj}) = \mathrm{TL}(l_{\mathrm{kj}}) + 1/B_{kj} , \qquad 1 \leq j \leq n \tag{3.12}$$

When a flow is terminated and removed from a path, the traffic load of all the links of the path will be updated according to the following simple procedure. If a flow is just removed from P_k, then:

$$TL(l_{kj}) = TL(l_{kj}) - 1/B_{kj}\,, \qquad 1 \le j \le n \tag{3.13}$$

We would like to emphasize that the description in this subsection is applied for both elephant and mice flows. The path selection processes and the updates of the traffic load of the links for mice and elephant flows, however, are performed separately.

3.2. Server Activation for Overload Traffic

In our proposed design, we implemented a server activation feature which will activate the available idle server to handle overload traffic conditions. When the total loads(load of mice plus elephant flows) of all the active servers reach the defined overload threshold value, SNLB scheme will trigger the system to activate one of the idle servers to handle new flows.

We query port statistics from the aggregation switches periodically(5 seconds in SNLB scheme). These statistics are used to track the current upstream load on the edge switches. Note: for the aggregation switches, the received traffic on the ports connected to edges switches is the upstream traffic of the edge switches.

Let L_{ov} be the defined overload load of the server and N be the number of active servers. When a new flow arrives, if the load of all the active servers has reached L_{ov} then a new server is activated. Mathematically, the condition for activating a new server is formulated as follows:

$$TL(S_i) \ge L_{ov}\,, \quad 1 \le i \le N \tag{3.14}$$

where S_i, $1 \le i \le N$, are the N active servers.

In server activation, the server selection from the list of idle servers is based on the load on edge switch where the idle server is connected. Specifically, the idle server connected at an edge switch with the least upstream link traffic load will be activated. Note that an upstream link of an edge switch is the link that connects the edge switch with the aggregation switch. On activation, the server is added to the list of available active servers where it will be selected to service new flows based on SNLB scheme server selection. The activated server is also added to the list of activated servers which is used to keep track on the order of servers activation. This list is utilized for the process of server deactivation. Figure 3.3 illustrates an example of the server activation process.

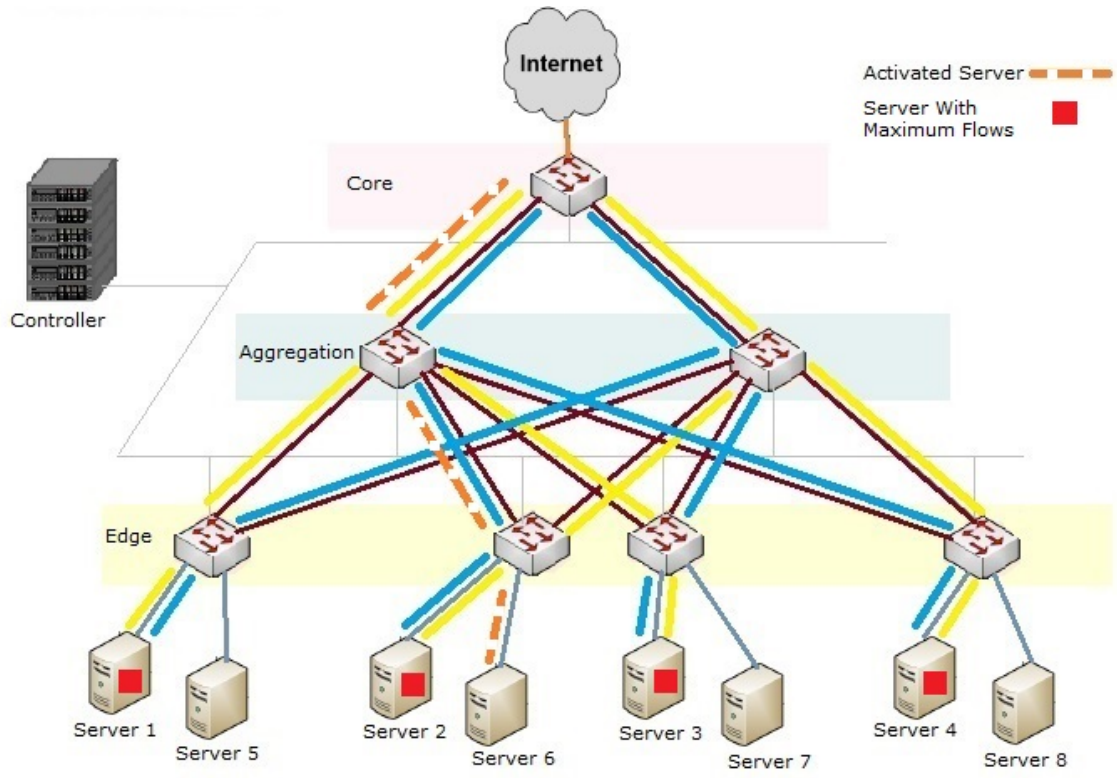

Figure 3.3: Server Activation

In this example, active servers 1, 2, 3 and 4 are in overload condition. Servers 5, 6, 7 and 8 are inactive. If a new flow arrives and if the edge switch that is connected to server 6 has the lowest upstream link traffic load, then server 6 will be activated. In what follows, we will describe two additional processes: one addresses server deactivation condition and the other describe how to select a server to de-activate.

Server deactivation condition:

The server deactivation process will de-activate an active server if the traffic loads of all active servers fall below pre-defined thresholds. Let define th_E and th_M be the thresholds defined for server de-activation. In order to satisfy both elephant and mice flows requirements, an server is de-activated only if all the active servers satisfies the following condition:

$$F_E < th_E \text{ and } F_M < th_M \tag{3.15}$$

How to choose a server to de-activate:

In server deactivation, the added server is removed from the list of available servers hence any new flow will not be allocated to the deactivated server. But the flows which are already assigned to this server will be serviced normally and eventually they will all terminate, leaving the server to the idle state.

To maintain the spread of flows all over the network, we deactivate the server in the last in first out(LIFO) fashion i.e. the server that is activated last will be de-activated first. If we deactivate any other server then it may cause a situation where there are two active servers at the same edge switch and no active server at some of the edge switch. This situation will degrade the performance of the network. During the deactivation process, if

the loads of all the active server that is being de-activated in the list of available active servers again reach the maximum load, the server will be reactivated.

The server activation in SNLB scheme will improve the performance in the event of high traffic load provided the network is capable of handling the overload. This scheme promises a better response time from servers and also utilizes available resources smartly under overload traffic conditions.

3.3. Server Fault Tolerance in SNLB

To handle possible server failure, a fault tolerance feature is introduced in SNLB. This scheme is designed to minimize the impact of server failure by choosing a replacement server locates closest to the failed server. The working of fault tolerance is as illustrated in figure 3.4.

After the failure of active server, an idle server will be activated to service the flows. Activating a new server may require the controller to re-establish all the flows associated with the faulty server. If there is an idle server available at the same edge switch this server will be activated in order to reduce the disruption of the load balancing pattern established by the SNLB scheme and the sever activation process. For example, referring to figure 3.4, originally, servers 1, 2 3 and 4 are active while servers 5, 6, 7 and 8 are inactive. Note that the 4 servers are spread out as much as possible to achieve a better load balancing condition. If server 3 crashes, the algorithm will try to activate a replacement. The best option is to activate server 7 because it shares the same edge switch with server 3 and the selection will maintain the desirable load balancing

condition. If the algorithm chose server 5 for example, more traffic would use the paths that connect the core and the edge switch that is connected to servers 1 and 5 while the paths lead to server 7 is underutilized.

If the server is failed due to the failure of the edge switch, the idle server on the same switch will also be unreachable. In this case, the new server will be activated at the edge switch whose upstream link has the least traffic load similar to the server activation process.

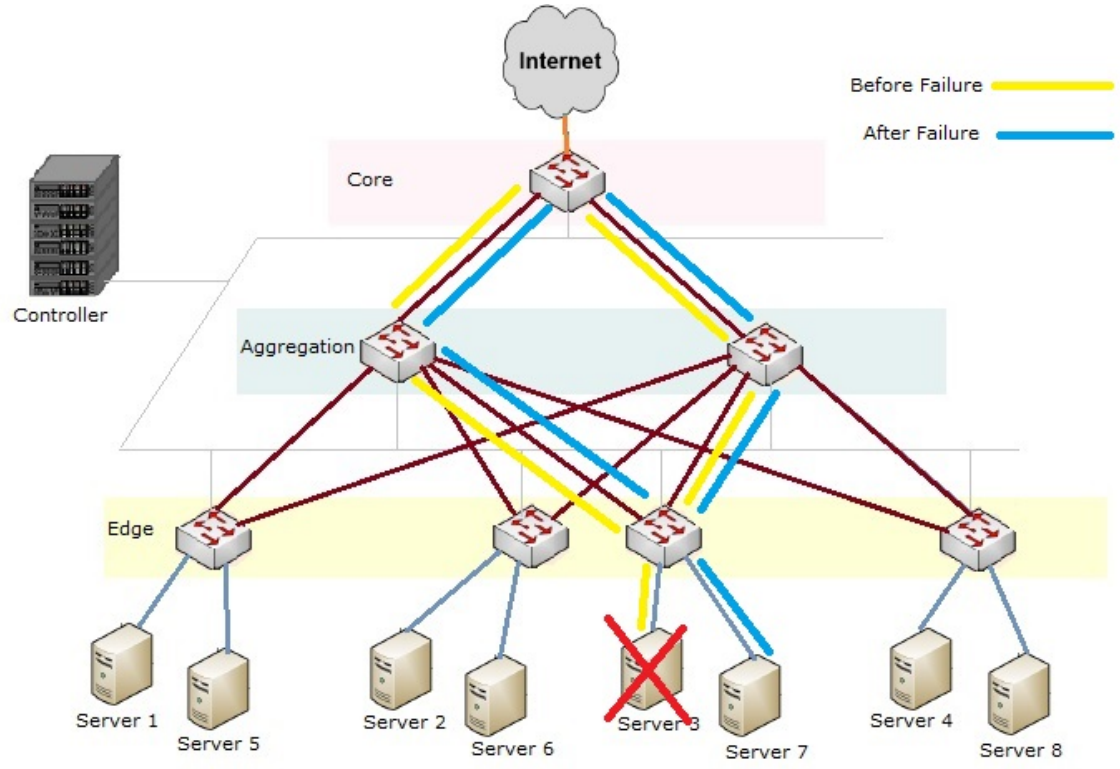

Figure 3.4: Fault Tolerance in SNLB

This fault tolerance technique in SNLB scheme will promise even flow distribution among available links in case of fault. Better flow spread in the network will be maintained even in the case of server failure. It will ensure better performance of the content delivery datacenter network.

Chapter 4

Implementation, Results and Performance Evaluation

To test the designed Server and Network Load Balancing scheme, we have used typical Datacenter network topology. The network is emulated using Mininet and the POX controller is used to implement the Load balancing scheme. Detailed operation of SNLB scheme and implementation scenarios are presented in this chapter.

This chapter also presents evaluation and validation of SNLB design using different scenarios tested on the typical datacenter network infrastructure. Additionally, a brief insight of the expected results is given in order to estimate the possible outcome.

4.1. Technologies and Simulation Tools

This section provides details about software and technologies used to implement our design.

4.1.1. System Capability

We have used machine with Processor: 2.6GHZ Turbo boost Intel Core I5 processor, Installed memory: 8GB RAM, System type: 64-bit operating system, x64-based processor and OS: Windows 8. Virtualbox, an open source hypervisor, is used on this machine to run Ubuntu 14.04 OS. Mininet and POX controller are installed inside Ubuntu to set up the simulation test environment.

4.1.2. Network Emulation Testbed: Mininet

Mininet is a network emulator which creates a network of virtual hosts, switches, and links. Mininet hosts run on standard Linux network software and can be used as a flexible network testbed for developing OpenFlow applications. Mininet tool allows complex topology testing using Python Application Programming Interface (API).

4.1.3. Network Topology

The network topology used in this thesis is presented in Figure 4.1. It consists of a wide fat-tree topology with 8 servers, four are active and four are standby, 7 OpenFlow

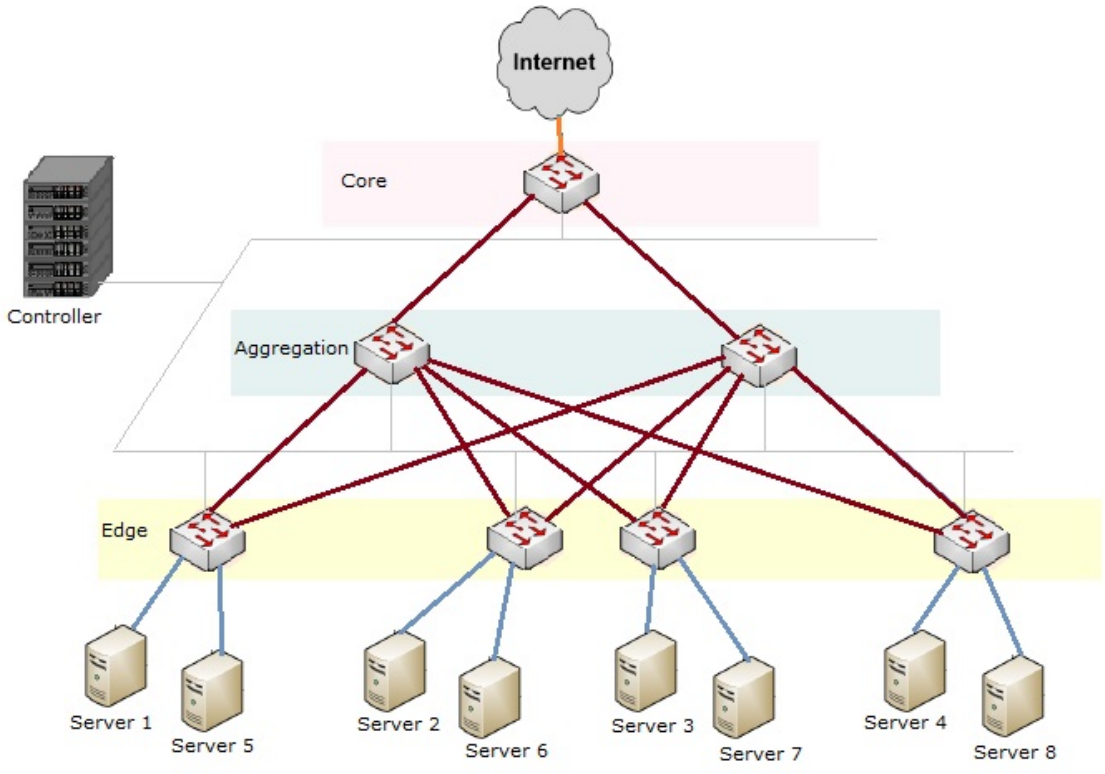

Figure 4.1: Typical Datacenter Network Topology

switches for multi equal path connectivity among all the servers and 1 controller. The controller is connected logically to all the switches whereas physically it can be present anywhere as long as it is reachable by all the switches. This topology is also known as ”Clos” where every lower-tier switch is connected to each of the top-tier switches in a full-mesh topology. The core switch has the same number of links as the number of aggregation switches, whereas the aggregation switches have the same number of links as the number of edge switches.

This type of topology is commonly used in datacenters due to its high resiliency and potential of load balancing.

4.1.4. Software Defined Networking Controller: POX

The POX [23] is an open source controller for developing SDN applications. POX controller provides an efficient way to implement the OpenFlow protocol which is the de facto communication protocol between the controllers and the switches. Using POX controller you can run different applications like hub, switch, load balancer, and firewall. Tcpdump packet capture tool can be used to capture and see the packets flows between POX controller and OpenFlow devices.

4.2. Datacenter SDN Architecture

The implemented architecture aims to slightly change the existing content delivery datacenter architecture in order to integrate the legacy core components into the OpenFlow network. The aggregation switches and access switches functionality are

maintained. The Core switch is no longer responsible for server selection and load balancing decision. It queries the controller via southbound interface using the Open API. The main objective of the proposed scheme is to achieve equal utilization of all the servers and links. It allows the traffic to spread in network to avoid the overloading of any single server or congesting any single link.

Since we focus on content delivery to the external users, new flows are assumed to be coming from the internet to the core switch. We are using two virtual IP addresses, one is Mice virtual IP address(MVIP) which is used to represent all the replicated server to handle mice requests and other is Elephant virtual IP address(EVIP) which is representing all the replicated server handling elephant requests. When core switch receives any new flow, it communicates with controller to service it. Depending on the destination virtual IP address, the server and path are computed by the SNLB scheme in the controller.

4.3. Load balancing Scheme of SNLB

To have an overall view of how SNLB works, let us consider the following simple scenario. Let us assume that a network is just initiated and the flows from external clients begin to enter in content delivery datacenter network. Initially when flow arrives to the core switch then the switch will contact controller for the flow due to the absence of matching flow entry. The controller will distinguish the flow type based on MVIP and EVIP. The corresponding module for server selection will be executed where the server with least number of flows will be selected to service the flow. Once the server is selected,

the path selection algorithm will compare multiple paths available to reach the selected server and the path with least flow will be selected as discussed in section 3.1.

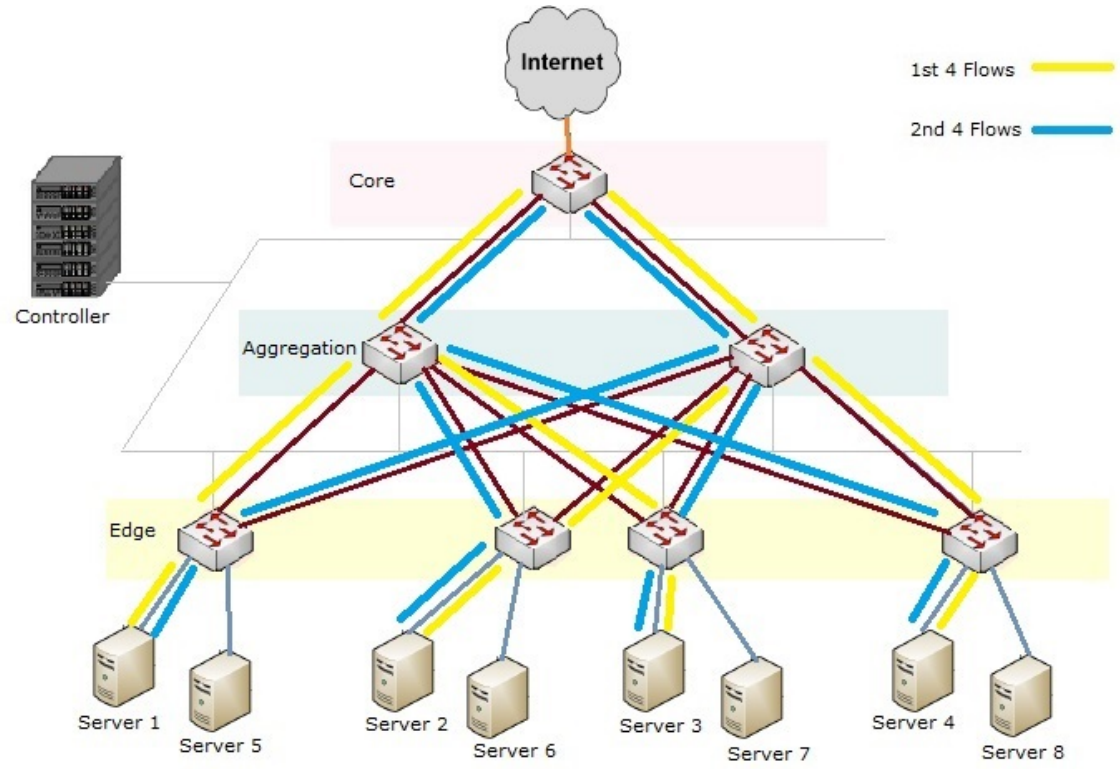

Figure 4.2: Load balancing Scheme of SNLB

Controller will install the flow rules in switches to handle this flow. Flow rules installed in switches will direct the flows as shown in the topology in figure 4.2.

SNLB scheme will separate the server selection and path selection for mice and elephant flows. This will allocate large elephant flows and small mice flows evenly in the network. Also only the servers in the available active servers list will be selected. Idle servers will only be activated and assigned in the case of overload or fault tolerance.

4.4. Testing SNLB

Server and Network Load balancing is tested and compared to currently used Global first fit(GFF) load balancing, Round robin(RR) based load balancing as well as Load based load balancing. The main differences of all these controllers are mainly in the server and path selection processes. These differences lead to different traffic loads on the links of network.

Global First Fit: As global first fit algorithm is widely used in the industries, we selected it to compare with our proposed SNLB system. When a new flow is received, GFF algorithm linearly searches all the possible servers and paths to the selected server. It find the path which have enough bandwidth to accommodate the flow. If such a path is found then that flow is placed on that path.

This algorithm depends on the bandwidth requirement of the flow and path capacity information in order to compute the path. Note that a flow is greedily assigned the first path that can accommodate it. Below is the algorithm for Global first fit:

Let S_n is the server with load n. N is the number of active servers. S_{Load} is described as the maximum load handle by the server. If a flow F_k arrive with load requirement k. Then server will be selected with an algorithm defined as:

$$\text{if } S_n + F_k < S_{Load}, \text{ where } n \in \{1,..,N\} \tag{4.1}$$

$$\text{select } S_n$$

Let P_i is the path i for the server n. N is the number of equal cost paths from the core switch to server S_n. $P_{capacity}$ is described as the maximum bandwidth of the link. If a flow F_j arrive with bandwidth requirement j. Then path will be selected with an algorithm

defined as:

$$\text{if } P_i + F_j < P_{capacity} \qquad (4.2)$$

$$\text{select } P_i$$

Round robin: This algorithm is also one of the widely used in industry hence we used it to compare with our SNLB algorithm. When a new flow received. This algorithm selects server and path in round robin fashion. Irrespective of the type of flows, server is picked and allocated in round robin fashion to service the request. And path towards server is also selected in round robin fashion without any consideration of the type of flows. This algorithm requires to track the last server and path allocated. Below is the algorithm for Round robin.

Let S_n be the server selected for the last flow. N is the number of active servers. If a flow F_k arrive, k is k^{th} flow. Then server will be selected with algorithm defined as:

$$\text{if } S_n \text{ selected for } F_{k-1} \text{ , where } n \in \{1,..,N\}$$

$$\text{if } (n == N) \text{ then } n = 0, \text{ else } n = n+1 \qquad (4.3)$$

$$\text{select } S_n$$

Let P_i is the path i for the server n. N is the number of equal cost paths from the core switch to server S_n. If a flow F_k arrive. Then path will be selected with algorithm defined as:

$$\text{if } P_n \text{ selected for } F_{k-1} \text{ , where } n \in \{1,..,N\}$$

$$\text{if } (n == N) \text{ then } n = 0, \text{ else } n = n+1 \qquad (4.4)$$

$$\text{select } P_n$$

Load based: This algorithm has recently evolved and performed better than GFF and Round robin so we used it to compare with our SNLB algorithm. When a new flow

received, this algorithm selects server and path based on least load on server and least load on the path towards server, irrespective of the type of flow. This algorithm requires to track the load on servers and paths. Algorithm for Load based scheme is discussed below:

Let S_n be one of the servers. $L(S_n)$ be the load on S_n. When a flow F_j arrives and server selected for flow F_j is $S(F_j)$. Then server will be selected with algorithm defined as

$$S(F_j) = \min_{n} L(S_n), \quad 1 \leq n \leq N \tag{4.5}$$

Let P_i be path i which is one of the k equal cost paths from the core switch to server n. $L(P_i)$ be the load on P_i. If a flow F_j arrive and path selected for flow F_j is $P(F_j)$. Then path will be selected with based on the following condition

$$P(F_j) = \min_{i} L(P_i), \quad 1 \leq i \leq K \tag{4.6}$$

4.5. Performance Measurement and Expectations

The SNLB controller is compared with Global first fit, Round robin and Load based controllers in terms of performance. Following parameters are used for evaluating the performances of different schemes.

Latency: Network latency is measured by sending a packet that is returned to the sender; the round-trip time is considered the latency. Ideally latency is as small as possible. Excessive latency creates bottlenecks that prevent data from filling the network pipe, thus decreasing effective bandwidth..

Jitter: It is defined as a variation in the delay of received packets. The sending side

transmits packets in a continuous stream and spaces them evenly apart. Because of network congestion, the time between arrival packets can vary instead of remaining constant. Jitter is used to measure such variation.

Network Throughput: It is the rate of successful message delivery over a network.

Convergence: Convergence addresses the manner in which networks recover from network changes.

Scalability: It refers to the capability of the network to remain efficient and operational as the size increases.

Overload: It is the situation that network congestion occurs. It deteriorates network performance , causes packet loss or prevents the establishments of new connections.

Fault Tolerance: Fault tolerance is the property that enables a system to continue operating properly in the event of the failure of some of its components.

Results are expected to be better for SNLB scheme because in this scheme flows have better distribution among servers and links in network. In load based, it believed to perform better than GFF and Round robin because of better load balancing in network but it does not distinguish mice and elephant flows. GFF is expected to perform worst because it tends to congest a path before it start allocating flows in other alternative paths.

4.6. Experimental Simulated Setup

Mininet

The emulated network for testing is created in mininet using python API. It consist of 7 OpenFlow switches, 8 hosts connected on edge switches which are utilized as

servers, 20 hosts are connected with core switch which are simulating internet and a remote controller on port: 6633. The topology used for testing is discussed in section 3.4.3.

Iperf

It is the traffic generator used to simulate the flows from client. This tool is used to create requests to the servers, also we used it to capture performance parameters: jitter and network throughput using tcp and udp traffics. Following are the commands used to generate traffic:

```
To start the server in tcp,
  iperf -s -i 5
To start mice flow in tcp:
  iperf -c 10.0.0.230 -i 5 -w 32k -t 600
To start elephant flow in tcp:
  iperf -c 10.0.0.200 -i 5 -w 128k -t 600


To start the server in udp,
  iperf -s -u -i 1
To start mice flow in udp:
  iperf -c 10.0.0.230 -u -i 1 -b 5 -t 600
To start elephant flow in udp:
  iperf -c 10.0.0.200 -u -i 1 -b 13 -t 600
```

Simulation Parameters

Clients	Servers	Protocol	Bandwidth TCP (window size)	Bandwidth UDP	Number of Run
Total = 20 Mice = 14 Elephant=6	Total = 8 Active=4 Idle = 4	TCP UDP	Mice= 32kbyte Elephant=128kbyte	Mice= 5Mbps Variable +/- 20% Elephant=13Mbps Variable +/- 20%	20

Table 4.1: Testing parameters

The parameters in table 4.1 are used for testing the SNLB scheme implementation in comparison with GFF, RR and Load algorithms:

For the testing, servers are connected with 100 Mbps link bandwidth. Time of simulation run is 600 seconds with 2 flows entering the network every minute. For 20 simulation runs, type of flow entering every minute is randomize between mice and elephant over the time. By randomizing the flows for simulation runs, the congestion in network varies over the period for all the simulations. Each of the algorithm is tested using 20 simulation runs and then the average values are used in results.

4.7. Results and Observations

This section discusses the testing results obtained after running the simulations. These results are presented as graphical and/or tabular form in order to make it easy to observe the results and compare among different algorithms.

4.7.1. Network Latency in SNLB scheme w.r.t. GFF, RR and Load algorithms

We can observe in Figure 4.3, Flow based SNLB scheme datacenter network has experience least latency over the increasing traffic with time. As this scheme has equally spread the traffic over the links and servers thereby utilized all the paths evenly, it suffered least load in paths which directly affects the latency values.

SNLB flow based scheme almost always performs better than other schemes in terms of latency. Load-based scheme also performs better than GFF scheme and Round robin scheme. GFF performs worst because of the fact that it is greedy in allocating the

path to flows and as a result more load in one particular path.

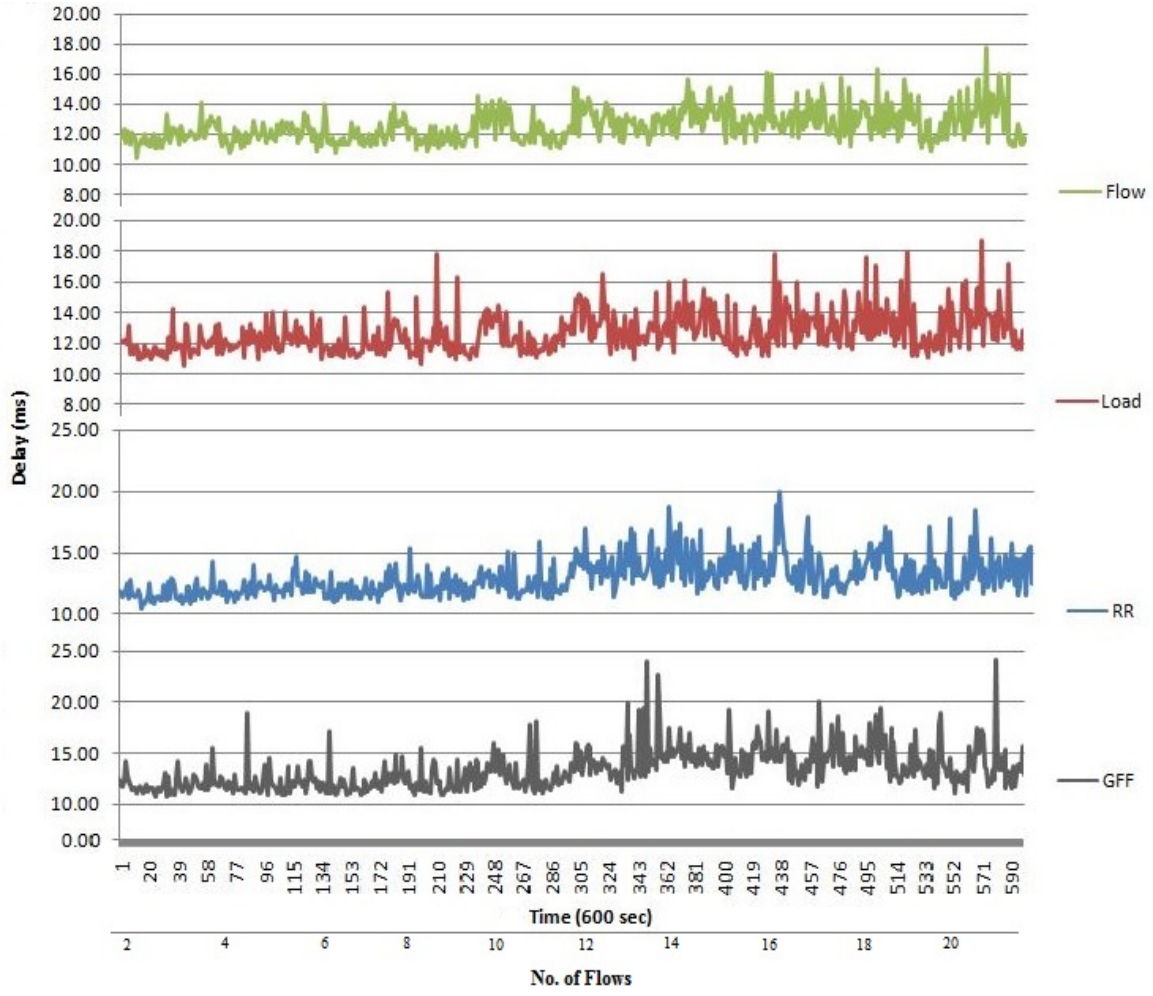

Figure 4.3: Latency in Network

Table 4.2, showing the average latency in the network in the last 60 sec of the simulation when the traffic load is the highest. We can see that SNLB is performing better than all other load balancing schemes.

We can use the results to predict that implementing SNLB scheme in datacenter network could be a good choice for improving performance.

Load Balancing Scheme	Average Network Latency (last 60 seconds)
SNLB	13.12 ms
Load based	13.28 ms
RR	13.44 ms
GFF	13.72 ms

Table 4.2: Average Latency in Network during traffic load

4.7.2. Jitter in SNLB scheme w.r.t. GFF, RR and Load based load balancing

Figure 4.4 shows that the SNLB scheme has smaller jitter in general than the other schemes. All the schemes have almost the same Jitter values at the beginning(the first 4 minutes) as the traffic is light. But as the traffic is increasing the jitter in GFF case is getting worst quickly, Load based experience less jitter than Round robin but SNLB has the best performance. These differences are due to the link utilization differences. GFF has allocated more flows on single path, Round robin distributes flows over multiple equal-cost paths but does not consider the types of flows or loads, Load based divided based on least load but may have allocated more elephant requests on any single link which may cause a sudden congestion. SNLB has the best distribution of traffic among servers and paths by utilizing all the links evenly. This leads to least delay and better jitter performance.

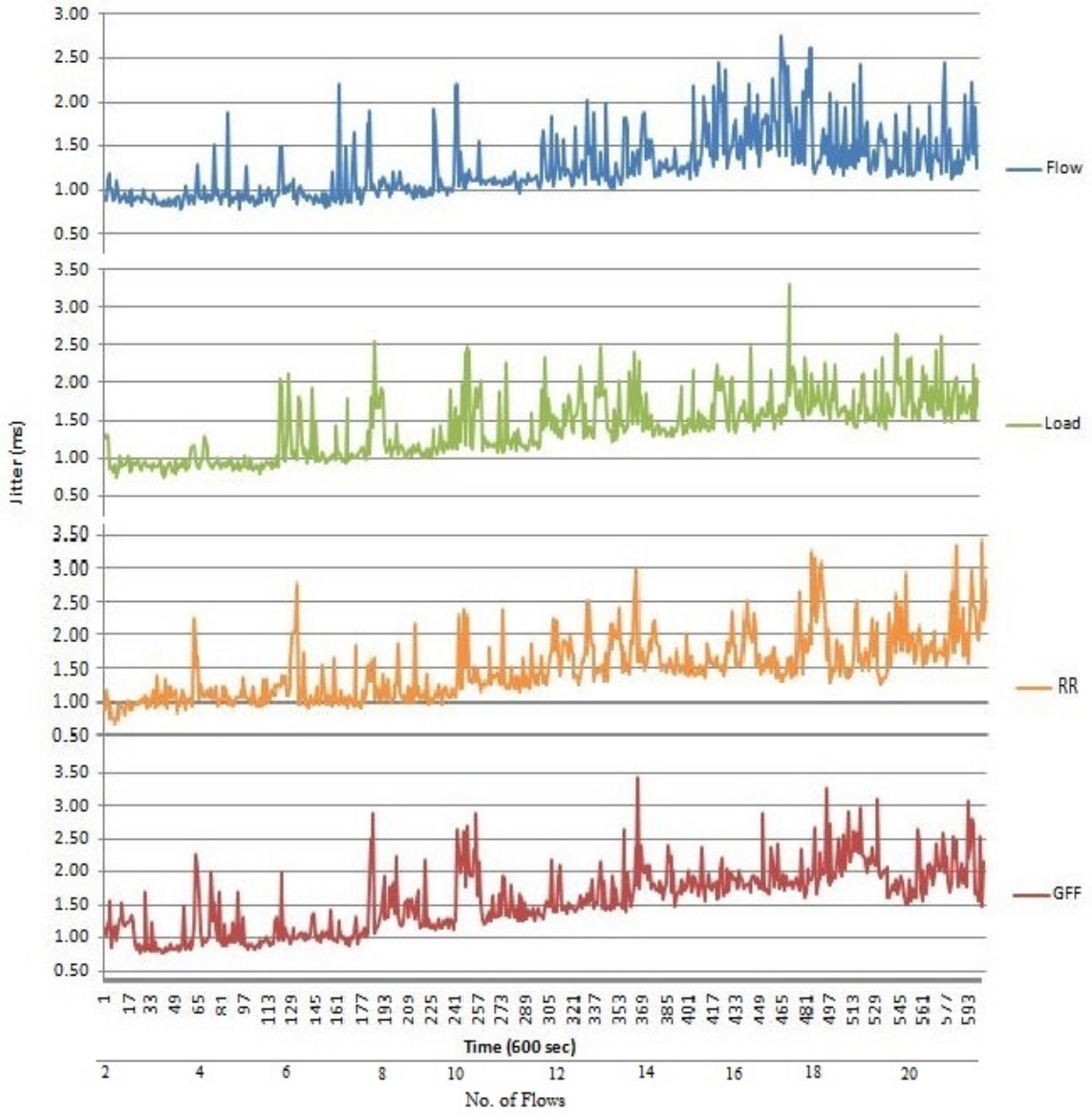

Figure 4.4: Network Jitter

Table 4.3, showing the average jitter in the network in the last 60 seconds of the simulation when the traffic load is the highest, in this period the maximum number of flows are present, increasing the load in the network. We can see that SNLB is performing better than all other schemes.

Load Balancing Scheme	Average Network Jitter (last 60 seconds)
SNLB	1.42 ms
Load based	1.81 ms
RR	2.06 ms
GFF	2.01 ms

Table 4.3: Average Jitter in Network during traffic load

Observing the performance of SNLB scheme in terms of jitter we can say that SNLB scheme could be better performer in datacenter networks.

4.7.3. Network Throughput comparison using TCP (fixed bandwidth of Mice and Elephant flows)

In this simulation study, we measured network throughput for every 5 second period in total 600 seconds of simulation run. Consequently, we have 120 instances of network throughput outputs, the horizontal axis is showing time in 120 instances which covers 600 sec. The traffic is TCP based. Thus TCP flow control is in effect.

Fig. 4.5 shows the throughput per flow. As the traffic load increases, the throughput per flow decreased. The decrease of the throughput is caused by the TCP flow control mechanism. Nevertheless, we can see that in Figure: 4.5, the network throughput for SNLB scheme is better than other schemes. Load based algorithm has also performed better than GFF and Round robin based algorithms but when compared with Flow based SNLB scheme, SNLB has performed better at the heavy traffic situation. In the graph, we

can observe that during initial 4 minutes as the traffic load is relatively low so all the schemes has

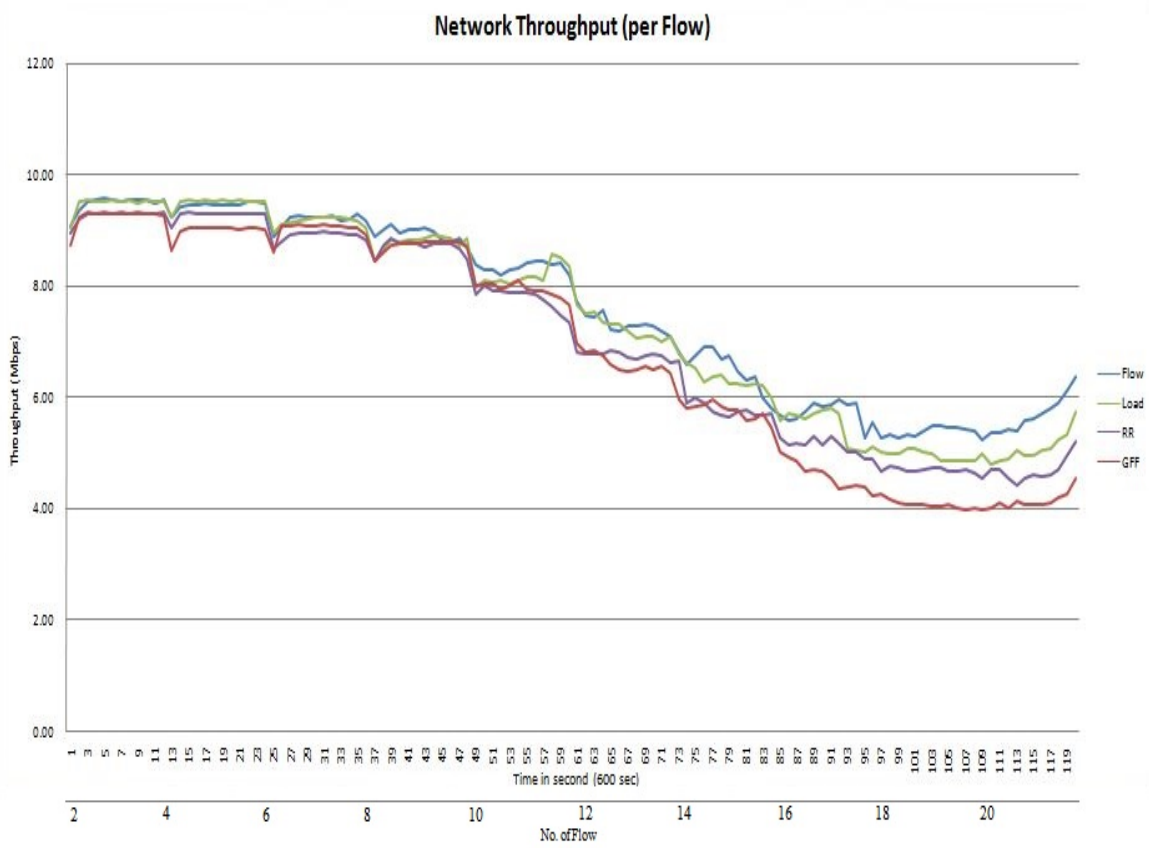

Figure 4.5: Network Throughput per Flow, TCP (fixed bandwidth)

performed nearly the same. But as the traffic increases the network links experiencing more load, SNLB throughput remains best followed by load based whereas GFF has the worst performance.

Table 4.4, shows the average TCP throughput per flow for the traffic in the last 1 minute of the simulation. During this period the number of flows is the maximum. The results shows that SNLB performs better than all other schemes.

Load Balancing Scheme	Average Network Throughput (last 60 seconds)
SNLB	5.67 Mbps
Load based	5.09 Mbps
RR	4.67 Mbps
GFF	4.14 Mbps

Table 4.4: Average Throughput in Network during traffic load, TCP

We can conclude from the results that SNLB scheme will give better throughput in the heavy traffic load condition.

4.7.4. Network Throughput comparison using UDP (variation in bandwidth of Mice and Elephant flows)

We have measured network throughput for the periods of 5 seconds in total 600 seconds of simulation run. In total we have 120 instances of network throughput outputs, the horizontal axis is showing time in 120 instances which totals 600 sec. The bandwidth requirements of the mice flow and elephant flow are varied. For the mice flows, the bandwidth is between 4 and 6 Mbps with the mean of 5 Mbps. For the elephant flows, the bandwidth is between 10.5 and 15.5 Mbps with the mean of 13 Mbps. The introduction of bandwidth variation simulates real content delivery datacenter flow behavior.

We can see in Figure: 4.6 and 4.7, the network throughput for SNLB scheme is better than other schemes. Load based scheme performs better than GFF and Round robin based algorithms. In the graph, we can observe that during initial 4 minutes as the traffic

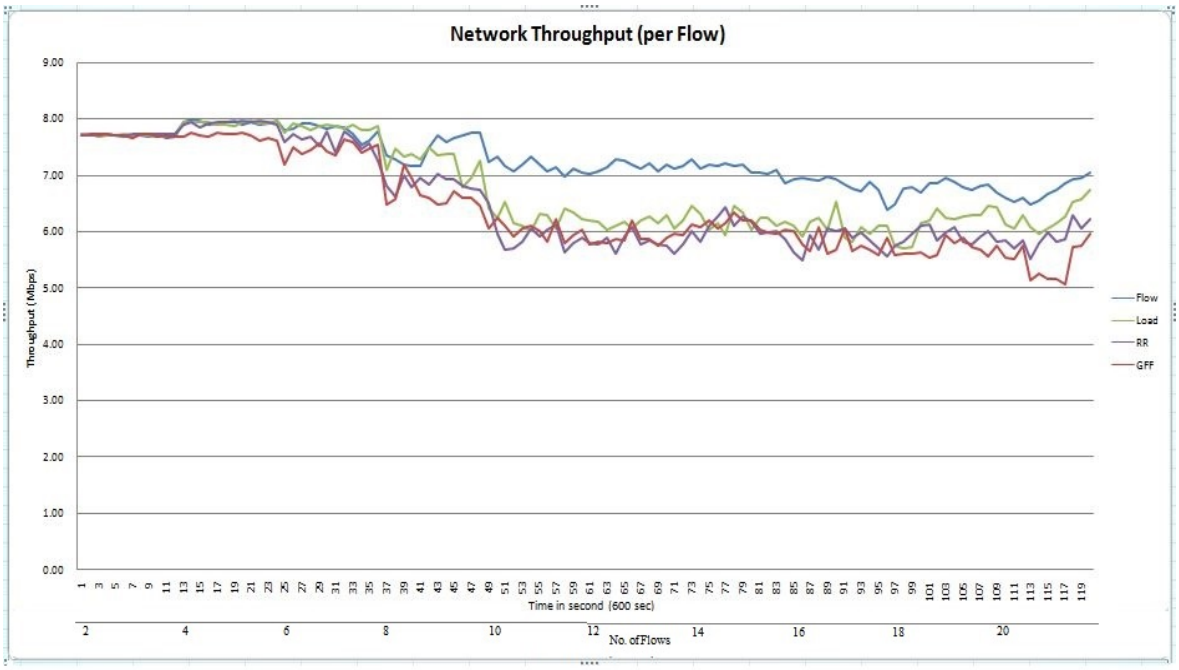

Figure 4.6: Network Throughput per Flow, UDP (medium load)

load is not high, all the schemes perform the same. But as traffic increases, SNLB throughput remains best followed by load based whereas GFF has the worst performance.

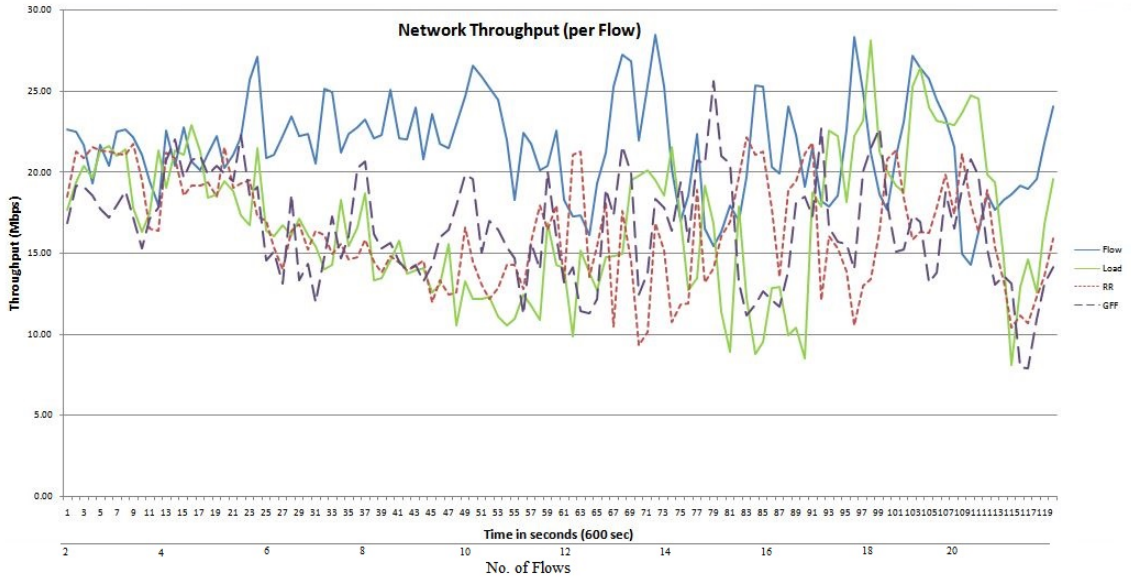

Figure 4.7: Network Throughput per Flow, UDP (high load)

Load Balancing Scheme	Average Network Throughput (last 60 seconds)
SNLB	6.72
Load based	6.27
RR	5.90
GFF	5.48

Table 4.5: Average Throughput, UDP (medium load)

Load Balancing Scheme	Average Network Throughput (last 60 seconds)
SNLB	18.51
Load based	17.59
RR	14.75
GFF	14.05

Table 4.6: Average Throughput, UDP (high load)

Table 4.5 and 4.6, showing the average throughput for the traffic using UDP in the network in last 1 minute, during this period the number of flows is maximum. We can see that SNLB is performing better than all other schemes.

We can conclude from the results that SNLB scheme will give better throughput in the network. In datacenter networks it could be considered a very good option to implement SNLB for improving the throughput performance.

4.7.5. Network Throughput comparison for Server spreading using UDP (variable bandwidth of Mice and Elephant flows)

We have measured network throughput for the servers location spread across edge switches compared with servers without spreading across the network. In the case where servers are not spreading across the network, we have 2 servers connected to same edge switch. The mice flows and elephant flows have variable bandwidths as described in the previous section.

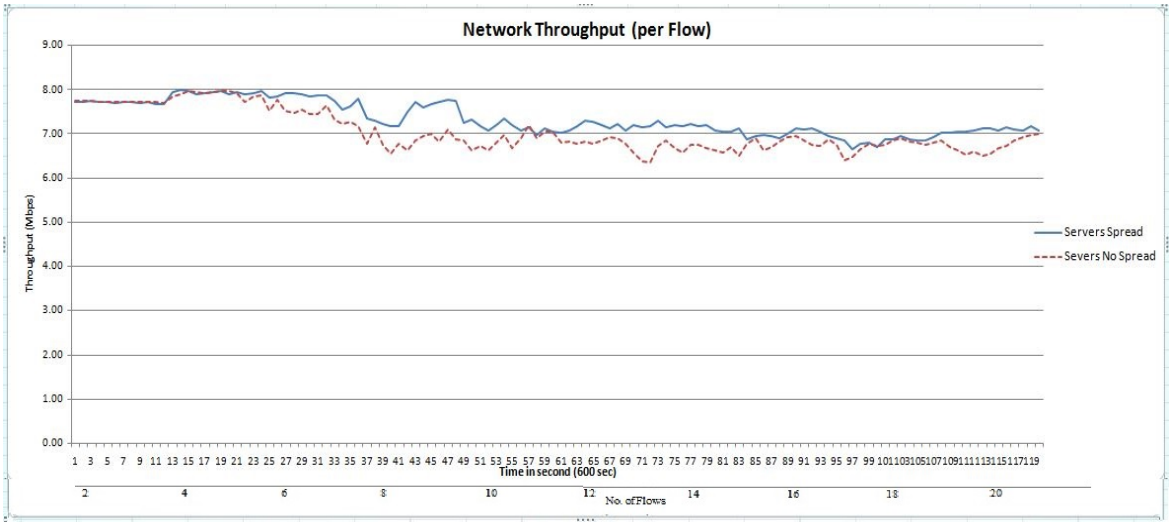

Figure 4.8: Network Throughput per Flow, Server spread (medium load)

We can see in the Figure: 4.7, the network throughput for the servers spreading across the network is better than servers without spreading. In the graph, we can observe that during initial 4 minutes as traffic is not very much in the network so both the schemes has performed nearly the same. But as the traffic increases, the throughput of the servers spreading scenario is consistently better than that of the no spreading scenario.

Table 4.6, showing the average throughput performance in the last 1 minute of the simulation. We can see that SNLB scheme with server spreading has a better throughput.

Load Balancing Scheme	Average Network Throughput (last 60 seconds)
SNLB Server with Spread	7.08
SNLB Server without Spread	6.71

Table 4.7: Average Throughput in Network, Server Spread (medium load)

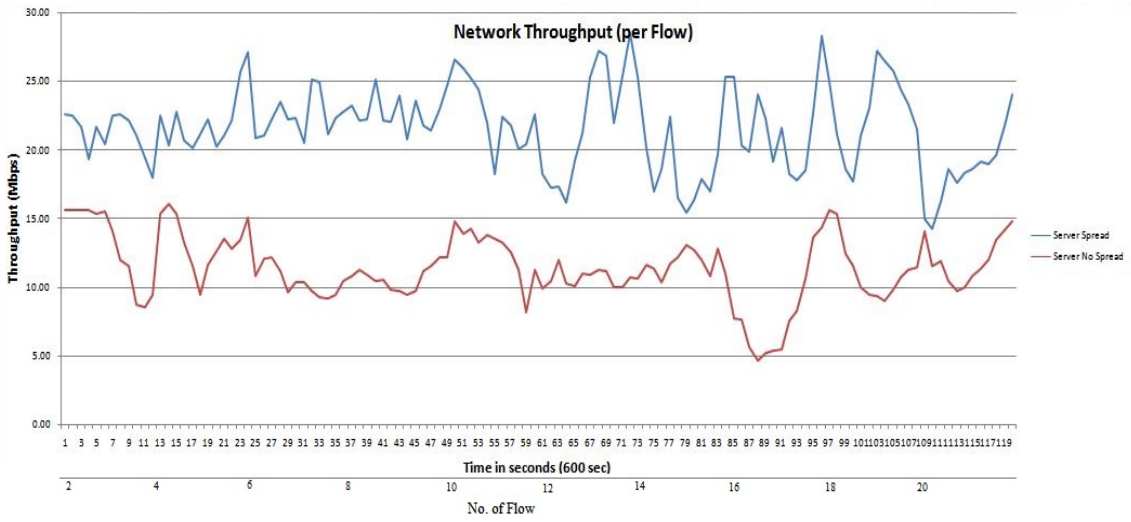

Figure 4.9: Network Throughput per Flow, Server spread (high load)

Load Balancing Scheme	Average Network Throughput (last 60 seconds)
SNLB Server with Spread	18.51
SNLB Server without Spread	12.02

Table 4.8: Average Throughput in Network, Server Spread (high load)

We can conclude from the results that in server activation/deactivation in overload handling scheme and server activation in fault tolerance scheme, spreading out the servers across the network will give better throughput. By selecting the server to activate and de-activated according to the loads of the servers does somehow help to spread out the servers locations. In content delivery datacenter networks it could be considered a very good option to implement SNLB for improving the performance.

Chapter 5

Conclusion and Future Work

Software Defined Networking (SDN) is seen as emerging and important networking technology. It has been adopted in many datacenter networks. SDN could bring flexibility in controlling architecture components, smart usage of the network resources (e.g. load balancing) and intelligent traffic steering (e.g. Quality of Service (QoS)), while decreasing the Capital Expenditure (CAPEX) and Operating Expenditure (OPEX) costs.

Our proposed SNLB scheme improves the performance in Content Delivery Datacenter Networks. The scheme intelligently utilizes the features emerged from SDN. In this scheme, the real time statistics of network to spread the traffic flows evenly among the servers and paths to the servers are utilized. In addition, the traffic overload and fault tolerance situations are handled efficiently using available network resources.

The performance results have shown that SNLB scheme performs better than GFF, Round robin and Load based algorithms in terms of:

Latency experienced in network

Jitter in the network

Network Throughput for TCP and UDP

Future Work

Server fault tolerance can be improved by introducing a mechanism to re-balance the active server distribution when new servers at a more desirable locations are available for

activation.

Another possible improvement is to adopt a better server spreading algorithm. In our results, we observes that if the servers are spreading in the network, network throughput is better. The scheme proposed in here does not guarantee the perfect servers spreading.

Bibliography

[1] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, Amin Vahdat, “Hedera:dynamic flow scheduling for data center networks,” in 7th USENIX Symposium on Networked Systems Design and Implementation, 2010.

[2] J. Touch, R. Perlman, Transparent interconnection of lots of links (TRILL): problem and applicability statement, RFC 5556, IETF, 2009

[3] C. Hopps, Analysis of an equal-cost multi-path algorithm. RFC 2992, IETF, 2000.

[4] G. Iannaccone, C. Chuah, S. Bhattacharyya, C. Diot, Feasibility of IP restoration in a tier-1 backbone, IEEE Network 18 (2) (2004).

[5] Z. Cao, Z. Wang, E. Zegura, Performance of hashing-based schemes for internet load balancing, in: Proc. IEEE INFOCOM, 2000.

[6] I.S. Association, IEEE Std 802.1AX-2008 IEEE Standard for Local and Metropolitan Area Networks – Link Aggregation, 2008.

[7] W. K. A. R. Curtis and P. Yalagandula, “Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection,” in INFOCOM, 2011.

[8] Load balancing with Cisco Express Forwarding. Tech. rep., Cisco Systems Inc., 1998.

[9] Cisco, “Quality of Service - The Differentiated Services Model,” [Online; accessed December-2014].

[10] IETF RFC 1349, “Type of Service in the Internet Protocol Suite,” 1992.

[11] RFC 2474, “Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers,” [Online; accessed January-2015].

[12] Open Networking Foundation (ONF) White Paper, “Software-Defined Networking: The New Norm for Networks,” 2012.

[13] Open Networking Foundation (ONF), “OpenFlow-enabled Software Defined Networking (SDN) and Network Functions Virtualization,” 2014

[14] https://en.wikipedia.org/wiki/Elephant_flow [Online; accessed November-2014].

[15] Open Networking Foundation (ONF), “OpenFlow Switch Specification, version v.1.4.0,” https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf, 2013, [Online; accessed February-2015].

[16] N. McKeown, “OpenFlow (Or: “Why can’t I innovate in my wiring closet?”),” http://cleanslate.stanford.edu, [Online; accessed February-2015].

[17] R. Wang, D. Butnariu, and J. Rexford, \Openow-based server load balancing gone wild," in Proc. of the 11th USENIX conference on Hot topics in management of inter- net, cloud, and enterprise networks and services (Hot-ICE 2011), (Boston, MA. USA), March 2011.

[18] T. Graf, “Underneath OpenStack Quantum: Software Defined Networking with Open vSwitch.”

[19] https://en.wikipedia.org/wiki/Mouse_flow [Online; accessed November-2014]

[20] GREENBERG, A., JAIN, N., KANDULA, S., KIM, C., LAHIRI, P., MALTZ, D.,

ATEL, P., AND SENGUPTA, S. VL2: A Scalable and Flexible Data Center Network. In Proceedings of ACM SIGCOMM, 2009.

[21] http://sdn.calsoftlabs.com/downloads/WhitePapers/Magic-of-SDN-in-Networking.pdf

[22] Lara, Adrian, Anisha Kolasani, and Byrav Ramamurthy. "Network innovation using openflow: A survey." (2013): 1-20.

[23] Fernandez, Marcial. "Evaluating OpenFlow controller paradigms." In ICN 2013, The Twelfth International Conference on Networks, pp. 151-157. 2013.

[24] http://www.juniper.net/techpubs/en_US/junose14.1/topics/topic-map/ip-ecmp-load-sharing.html#jd0e133

[25] GREENBERG, A., LAHIRI, P., MALTZ, D. A., PATEL, P., AND SENGUPTA, S. Towards a Next Generation Data Center Architecture: Scalability and Commoditization. In Proceedings of ACM PRESTO, 2008.

[26] Kim, Hyojoon, and Nick Feamster. "Improving network management with software defined networking." Communications Magazine, IEEE 51, no. 2 (2013): 114-119.

[27] N. Handigol, S. Seetharaman, M. Flajslik, N. McKeown, and R. Johari, \Plug-n-serve: Load-balancing web tra_c using openow," in ACM SIGCOMM Computer Communi- cation Review, Aug. 2009.

[28] Cisco Data Center Infrastructure 2.5 Design Guide. http://www.cisco.com/univercd/cc/td/doc/solution/dcidg21.pdf.