Analysis of SDN based ECMP for Data Center Networks

A Project Report Presented to
The faculty of the Department of Electrical Engineering
San José State University

In Partial Fulfillment of the Requirements for the Degree
Master of Science

By

Nirmal Karia 010018723
EE297B Section 01, Spring 2016
nirmalkaria@gmail.com (669) 300-7019

Dharmesh Bhanushali 010020114
EE297B Section 09, Spring 2016
dharmeshdpb@gmail.com (669) 225-8750

Department Approval

Dr. Balaji Venkatraman Project Advisor	*(signature)*	Date
Dr. Nader Mir Project Co-Advisor	*(signature)*	Date
Dr. Thuy T. Le Graduate Advisor	*(signature)*	Date

Department of Electrical Engineering
Charles W. Davidson College of Engineering
San José State University
San Jose, CA 95192-0084

ACKNOWLEDGEMENTS

We would like to take this opportunity to thank Dr. Nader F. Mir, our advisor, for guiding and advising us all along the progress of this Project. He encouraged us to provide our views related to the feasibility of our project implementation by creating a free learning environment that gave us the opportunity to express our views and opinions regarding the project. He patiently answered all our doubts and questions. The group meetings conducted helped us to enhance our understanding of the topic and area of research.

We would also like to take this opportunity to thank the Electrical Engineering department for providing us the tools and resources necessary for successful completion of our project.

TABLE OF CONTENTS

TABLE OF FIGURES

ABSTRACT

SDN is a rising phenomenon in today's computer network architecture. SDN splits the control plane and data plane of a network architecture. This split at each forwarding device is consolidated and handled by a centralized entity. This centralized control plane provides an open interface to forwarding devices and data plane. With lack of port densities and increased costs in data center architecture, fat tree architecture with high speeds links are more preferable. For multi-path routing data center architectures use ECMP as routing protocol to forward flows. Due to lack of dynamic scheduling capabilities and flexibility of ECMP, ECMP cannot not account for fair bandwidth utilization of a link. In this report, we performed two experiments in a fat tree topology based on SDN environment.

The first is based on ECMP in fat tree topology. The second experiment involves the use of optimized ECMP in fat tree topology with the use of SDN based environment. The use of ECMP through open flow protocol (SDN based environment) made it possible to dynamically adjust flows. Thus, with the help of centralized control it is possible to dynamically assign traffic loads as per links bandwidth utilization. The results demonstrate that the second experiment can improve the performance of throughput in data center networks. Thus the use of central controller can significantly outperform the traditional ECMP algorithm in data center networks.

CHAPTER 1. INTRODUCTION

1.1 HISTORY OF DATA CENTER TECHNOLOGY

With the fast improvement of big data, distributed computing and Internet, for example, colleges, research labs and organizations, are building tremendous Data center to bolster their applications. Those applications, for example, scientific and data analysis, information investigation, warehousing and huge scale system administrations, require significant intra-cluster data transfer capacity (bandwidth). As those applications keep on expanding, scaling the limit and handling capacity of server farms has turned into a test issue. So as to adjust to existing network conditions, make proficient utilization of network resources, specialists use traffic engineering to manage flows for forwarding paths.

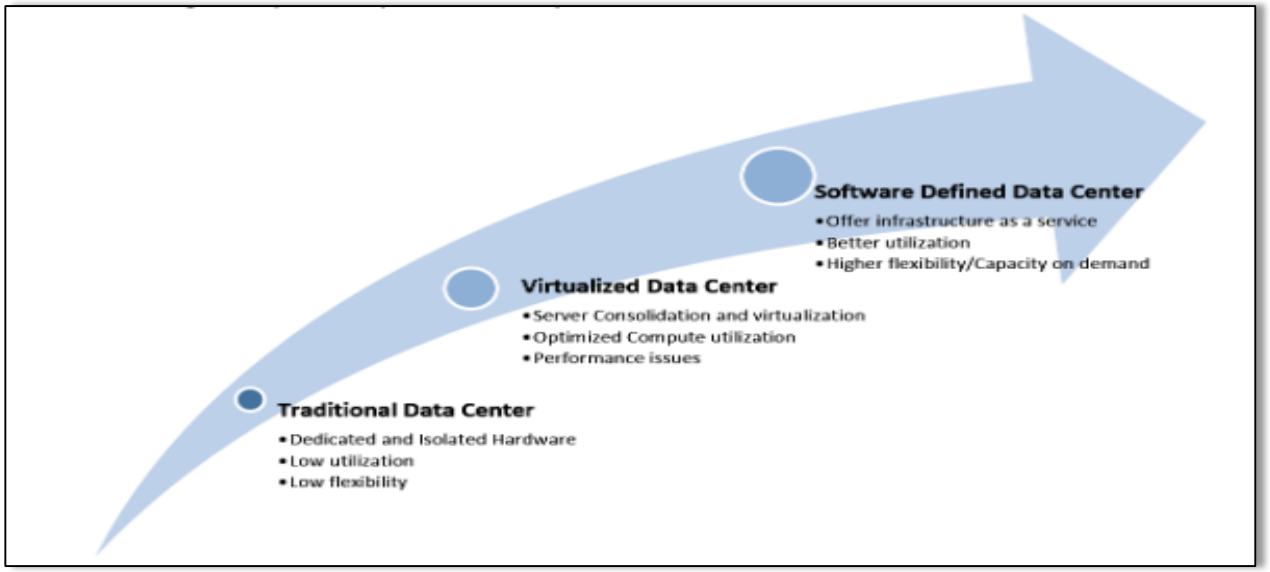

Figure 1:- Timeline of Data Center Evolution

Today's data centers constitute by a huge number of machines with huge total data transfer capacity requests. The conventional Internet correspondence are controllable through an unassuming number of given host sets. There are especially a little measure of ways between correspondence host sets. The data centers are always showing signs of change over both time and space. Those properties intrinsic in the server farm makes the customary network design and protocols no more suitable. An average data enter topology consists of three level trees of switches or switches. For instance, a three-layered topology has a core layer in the foundation of the tree, an aggregate layer in the center and an edge layer at the leaves of the tree. Those designs have different ways points of interest, which can convey full transfer speed between discretionary hosts. To decrease costs, specialists proposed an enhanced design, Fat Tree topology.

Most extensive IP systems work Open Shortest Path First (OSPF) to select a shortest path for every pair of correspondence host. In any case, if we use OSPF in traditional data center architecture, it cannot utilize multipath advantages of this data center topology. In server farm systems, Equal Cost Multipath (ECMP) is regularly used to statically select flows across multiple path available. As indicated by the stream's hash esteem, distinctive scopes of stream are sent to various way. This characterized mapping of streams to ways does not represent data transfer capacity use of current system. ECMP cannot understand reasonable utilization of connection's data transfer capacity. To tackle this issue, we utilize Open Flow convention to enhance the ECMP protocol.

In this report, we created a data center topology in an SDN based environment calculated the bandwidth utilization of links in different scenarios and evaluated the performance of two experiments. We used ECMP in the first experiment in traditional data center topology and evaluated the results based on bandwidth utilization of links. In the next experiment we used

ECMP in SDN based environment, where flow handling is managed by a central controller and compared the results based on bandwidth requires at the core links. Depending upon the bandwidth used at links and flow being forwarded in both experiment we evaluated the results.

Most data center topologies have been composed as Fat Tree topology. These multi-established trees have numerous equivalent cost paths between all host and destination pairs. A key test is to powerfully forward streams along these equivalent cost paths to enhance the throughput of Data centers. ECMP has been broadly utilized as a part of data center architecture to statically forward streams over numerous equivalent cost ways. This static mapping of streams to ways does not represent either current system usage or stream size, with coming about some connections data transmission use surpass limit while other still sit out of gear. To take care of this issue, we proposed SDN-based ECMP calculation to acknowledge dynamic stream forwarding as indicated by connections transmission capacity use. SDN is another system design that permits system overseers to control system movement adaptable. There is less research work in server farm system of SDN engineering. Consequently, we explored how to utilize the adaptability of SDN innovation to enhance the throughput of server farm systems.

There are three fundamental focuses for a SDN-based server farm engineering. They are physical topology, forwarding flows as per bandwidth utilization and routing protocol. They are not autonomous, that is, the execution of one will be impacted by the decisions of others.

1.2 TOPOLOGY

Bandwidth is turning into the bottleneck for adaptability of large scale data center topology networks. Existing answers for explaining this bottleneck basically utilize progressive systems of switches. With the costly, non-item switches at the highest point of the hierarches convention, one issue is that the port thickness of top of the line switches limits general group size. Furthermore, it brings about high cost. Late methodologies, for example, Fat Tree and are Clos topologies that utilization different center changes to give full transmission capacity between any pair of hosts in the system. Those topologies are interconnected by a few layers of changes to evade imperfections in port densities which exists in business switches. Late research advocates the even development of server farm systems. These exploration introduce a server farm correspondence design that influences ware Ethernet changes to convey adaptable transmission capacity for extensive scale groups. The consequences of their tests demonstrate that this topology can convey versatile data transmission at fundamentally bring down expense than existing strategies.

1.3 ROUTING

The technique we used for multi path routing is ECMP. There is another way of static VLANS also. Both of these ECMP and static VLANS properly utilize the benefit of multiple equal cost path in a fat tree architecture. They perform load balancing and select a best path among all the equal cost path in a fat tree topology. This helps to achieve multi-path routing in an efficient manner and flows are forwarded along the same path and order is maintained.

The first step is to find the equal cost path in fat tree topology. Then flows are forwarded statically across multiple available paths. This static flow forwarding does not take into consideration efficient bandwidth utilization and packet size which results in collision and also effect on throughput.

The biggest limitation of ECMP is that if, there two or more flow which happen to take the same path available there is possibility that the bandwidth utilization of core links might exceed the threshold. The problem with ECMP is dynamic flow selection so we suggest a use of centralized scheduler which will overcome the issue of ECMP in data center network.

1.4 REQUIREMENTS OF DATA CENTERS

The primary prerequisite of most associations is business coherence; if there is a framework disturbance, IT operations may get to be impeded which can affect accessibility of administrations to clients. To minimize any odds of interruption, accessibility of solid base is an unquestionable requirement. Moreover, data security is likewise vital and henceforth a server farm must offer a protected domain with controlled access to decrease breaks. A server farm in this way ought to guarantee its usefulness and uprightness.

The TIA-942 of the Telecommunication Industry determines the necessities for telecom server farms including both undertaking server farms web server farms. It additionally indicates natural necessities for server farm particular gear. For powerful server farm operation, it is important to give a fundamental domain appropriate to establishment of hardware in an office. Telecom server farms require building pieces which are dull in nature to give incorporated hardware and building designing and simple development and versatility.

A few server farms convey remote frameworks to get to and oversee server farm hardware, which are run utilizing computerized scripts to perform nonexclusive operations which don't require nearness of work force. Such server farms are worked without lighting and are known as "Light-out" server farms. This takes into account to find server farms in remote scantily populated regions, in this manner expanding vitality productivity, decreasing staff costs and evoking noxious assaults on base.

Maturing server farms and quick IT development has provoked associations to exploit vitality and execution based efficiencies of current gear and perform server farm change. This procedure is an incorporated methodology comprising of a few ventures did all the while as against the customary serial server farm approach. The Projects in a Data Center Transformation are as per the following:

- **Standardization/Consolidation:** Its motivation is to lessen the aggregate number of server farms required by an association. It involves supplanting matured gear with more up to date ones which give enhanced limit and execution, alongside institutionalization of systems administration and administration stages for simple sensibility.
- **Virtualization:** IT Virtualization can be utilized to supplant servers in server farms and can be utilized to make virtual desktops which are then facilitated in server farms and leased on a membership premise. It brings down operation and capital costs.
- **Automation:** Automation of provisioning, fixing, setup, obligingness and discharge administration helps server farms run all the more productively.
- **Security:** Security of information and security of framework is interrelated. Information Security, User Security, Physical and system security should all be considered.

The Infrastructure Requirements of server farms can be characterized with the utilization of TIA characterized Tiers. Level 1 recognizes an essential server room with non-excess dissemination way and a normal high accessibility estimation of 99.67%. Level 2 is a server room which serves repetitive limit foundation and a normal accessibility of 99.74%. Level 3 is an extensive server room with numerous information appropriation ways, all base being double fueled and perfect with topology of site and having an accessibility of 99.98%. Level 4 must surpass all level 3 prerequisites and what's more ought to have all cooling framework as double fueled, flaw tolerant foundation with the capacity to store electrical force with a normal accessibility of 99.99%.

1.5 MODERN DATA CENTER DESIGN

Data centers generally keep running by substantial partnerships or government supported specialists. Run of the server farms can possess from one space to a complete building. Most hardware are server-like mounted in rack cupboards. Servers vary in size from single units to huge autonomously standing stockpiling units which are at times as large as the racks. Gigantic server farms even make utilization of transportation holders comprising of 1000's of servers. Rather than repairing singular servers, the whole holder is supplanted amid redesigns.

A run of the Data Center comprises of a solid, well-manufactured building lodging stockpiling gadgets, servers, web availability and broad cabling and systems administration hardware. It additionally comprises of cooling gear and foundation to supply power, alongside robotized fire dousing frameworks. It is crucial to take reinforcements occasionally to guarantee operability and high accessibility. The more basic programming and equipment, more endeavors are required for security.

Power in individual segments and repetition of basic areas in a server farm characterizes the accomplishment of a server farm. An excess force supply instrument permits managers to perform repairs on systems without disabling network to clients. Every single electric gadget produce heat while working and in the event that this warmth isn't scattered, it lessens productivity and can prompt disappointment of parts. To guarantee productive cooling, racks like standard racks are utilized to house servers. The server farms racks are put such that two racks confront each other, permitting the manager to get to the front end of the servers. Coolers are put above and beneath the racks permitting them to course through them and in the middle of the servers, dispersing the warmth.

Checking, clever scope quantification and administration of server farms' basic frameworks can be accomplished through Infrastructure administration. It empowers constant observing and overseeing of all IT frameworks and office base utilizing unique programming, equipment and sensors. It distinguishes and kill hazard sources to guarantee high accessibility of assets, additionally to recognize crevices in excess of foundation and give all-encompassing information on force utilization to quantify adequacy.

Correspondence inside the datacenter depends on IP convention based system comprising of switches and switches that exchange activity between the inward servers to the outside world. A few servers are regularly utilized for facilitating intranet and different administrations required for inward clients, for example, email servers, DNS, DHCP and intermediary servers. System security variables, for example, firewalls, doors, Intrusion location instruments and other observing components are additionally sent inside a datacenter.

CHAPTER 2. FAT TREE TOPOLOGY

2.1 SELECTION OF THE TOPOLOGY

Fat-Tree Topology is best suited and widely used at present in the Data Center Networks. Before deciding the the appropriate Topology to be used in the Data Centers. Let us learn about the different types of Topologies that can be used in the Data centers, their methodology of functioning and how they do into different networks. There are many topology structures from basic direct connections to the complex routing networks. Topology structures like **Point-to-Point** is the basic type of structural connections between the two direct hosts. As the number of networking devices in the network increase this starts getting complicated because we need to manage the the exact number of physical connections that of the number of devices in the network.

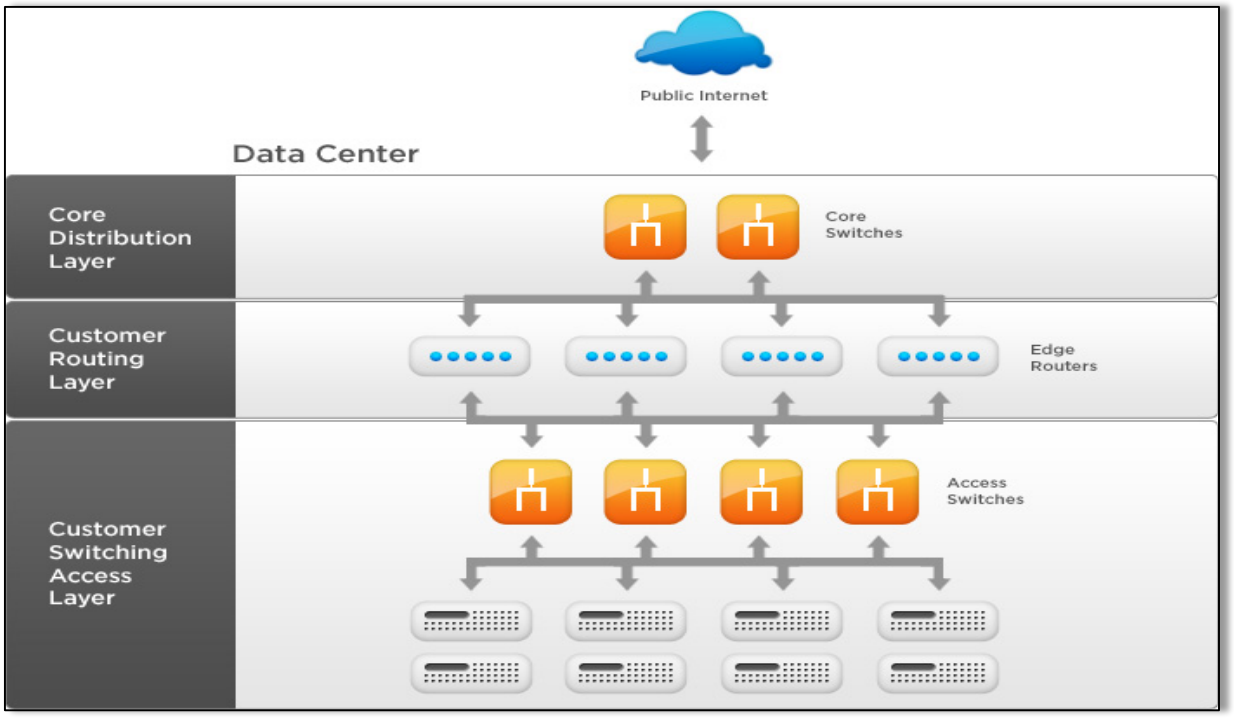

Figure 1: DCN Architecture

Secondly, we have **Bus topology**. It consists of a single main line bus which is connected to all the devices in the network and whole topology fails if the bus gets a lag or fails to function properly. We then have **Star and ring Topology** which is the smart device centric type of Topology. These topologies have a smart device like a router or a switch in between and are connected to the hosts. The difference between a start topology and a ring topology is that in the star the central device is the vital part, all hosts and the bus connection fail if the central device fails. Whereas in Ring topology, all the hosts are connected to the next one and they form a ring with their connection manner. This gives the point of failure to be the failure of any one host in the ring.

However, with the complex growing networks it was very challenging to get a reliable and consistent topology deign. Tree Topology served the purpose later. This structure was first named as Hierarchical Topology. It is most widely used structure in the networks at present. It is actually evolved from the cumulative properties of the Star and Bus topology. Structural features of the star topology were developed. Instead of keeping one central device, all the layer 2 and layer 3 devices were given importance and they were the building blocks of the core layer. On the other hand, properties of bus topology were inherited and the connection format was built in a systematic hierarchical manner. Actually, the topology separates the network into numerous layers of network. For the most part in LANs, the blue print of the devices in the network is divided in three layers of networking devices.

The bottom layer is the access-layer. It consists of the Edge Routers/Switches which are connected to the host beneath them. Edge devices are also known as T.O.R switches- Top of the rack switches Hosts are the initiators, serving them is the sole and foremost function of the topology. As the number of hosts increase or decreases the upper layers have to adjust themselves

and vary according to the math on which the topology efficiency is built. The center layer is called Aggregation Switch layer. This basically provides the functionality of the distribution layer, which fills in as the intermediate building block between Edge layer and Core layer. These devices are responsible for organizing the connections form the core layer and distributing them to the underlying Edge devices. The upper-most layer is the Core layer, and it the back bone of the network. It is the root of the tree and outlets all the nodes from it. All the Core devices are connected with the bus with all the respective underneath devices and this is the basic Tree structure. This was the best invention but failed if the root failed and whole topology went scrap after that until the root device is up again. This gave rise to the flawless solution of connecting a single tree structure with more than one root. Even if one root is down, the network is still reachable using the other root.

The Fat-tree topology was invented by "Charles E Leiserson" from MIT. He proposed the most efficient and flawless network structure for the Data Center communication. The thickness of the branch is the term for the bandwidth available on that medium. High bandwidth links are known thicker and low bandwidth links are known as skinny links. Fat-tree Topology provides with the backward compatibility within the defined structure. Serves the best support for the Ethernet layer (Layer-2). It is cost effective and consumes very less power which is the basic need nowadays, with the ever growing networks. It's built out of the low cost infrastructure and emits Low heat. The line speed can be user defined and the network proves to serve to the best according to whatever is defined.

2.2 FAT TREE ARCHITECTURE

Knowing that Fat-tree works on constant bandwidth on all the bisection. Every layer still provides equal aggregate bandwidth. All the ports support constant speed on the hosts. In the case of uniformly distributed packets on all the paths that are available, we can notice that all the devices will transmit at the same speed of line. The methodology and the mathematical calculations for making an efficiently working topology is as follows. The server racks to be interconnected in a fat tree topology (Three Layer topology).

- Each of the pod consists of ***(k/2) * 2*** servers and 2-layers of ***(k/2)*** port-k switches.
- Each of the edge switches must connect with ***(k/2)*** server and ***(k/2)*** aggregate switches.
- Each of the aggregate switches must connect with ***(k/2)*** edge and ***(k/2)*** core switch.
- All of the ***(k/2) * 2*** core switch has to respectively connect with ***k*** pods each.

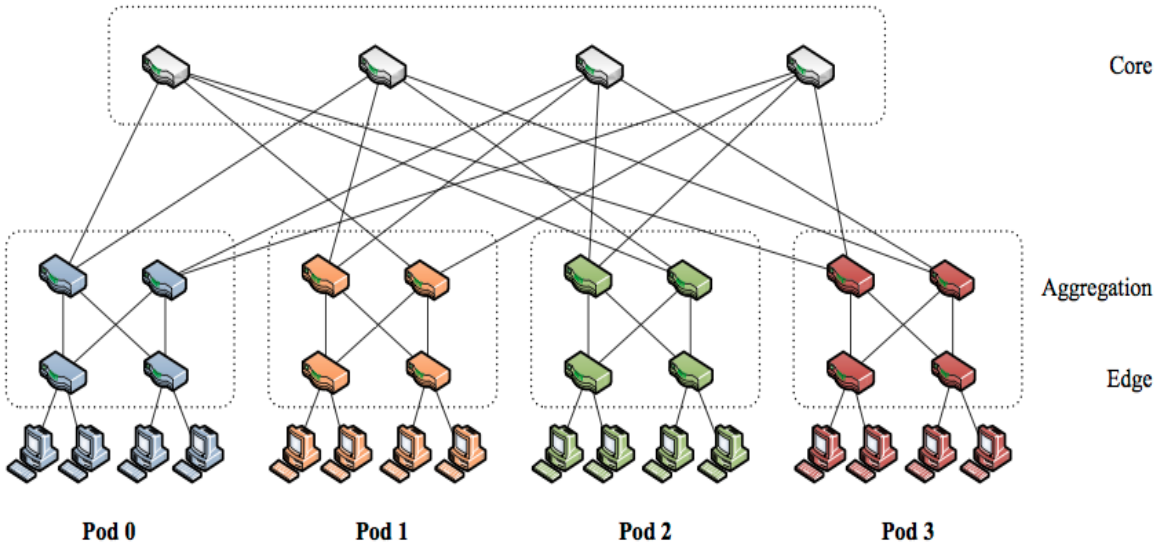

Figure 3:- Example of a Fat-Tree topology

2.3 NETWORK UTILIZATION

The following analysis shows the Network Utilization if the Fat-tree topology that we have used in this research project. We can derive that the bisection bandwidth of fat-tree structural networks in ideal case is 1.536 Gb/s. It shows all the following aspects in the table below such as, Aggregate Bandwidth of the network, Percentage of bisection bandwidth for the tree in ideal case, two level table, classification of the flow and scheduling method of the flow.

Test	Tree	Two-Level Table	Flow Classification	Flow Scheduling
Random	53.4%	75.0%	76.3%	93.5%
Stride (1)	100.0%	100.0%	100.0%	100.0%
Stride (2)	78.1%	100.0%	100.0%	99.5%
Stride (4)	27.9%	100.0%	100.0%	100.0%
Stride (8)	28.0%	100.0%	100.0%	99.9%
Staggered Prob (1.0, 0.0)	100.0%	100.0%	100.0%	100.0%
Staggered Prob (0.5, 0.3)	83.6%	82.0%	86.2%	93.4%
Staggered Prob (0.2, 0.3)	64.9%	75.6%	80.2%	88.5%
Worst cases:				
Inter-pod Incoming	28.0%	50.6%	75.1%	99.9%
Same-ID Outgoing	27.8%	38.5%	75.4%	87.4%

Figure 4 :- Network Utilization table for Fat-tree Topology

2.4 MULTIPATH ROUTING

Definition – "Multipath routing is the routing technique of using multiple alternative paths through a network, which can yield a variety of benefits such as fault tolerance, increased bandwidth, or improved security. The multiple paths computed might be overlapped, edge-disjointed or node-disjointed with each other."

Explanation - Methodology for using multiple different routing paths in a network to deliver the packets from a node to node. This technique actually considers best path and the successive best paths, for making the routing decisions. As a result, it gives an assortment of advantages such as better throughput (bandwidth), fault tolerance and enhanced security. All the different paths figured may be overlapping, edge-incoherent or hub disconnected within themselves.

These are the three vital components needed to properly implement multipath routing algorithm are Multipath Calculation algorithm that derives the logic for available multi-paths. Secondly an algorithm based on Multipath Forwarding capabilities that makes sure about the chosen path and guarantees to deliver the amount of traffic to the destination allotted on the path. Finally, the algorithm should execute a proper end-to end protocol which can determine and decide the multiple paths to be used.

The Important aspects to be considered while designing the multipath routing algorithm are to maintain the fault tolerance, the bandwidth availability and the load balancing in the designed network. To make full usage of available multiple paths for routing the packets in the network is commonly known as Concurrent Multipath Routing (CMR). This is achieved by making utilizing the resources by different carriers simultaneously, distributing the packet load all over the different available paths according to the resources available on each path, designing fast discovery system for all the path in case of failure.

2.5 EQUAL COST MULTIPATH ROUTING PROTOCOL (ECMP)

Definition – "Equal-cost multi-path routing (ECMP) is a routing strategy where next-hop packet forwarding to a single destination can occur over multiple best paths which tie for top place in routing metric calculations. This Multi-path routing is used in conjunction with most routing protocols, because it is a per-hop decision limited to a single router. It substantially increases the bandwidth by load-balancing traffic over multiple paths."

Explanation - ECMP proves to be an essential load balancing routing algorithm implemented in the networks. Despite using only single best path to route the packets to the destination, it uses all the different best paths that are available. ECMP figure outs the possible best path and checks their available resources for finalizing the routing decisions and when these multiple best paths are obtained, it sends out the traffic with available bandwidth.

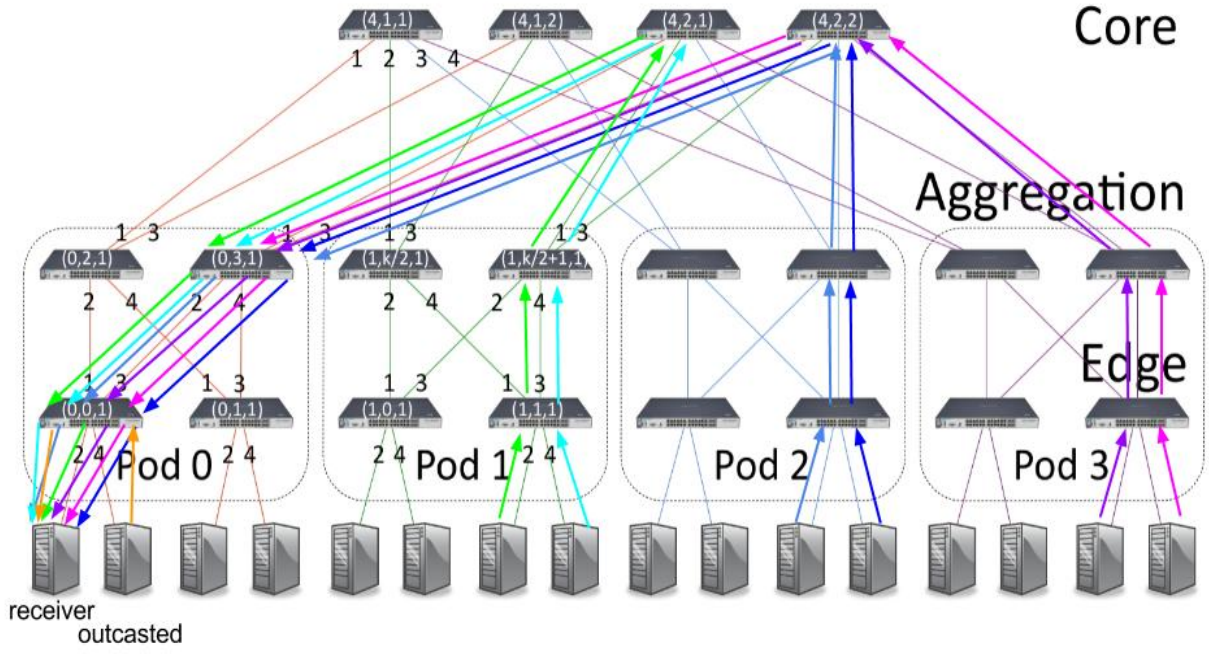

Figure 5 :- ECMP based routing in Data Centers Network

The working of ECMP varies based on different layers it is implemented on. ECMP is functioned to work differently on Layer 2 and it probably functions different on Layer 3. We, in this research have used Layer 2 functioning of ECMP and using open Vswitch. We implement ECMP on our topology constructed using 2 core switches, 3 pairs of Aggregate switches (total 6 aggregate switches) and same 3 pairs of Edge switches or Top Of the Rack switches (6 in total). Basically we break the arbitrary packet traffic and distribute it over a path and making sure that the respective single flow remain on that particular path for whole of its lifetime. This in turn helps to choose the hashing algorithm to be used and selecting the header information. Hence, we organize the in order delivery of the packet traffic at the destination. This information in header which is encapsulated on this sort of packets consists of the MAC address of the source, Mac address of the destination, IP address of the source, IP address of the destination, DSCP, Source port, Destination port and the next hop.

Load balancing for Layer 3 is achieved by separating the information load over multiple links. The probable measure we have to take care of here is by making sure that the individual TCP connection is established and maintained over a single link for achieving best performance in the built network. Hence, Layer 3 distribution of load must include Source IP, Destination IP, Source TCP port, Destination TCP port and lastly the protocol number. Forwarding in ECMP may cause jitter if the paths are chosen randomly and packets are sent over the path. Hence, we try to send the packets destined for a fix source-destination pair on the same path till the ongoing session is executed completely.

CHAPTER 3 SOFTWARE DEFINED NETWORK (SDN)

3.1 TERMINOLOGY IN SDN

Definition – "Software defined networking (SDN) is an approach to computer networking that allows network administrators to manage network services through abstraction of higher-level functionality. This is done by decoupling the system that makes decisions about where traffic is sent (the control plane) from the underlying systems that forward traffic to the selected destination (the data plane)."

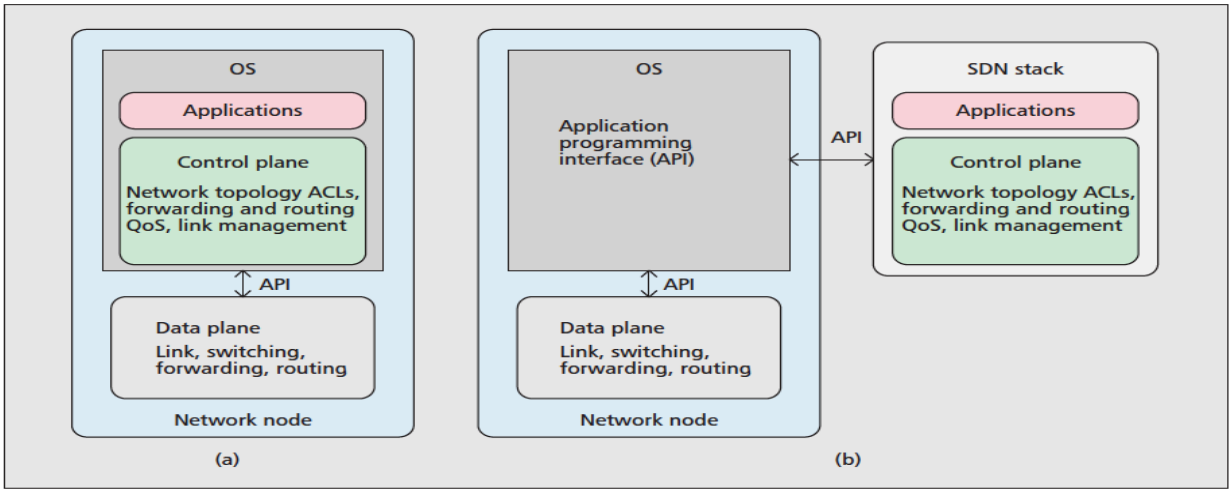

Figure 6 :- Basic SDN structure with separate Data and Control plane

Explanation- To cope up with the ever increasing Inter-Network and the complexity, it is important to maintain the performance and efficiency of the network. SDN proves out to be the solution which was the epitome of success in providing the ultimate network performance. Basically, SDN is built over three pillars of innovativeness:

- The data plane and control plane separation.
- The control plane Programmability.
- Application programming interfaces (APIs) standardization.

Normally for computer networks, packet flow is taken care of by the specifically devoted devices like routers and switches and other networking devices working together in a networks based and operated by different ISPs (Internet Service Providers). The functional characters and the rules which drives all these sort of devices can be organized and mapped in three basic sub-categories known as planes.

3.1.1 The Control Plane

This is designed to carry the signaling traffic and maintain open flow table. It performs tasks like routing setup, deciding the routing path from source to destination, QOS, etc. Management of the system and configuration of the same is also an important function of the control plane.

3.1.2 The Data Plane

This depends on the information derived by the control plane. According to the routes derived by control plane, the data plane switches data in the form of packets from source to destination over a selected path.

3.1.3 The Management Plane

Maintains the policy in the network and admins the network. It is like a sub-part of the control plane because even this one has to take care about the performance of the network.

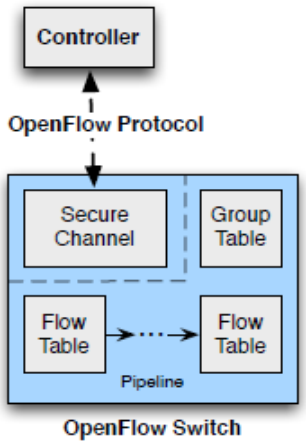

Figure 7 :- OpenFlow Protocol and Data Layer Interaction

The open flow model and design of the control plane allows to dynamically access all the resources and admin control of the network. SDN was actually invented to overcome the flaws in the old school routing algorithms like smartness of the switches, which is visible in every switch and each of them is equipped to function on a specific routing-algorithm. For example, we take distance-vector algorithm which will make a neighborhood routing table. Be that as it may, those switches are normally expensive. Moreover, since the forwarding table of every switch is built in a disseminated way, it gets difficult to troubleshoot and derive the best-path for routing.

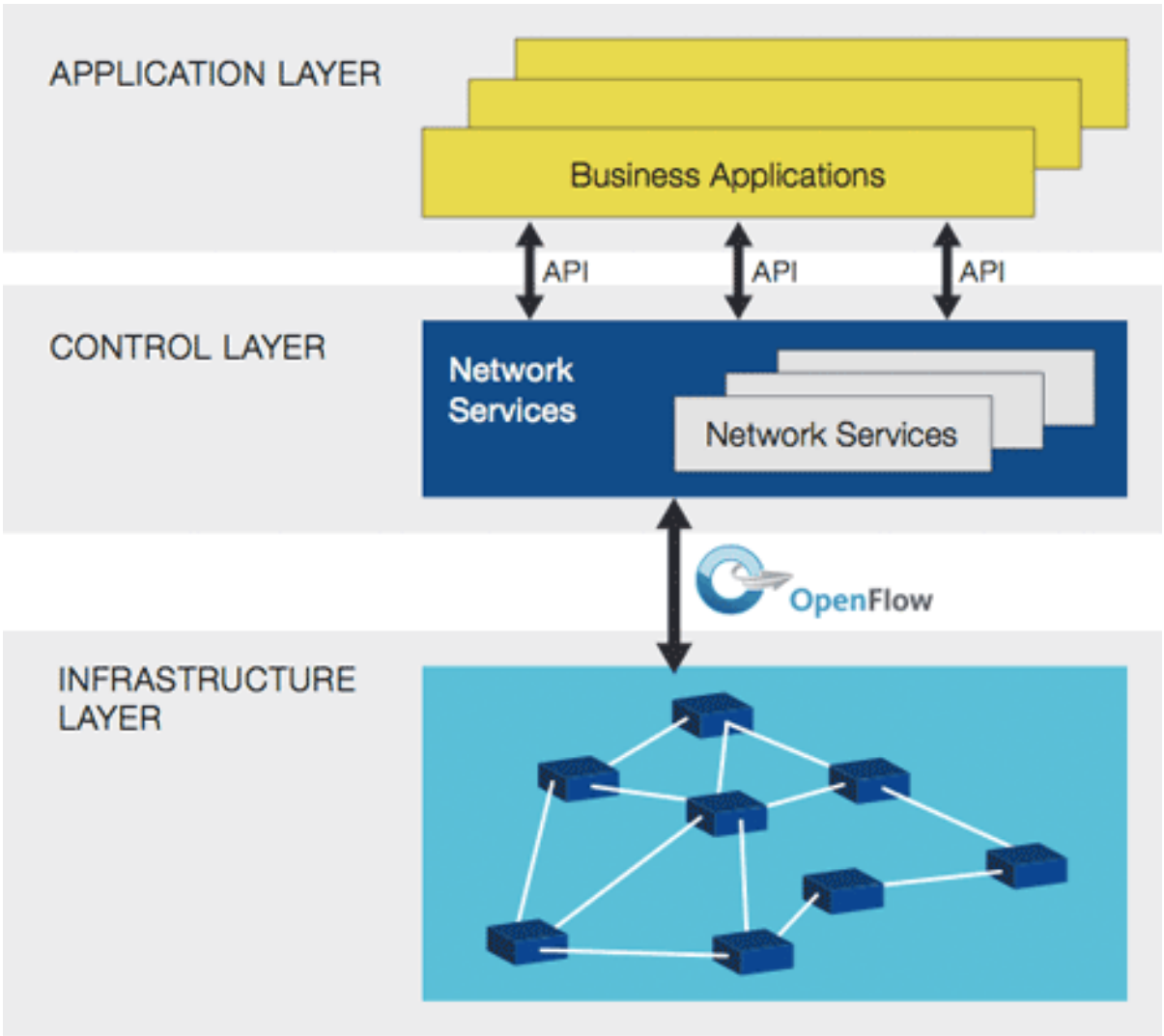

Figure 8 :- Structure and planning of a Software Defined Network

A rigid firmware serves as the platform for implementation of all the three planes. Firmware is the combination of hardware and the software. SDN separates the data plane with the control plane, this control plane is actually a centralized software in the network that works as a brain of the network. The vital role of the control plane is to build and maintain the routing table for each source-destination pair. Whereas, the vital role of the Data Plane is to implement that routing table and route the packet and maintain the flow over the link. The most beautiful feature of the Software Defined Network is the programmability of the control plane.

3.2 NEED FOR CENTRALIZED CONTROL IN DATA CENTERS

The new data centers require frameworks that give more control, adaptability and versatility than at any other time. You require more than server administration. We require a framework that decreases our Mean-Time-To-Repair (MTTR) while utilizing the current T assets you as of now have. In the event that we have to oversee server farms and branch workplaces at whatever time from anyplace, there is an answer. An incorporated administration approach without-of-band ability can expand server farm control, permit us to accomplish more with less, and give remote access from anyplace on the planet. Today's backing minded system managers need day and night access to servers 365 days a year - both at the rack and in remote areas. Luckily, with brought together administration (counting both in-band and out-of-band availability), data centers has never been so close. The use of central controller to manage flows gives us IP availability and security models in your system. Unified administration programming gives you a chance to control a machine that can be associated with servers, serial gadgets, even power dissemination units and ecological checking. With brought together administration, we can control different servers and system gadgets regardless of where they are found. Virtual media is a later expansion. It expands head effectiveness by permitting us to lead a vast host of errands remotely, for example, record exchanges. Whether we require access at the rack, in the Network Operating Center (NOC), or from our portable workstation at home, there are arrangements accessible that will give uncommon, secure get to and help you demonstrate a brisk profit for your venture.

3.3 POX CONTROLLER

Definition - "POX is an open source development platform for Python-based software-defined networking (SDN) control applications, such as OpenFlow SDN controllers. It has a simple OpenFlow Data plane implementation. It can be tested using OF Test framework. To connect data ports of POX switch implementation to OF Test Linux virtual Ethernet Interfaces (VETH) are used. And a default controller port 6633 is used for Openflow connection."

Explanation - POX controller is made of main components like additional python programs that are invoked as soon as the POX controller is started in the terminal from command line. All these components implement the functionality of the network in SDN. Running POX controller is similar to running the openflow SDN controller on the network topology. POX controller has reusable sample components for path selection, topology discovery, etc. POX controller runs on mininet. Mininet is a network emulator that is utilized for emulating Software Defined Network. All the servers and the switches are to be downloaded in the mini-net in the form of the in-built packages. Mini-net has a openflow controller and software in-built in it. Mini-net can be downloaded on all the Operating systems like Ubuntu, Linux/Unix, etc.

Steps to emulate POX controller with a user built topology in mininet.

- Step 1 – In the terminal 1, we follow this step and run out topology to start the network.

  we start with Mininet on our Operating System. Then creating a topology for SDN using any one control software like open Day-Light or Floodlight.

  Commands-

  a) *sudo mn* - This command starts the network.

b) *sudo python our_topology*.py – This compiles and starts the network topology built by the user. Hence the topology is emulated in one terminal and now we know that this a SDN topology so we have to give a controller to the topology so that it can learn the whole network and start building the routing tables to finalize the best paths and start communicating.

- Step 2 - In second terminal, we have to assign the controller to the emulated topology and make the SDN more efficient resulting for a better performance.

  We open terminal 2 and run the POX controller installed in Mininet.

  Commands –

  a) cd pox – To locate the appropriate directory where pox controller is stored.

  b) ./pox.py forwarding.L2_learning – This initiates the POX controller.

  c) sudo python our_topology.py – This compiles and starts the network topology built by the user. Hence the topology is emulated in one terminal and now we know that the controller is initiated.

  d) Sudo mn --controller = remote, ip=10.0.0.12, port=6633 – This indicates Mininet that a remote controller is active and this is to be used on our topology. As soon as it starts we see that it is listening for switch connection on the port number 6633 (this can be any port number above 1024).

  Now the network is on and it is controlled by POX controller which means that all the access points and forwarding device will learn about the neighbors and their neighboring connections. Eventually all the devices and hosts in the network will learn about whole topology and figure out the best paths, successive best paths with the help of individually maintained forwarding table. This network is now ready to route the traffic to and fro in the topology. The connectivity and

reachability in the network can be checked by ping command. Command *pingall* is used to ping all the hosts or the nodes connected in the topology and checks the network layer status of the host as well as the reachability to the host. We can ping the nodes individually as well using the command – h1 ping h2. What this command does is, it pings the host h2 from h1 and hence the reachability between host h1 and h2 is tested. If you capture the packets using any network monitoring toll like Wireshark or GNS3 and on close observation, we can notice the path that these ping packets follow. This derived path is the best path that the topology feels between host h1 and host h2. When we stop the POX controller we still notice the ping packets on the command prompt and this is because these ping packets are cached in the memory and they will go on until the hard timer goes out.

3.4 OPEN VSWITCH

Definition - "Open vSwitch is a production quality, multilayer virtual switch designed to enable massive network automation through programmatic extension, while still supporting standard management interfaces and protocols. In addition, it is designed to support distribution across multiple physical servers similar to VMware's virtual Network distributed vswitch"

Explanation - Open vSwitch is basically an open source implementation of the switch that is based on OpenFlow protocol and it is to be used as virtual switch in the virtual networks. OpenFlow particulars are focused at Layer 2 and layer 3 usefulness. The most recent networking movement is to empower a switch with Layer 4 to Layer 7 administrations like proxies, Load balancers, Firewalls, IPSec and so on Which makes the center boxes repetitive in the designed network. Hence, we propose an approach to broaden the most normally used Open vSwitch to Layer 4 to Layer 7 administration mindful OpenFlow switch.

3.5 IMPLEMENTING ECMP IN SDN

There can be such scenarios where multiple best paths are derived with same metric cost for obtaining the MAC address in particular. In the implementation of ECMP in SDN, the controller defines and implements the rules in all the openflow switches in the case of multiple number of best paths identified in the forwarding tables to use all the paths instead of just one best path for that particular MAC address. This is where we need to educate the controller with the rules that are of conjunction on the destination MAC address. POX controller, Flood light controller or Open Daylight controller can be used for this purpose and the methodology for educating the controller and making them understand the rules are different for all of them. Other measure which needs to be taken care of is that all the switches in the underlying data plane should be supporting openflow protocol and such switches are normally known as open vSwitches.

Details for implementation of ECMP in SDN.

- Each switch in the network is represented as GraphNode object.
- All the switches maintain the forwarding table with the entries of minimum cost paths for deriving the MAC address of the destination. Which needs Dijkstra's algorithm to be slightly modified for maintaining this sort of forwarding tables. As soon as we derive a new best path (with low cost metric than the last best path), the last forwarding table which was maintained by the switch is discarded. The switch builds and maintains a new forwarding table and starts with the first entry with the best path that it just learned. If we now discover any new path with same metric and cost, will be kept it in the table and this makes the table with multiple best paths. This table of paths with all the collection of rules in all switches are hard-state. This means that they are cleared explicitly and these do not timeout.

- As soon as a new host or switching device is added or subtracted from the topology, all the rules which were defined in the network prior will be discarded and all the rules are recomputed in all the switches. These rules are not yet installed in the whole topology. Only the nodes which are inspected by the controller are installed as of now. In the next step we show the exact procedure of how these rules are installed in each and every Graph Node inside the topology.
- Firstly, when any packet reaches the switch and realizes that there are no rules defined on the node as of now. This packet if forwarded to the controller as an initial step. Secondly, the controller then checks for the number of best paths that the switch has for any particular destined MAC address. If there are more than one best paths available for the destination MAC address, then the controller selects the path with the least used port. This rules is then matched at the destination and the source MAC address. This does not mean that all the packets with different destination MAC address will be routed from this path but it surely means that the packets for this same pair of source and destination MAC address is to be routed from this path. As a final step to define rules on the entire topology, the controller has to forward the packets to next hops on behalf of this considered switch.
- All the nodes have to keep track of counts for all of their ports which are to be used in deriving the rules. To balance the load evenly on entire network, it is important that the controller decides the least used port and allots the traffic to that port. This is most useful when we have a highly busy source which communicates with many destination MAC addresses. To avoid the conditions like using the path between two hosts supposedly, h1 and h2 which is better but we still have a better path which is less used and gives a good performance between these two hosts. Hence the basic rule in the controller to be defined must be only to choose the best path between two hosts and allot the traffic on that link.

- The final and most important thing to be taken care in entire process is that the rules never timeout. The controller will clear all the old rules defined in the network and it will re-define all the rules again considering a single switch at a time.

CHAPTER 4 PROJECT GOALS AND TOOLS

4.1 GOAL OF PROJECT

The basic idea of our project is to analyze multi path in data center by running the ECMP protocol. To achieve these goals, we conducted two experiments and depending upon the results we evaluated the outcome.

The fist experiment involves analysis of ECMP in data center topology. In this scenario we run ECMP Algorithm in the data center network and observe the behavior of ECMP in fat tree topology. ECMP is a multi-path routing protocol. If there are two or more paths with equal cost path it performs load balancing on all these equal cost path.

In the second experiment we perform the same experiment but in an SDN based environment. SDN here provides the use of central controller which manages the flow forwarding based on the bandwidth utilization of core links. The controller here acts as the brain of the network. The controller can perform dynamic scheduling based on the bandwidth requirement of the core links. The controller used here is POX controller.

Based on the two experiment we evaluate the results depending on the bandwidth requirements. We use IPERF to generate traffic on the links and based on the bandwidth utilization evaluate the results.

4.2 TOOLS

The Fat Tree topologies are outlined utilizing Python scripts which call upon the Software Defined Networks Simulation device Mininet for making of the topology. It is a copying situation which depends on the idea of holders and is broadly utilized for testing applications as a part of a SDN domain by sending them in a system of virtual switches and has, all of which are running with the assistance of assets of the same physical machine. Mininet runs programming applications on the virtual has and switches which can then be utilized to test organizing programming. SSH convention can be utilized as a part of mininet to send bundles from source to destination in the system through virtualized Ethernet interfaces and virtualized joins. Mininet is a helpful apparatus to imitate distributed computing and server farm systems as SDN outlined information streams can be sent to OpenFlow Switches made with a mininet topology for sending of parcels. Virtual Hosts in mininet can be made to copy a virtualized distributed computing environment. Mininet underpins OpenDaylight, Floodlight controllers and can likewise be utilized alongside Remote Controllers, for example, Pox. [7]

For our task, we have made modified Fat Tree topologies utilizing python scripting dialect. For Each topology we have made servers, Edge Switches, Aggregate Switches and Core Switches. We have characterized the naming tradition for the Switches and Hosts, characterized their interfaces and characterized joins between hosts-Edge Switches, Edge Switches-Aggregate Switches, Inter-layer Aggregate switch linkage and connections between center total layer switches. We have made scripts for conveying every one of the servers and switches that are required for a specific Fat Tree topology taking into account Table 1 and estimations of K= 4, 8. We have chosen the Remote Controller choice while imitating the Fat Tree topology utilizing Mininet and specified the same as a variable in the python script while calling the Topology

creation capacity. To empower learning among the switches, we have characterized a capacity for Spanning tree convention and gave the ovs-vsctl gave framework level order to every switch in the design for empowering STP. This encourages the changes to find out about the Network speedier and thusly help with setting up end –to-end availability in such a monstrous system. The Remote Controller utilized is POX. Further insights about POX will be examined later in this Chapter.

4.2.1 SPANNING TREE PROTOCOL

Definition – "STP convention makes a loop free topology amongst switches, furthermore keeps TV issues that outcome from circled systems. The protocol makes a tree amongst a system of interlinked switches and makes the briefest way from the root switch to all the switches in the system, in this manner making the best way for availability to every host."

Explanation - A Hub advances all parcels it gets to the greater part of its ports, as it doesn't have any system knowledge to recognize the destination address (IP or MAC) of the bundle. Albeit straightforward and reasonable, Hubs diminish the throughput in the system and can bring about flooding on the end hubs. All hubs forward the broadcast coming on their port and this can represent a security danger in a server farm organize particularly if any hub is working in the indiscriminate mode. Span tree learning is a valuable route for working switches in a tree system, nonetheless, the inconvenience of Bridge learning is that it can make circles and the subsequent redundant casings revolving around the system can devour adequate data transmission to influence the execution of the system. STP tackles the issues made by both Learning Bridges and centers, as it effectively chooses a solitary port on every extension for every information way for bundle sending in view of "root" scaffold with the expectation that every hub will locate the most brief

way to the "root" span in light of switch area when contrasted with the root switch: the way with the change nearest to the root extension is chosen as the assigned way and the relating port as assigned port and all different ports on the scaffold from the same hub to the root extension is crippled. Be that as it may, the ways regarded repetitive are incidentally debilitated, as when they chose way comes up short, the excess way is brought back online by recalculating the way to the root span.

4.2.2 OS SYSTEM COMMAND – OPEN VSWITCH - OVS

OVS switch is open source multi-layer switch. The switch gives a changing stack to virtualization and traditions use. It is a to a great degree suitable switch for the virtual framework environment. Ovs-vsctl joins with an ovsdb-server handle that keeps up an Open VSwitch setup database. Utilizing this association, it questions and perhaps applies changes to the database, subordinate upon the supplied summons. By then, on the off chance that it related any developments, according to common it holds up until ovs VSwitch has finished the procedure of reconfiguring itself before it exits. The switch performs various charges in a singular run operation. Here is a sentence structure of ovs-vsctl switch:

ovs-vsctl [options] - [options] summon [args] [- - [options] request [args]]

The ovs-vsctl demand line starts with overall decisions. The overall decisions are trailed by one or more demands. Every request ought to start with - with no other individual as a charge line question, to separate it from the running with summons. (The - before the chief solicitation is discretionary.) The call itself begins with solicitation particular choices, accepting any, trailed by the charge name and any debate [3].

While completing our framework designing we use ovs-vsctl to support the platform executed by open VSwitch. The single augmentation sponsorships ports on different VLANs and it also supports range with given 802.1Q VLAN development. Using the Ovs-vsctl charge, we realize crossing tree computation in the switches in our plans to engage data way recognizing evidence.

```
cmd = "ovs-vsctl set Bridge %s stp_enable=true" % ("CORE" + str(x))
os.system(cmd)
print cmd
```

4.2.3 MININET REFERENCE CONTROLLER

The Reference Controller in mininet is arranged coherently above OpenFlow interface. The controller performs Ethernet MAC learning alongside OpenFlow Switching. As a matter of course it contains no stream sections. For each parcel it gets from the switches, it gathers data in regards to the MAC location of the switch, the source location and destination location of the bundle, VLAN IDs, port number on the change from where it got the bundle. Taking into account this data, it makes the stream table sections until it has gotten data from all the switches in the system.

4.3 MININET TOPOLOGY CREATION

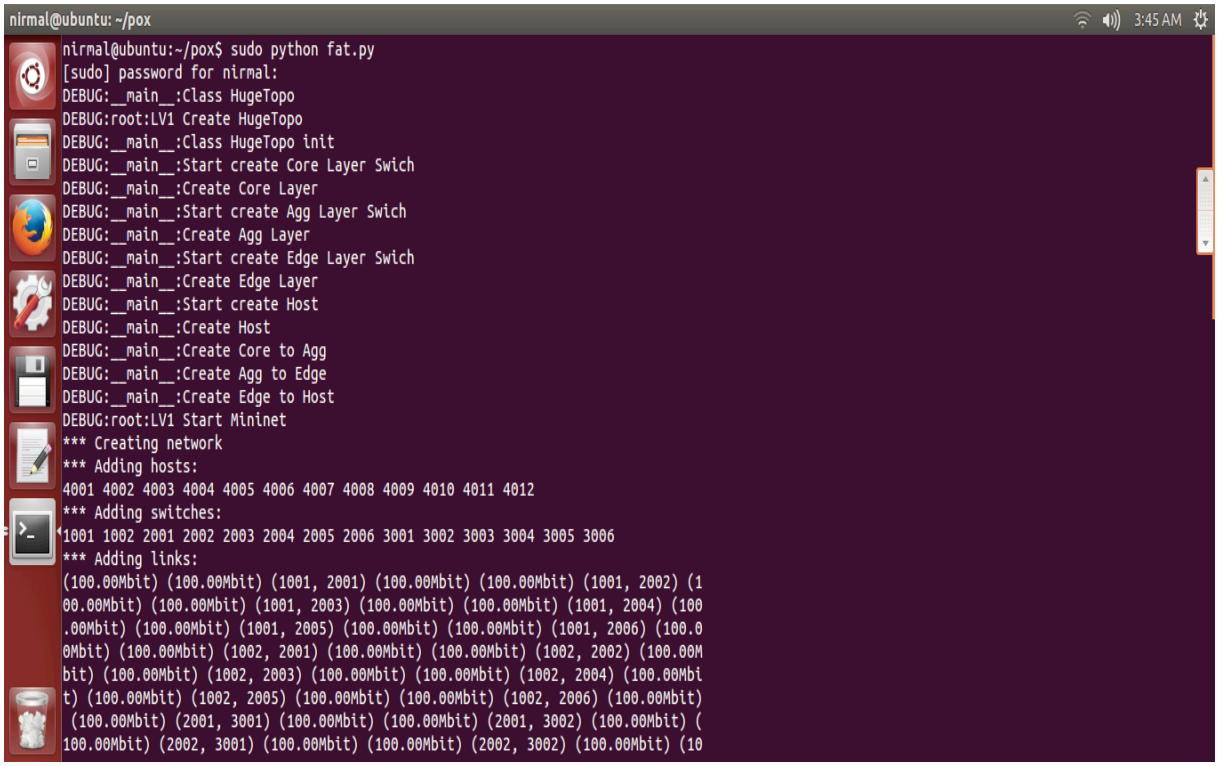

Figure 9 :- creation of Topology in Mininet (1)

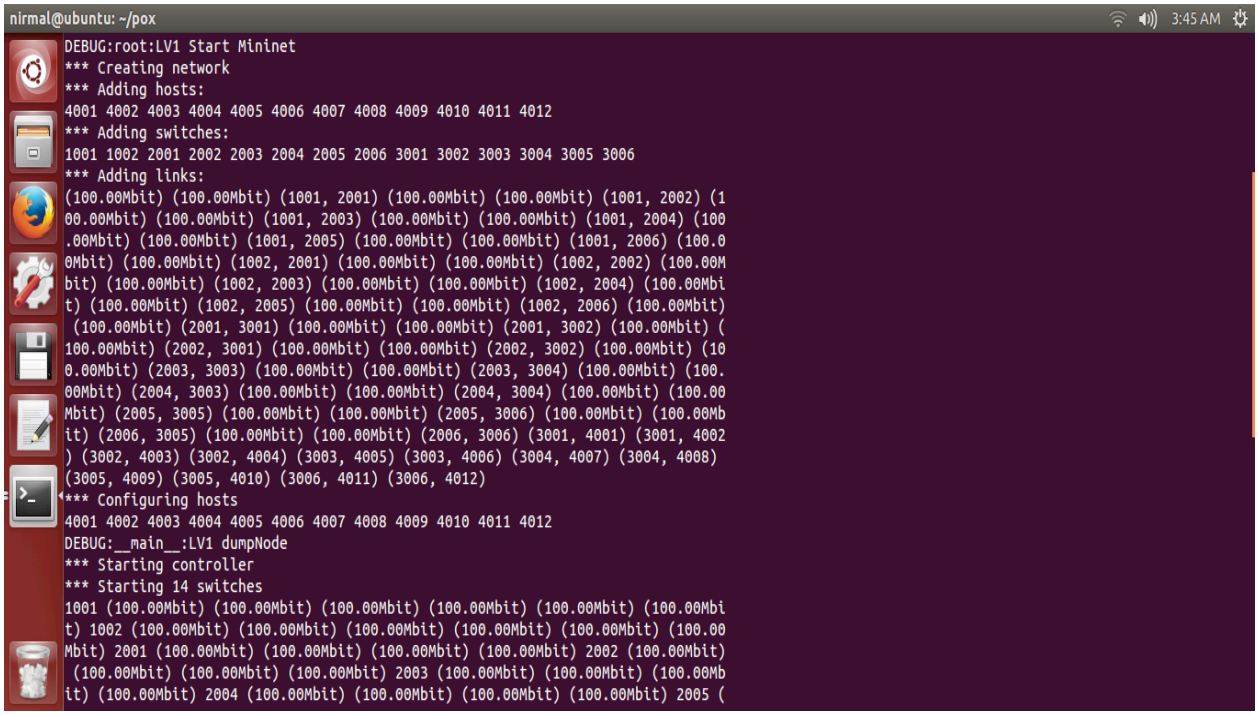

Figure 10 :- creation of Topology in Mininet (2)

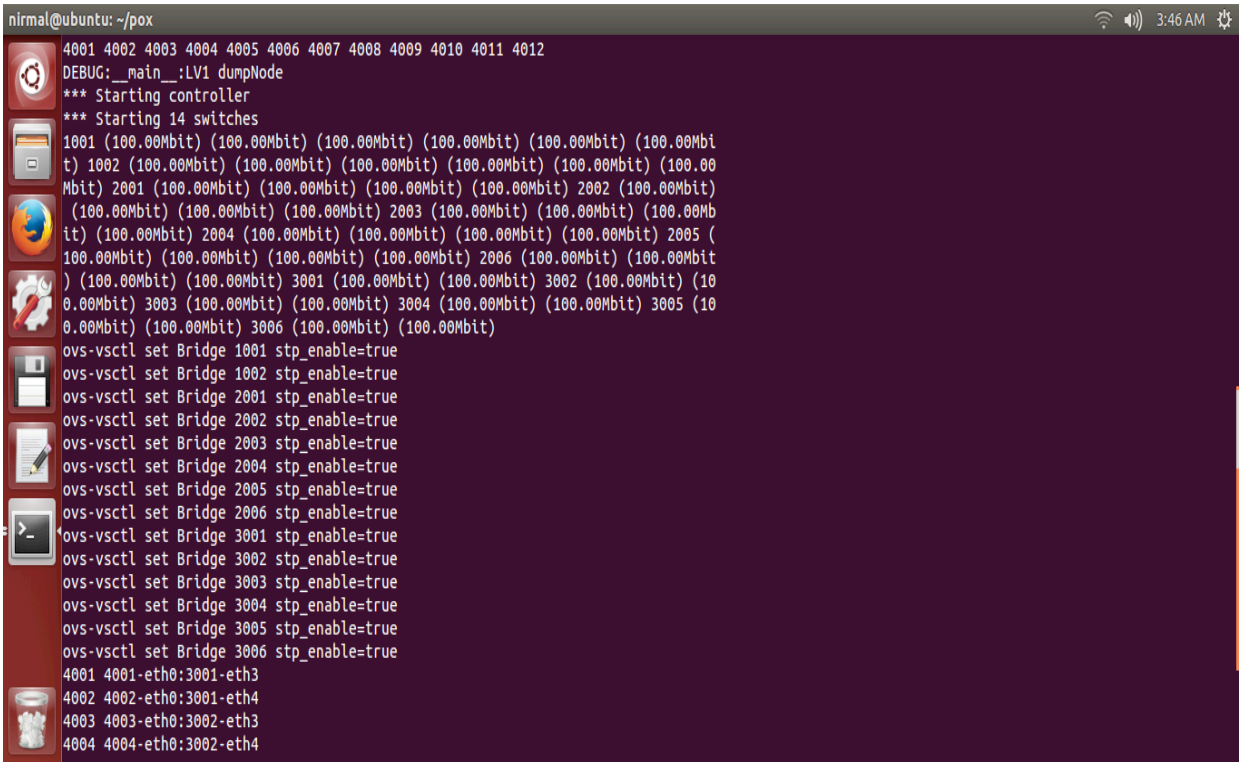

Figure 11 :- creation of Topology in Mininet (3)

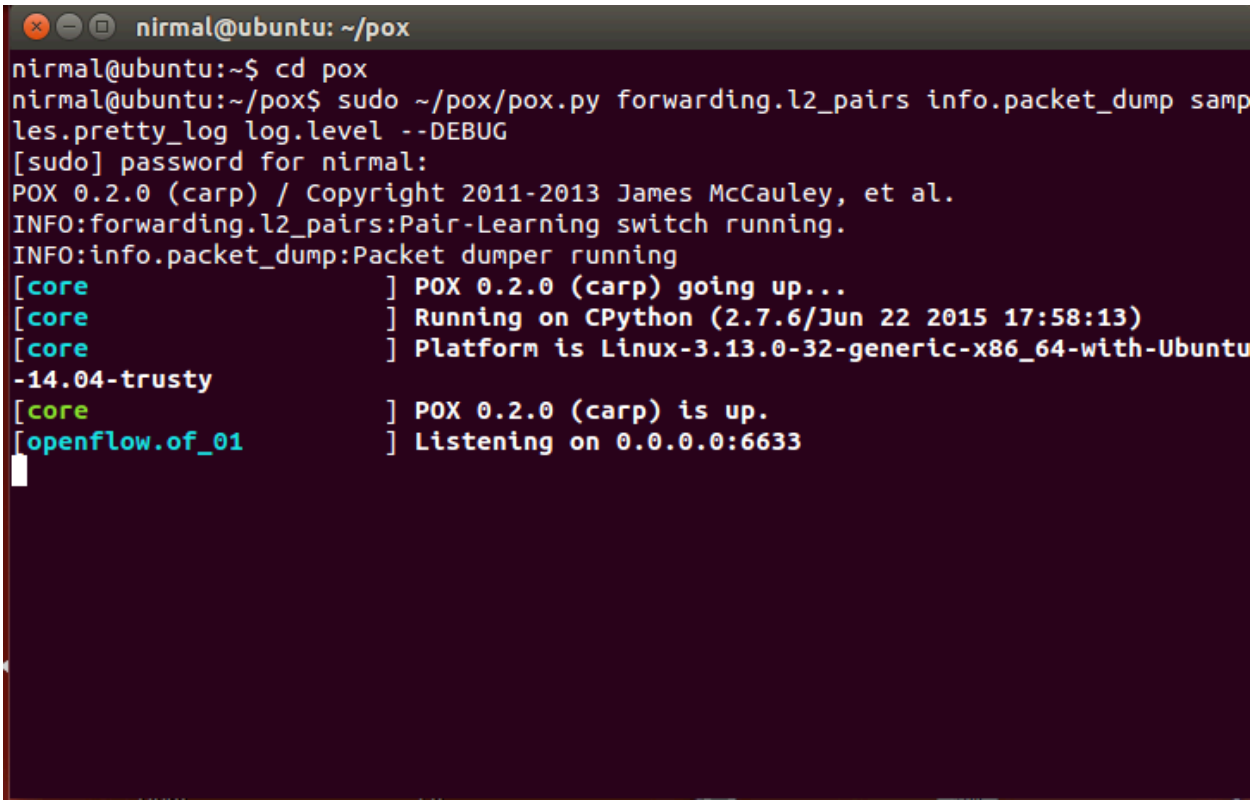

Figure 12 :- creation of Topology in Mininet (4)

4.4 WORKING OF FAT –TREE ARCHITECTURE CODE

4.4.1 CODE FLOW OF FIRST EXPERIMENT AND SECOND EXPERIMENT

After the execution of the program, the interpreter checks the main function and checks the users. If, the condition of user being a root user is true then the function to create the topology is called. At the beginning of the program a variable class is assigned in the create topology function.

- A class Multi Topo is created in which list is created for core, aggregate and edge layer switches and hots. The list is defined in the following way in the function *self.<component List>( )*.
- The arguments to the functions are passed as number of switches and host and are called which than creates the number of switches and host as defined. The values of number of host and switches to be created are passes as *self.<component>( )* .
- The switches and the host created on calling the function gets appended in the host and switch list created earlier. The above task is achieved by creating a function. *self.addSwitch()* and *self.addHost()* are used as the key componets to create the switch and host and append them to list.
- After that the function is called in the class to create the link between the core switches, aggregate switches, top of rack switches and hosts. *self.addLink()* command is used to create links between the switches and Hosts.
- *net.addController()* Function is used to create a SDN controller and IP and port number are assigned to the controller. The IP address assigned is 127.0.0.1 and port number is 6633.
- *net.start()* is used to start the controller and activate the core, edge and aggregate layers.

- Then we start the spanning tree by giving the command "*ovs-vsctl set Bridge %s stp_enable=true* (%s is the name of the switch)".
- To start the command line in the mini-net after creating the topology we used *CLI()* which provides an interface to talk to the nodes.

4.4.2 **THE FLOW CHART FOR THE FIRST EXPERIMENT**

In the first experiment we first create a fat tee topology in mininet. The topology consists of three layers core layer at the top followed by the aggregate and edge layer. The edge and core layers are connected to each other. All the host in the topology are connected to the edge layer. In our scenario each edge layer is connected to 2 host. Then we add links between the switches and host. The links are created between core and aggregate switches and aggregate switches are connected to edge switches. Each edge switches than connect to two hosts. After defining links in the topology we run STP. Then to perform multi-path routing we run the ECMP algorithm. ECMP defines the path each host takes to reach other host. Each path passes through the core switch. To test the connectivity between switches and the host we run ping and to generate the traffic on links and test the bandwidth between the links we use IPerf.

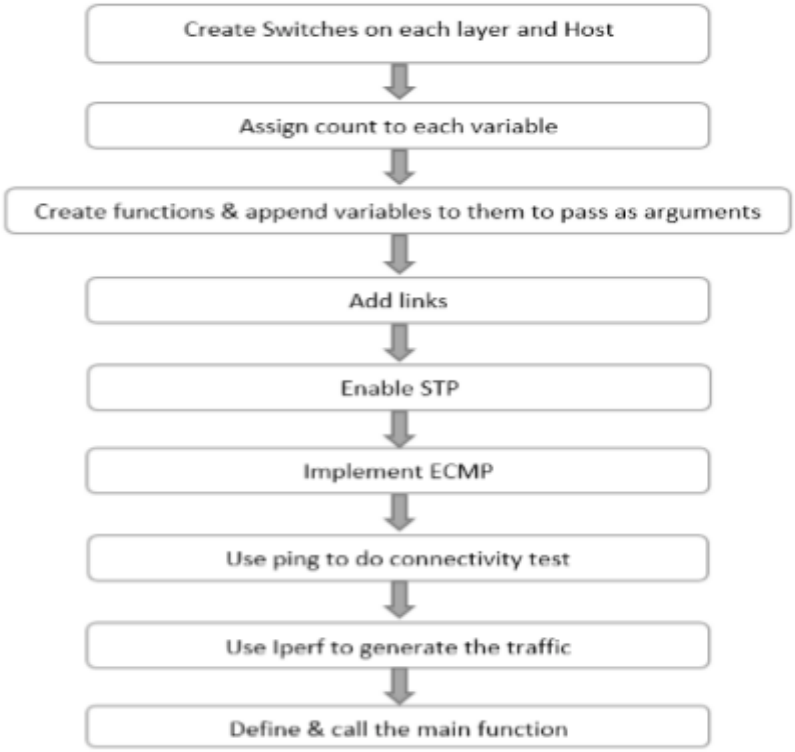

Figure 13 :- Flow Chart for Experiment 1

4.4.3 THE FLOW CHART FOR THE SECOND EXPERIMENT

In the second experiment the topology remains the same. The same fat tree topology created is used for the second the experiment. The only difference is that the core switches are connected to a central controller. The central controller here manages the flows based on bandwidth utilization of core links. The controller we used here is POX controller. The controller here dynamically manages flows. The SDN based environment solves the drawback of normal fat tree topology. When the bandwidth utilization of the core links exceeds the threshold the controller here dynamically adjust the flows. It directs the exceeded the flows to select a path which is used less. Thus the use of controller overcomes the drawback of ECMP in fat tree topology.

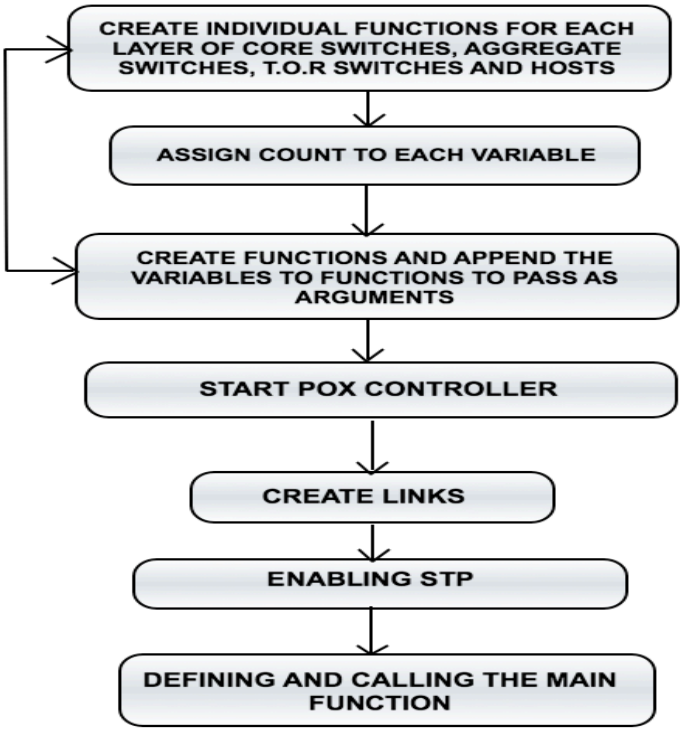

Figure 14 :- Flow Chart for Experiment 2

CHAPTER 5 PERFORMANCE EVALUATION RESULTS

5.1 SIMULATION AND EVALUATION

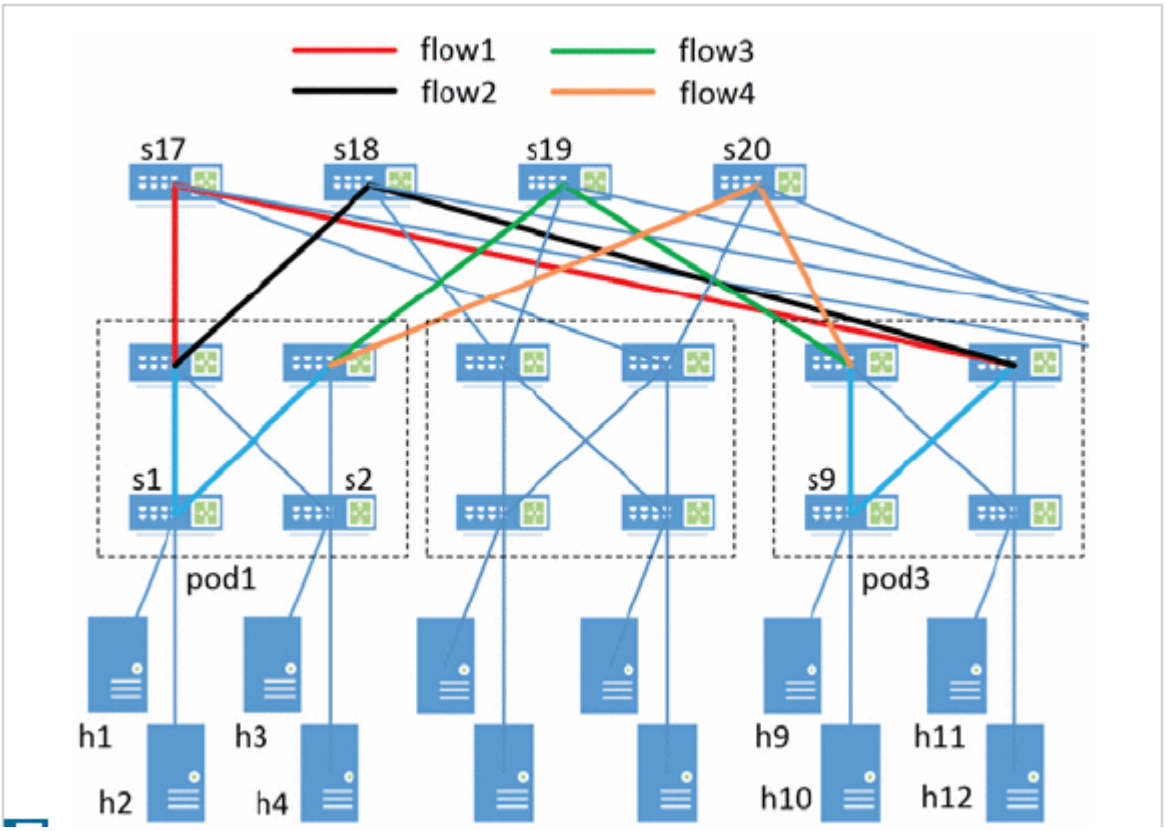

Figure 15 :- Final topology for DCN

To compare the performance of our two experiments we select host destination pairs to different edge switches and measure the bandwidth utilization. For every host there are four equal costs paths as shown in the figure there are four equal cost path between s1 and s9. To perform the evaluation results we transmit traffic between pair's h1-h9, h1-h10, h2-h9, h2-h10, h3-h9, h3-h10, h4-h9 and h4-h10. To compare the performance of different experiment we calculate the bandwidth utilization at the core switches. To generate the traffic on this flows we use Iperf . Some

host are set as server and clients and traffic is sent between. The basic idea is to calculate the bandwidth utilization on core links evaluate the performance.

5.2 Traffic Design and Measurement

5.2.1 IPERF - Iperf traffic generation tool,

IPERF is an Internet Performance working group which is generally used to create and a lot different types of traffic in the network topology. We are using IPERF to generate UDP traffic as to start with. The main reason for using IPERF is to actively measure the maximum achieved bandwidth on the Data Centre Network which we have built. IPERF also helps in the tuning of different sort of parameters relevant to that of Timing, Buffer handling and protocols like Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Stream Control Transfer Protocol (SCTP) with both IPV4 and IPV6. The most useful results that IPERF provides is that every test reports the parameters like loss, Bandwidth, etc. Some of the IPERF versions for generating traffic are backwards compatible and some are not.

For TCP and SCTP, IPERF results consists of the the supporting TCP window size through the buffers of sockets. Secondly the results consist of measured Bandwidth and returns the size of MSS/MTU. Whereas, UDP packets are created and the streams are created out of those packets of particular bandwidth and then we can measure different aspects like Packet loss, Delay jitter and whether it is capable of Multicasting.

IPERF traffic allows the server and the client to have multiple simultaneously ongoing connections. It also supports multiprocessing in the server that is, it can handle multiple clients one after one and the server never stop until is commanded to do so.

How do we define TCP or UDP traffic in IPERF – There are several options available for playing with the IPERF traffic generation. Some of them which are used in this research project are explained below.

- iperf –s [options] - *Runs the TCP traffic in Server mode with fix sized streams after handshake is done.*
- iperf –c server [options] - *Runs the TCP traffic in Client mode, connecting IPERF server which is activated on host.*
- iperf –u –s [options] - *Runs UDP packet streams on the Server mode*.
- Iperf –u –c server [options] - *Runs the UDP traffic in Client mode, connecting IPERF server which is activated on host.*

A typical Iperf Command is as follows

- *Mininet> iperf <Source Host Name> <Destination Host Name>*
- **** Iperf: testing TCP bandwidth between <Source Host Name> and <Destination Host Name>*
- **** Results: ['1.69 Gbits/sec', '1.70 Gbits/sec']*

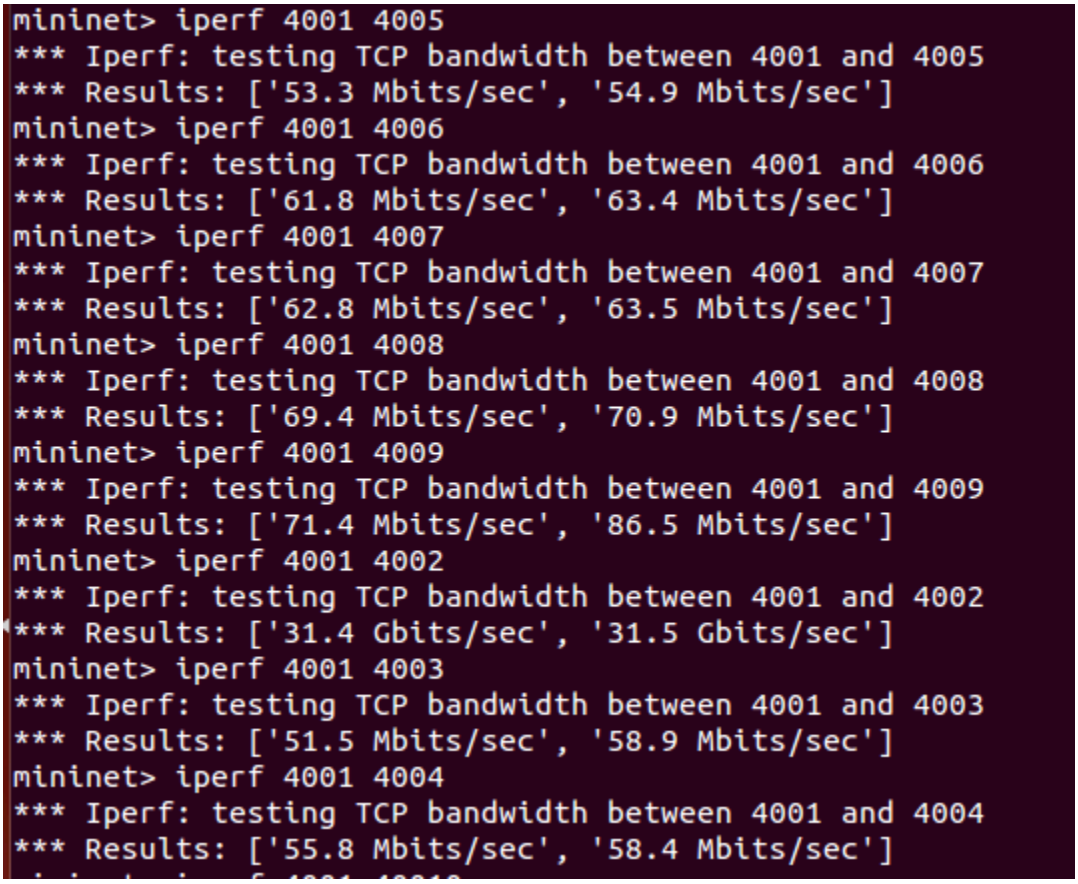

Figure 16 :- Results of IPERF

5.2.2 PINGALL AND PINGALLFULL

Pingall command is generally used to ping all the hosts, switches and other network devices connected in the network to check the connectivity between all of them. There might be the case sometime that some host are reachable from some hosts and others just have some problem finding the path or less bandwidth or any other reason from millions of them. We want to ignore the hustle of trial and error method to ping every host from host 1 and again all the other hosts from host 2 and so on, for all the number of hosts. *pingall* is the best solution for such kind of situations and it can be encroached on the whole topology at once. The result for pingall command shows many information if you look at it closely and you will find,

If the *pingall* command is successful.

- We obtain a list of the following analysis as Host_name, IP_address, number of messages sent, number of messages received back from the host, Round Trip Time (RTT) in milliseconds, etc.

If the *pingall* command is unsuccessful.

- When this command did not run successfully that means one or more of the nodes in the network is not reachable and the output for this command on the terminal will display an error message. All the error messages must be about different failure points in the network.
    - **100 % packet loss** – If we see this message on prompt then it can mean,
        - Host in the network is down.
        - Host is active but the packets are denied at the host.
        - The network itself is down.
        - Probably, wrong host was pinged.

- **Packets are rejected** – this says that the packets are reached ta the host and are also read but they are rejected at the host.
- **Packets did not reach the host** – There can be a problem with the connecting link, hence the packets are sent out proper and the receiving host is all set for accepting it but the network fails to deliver the packets at the host.

Output for both PINGALL and PINGALLFULL

Mininet>pingall

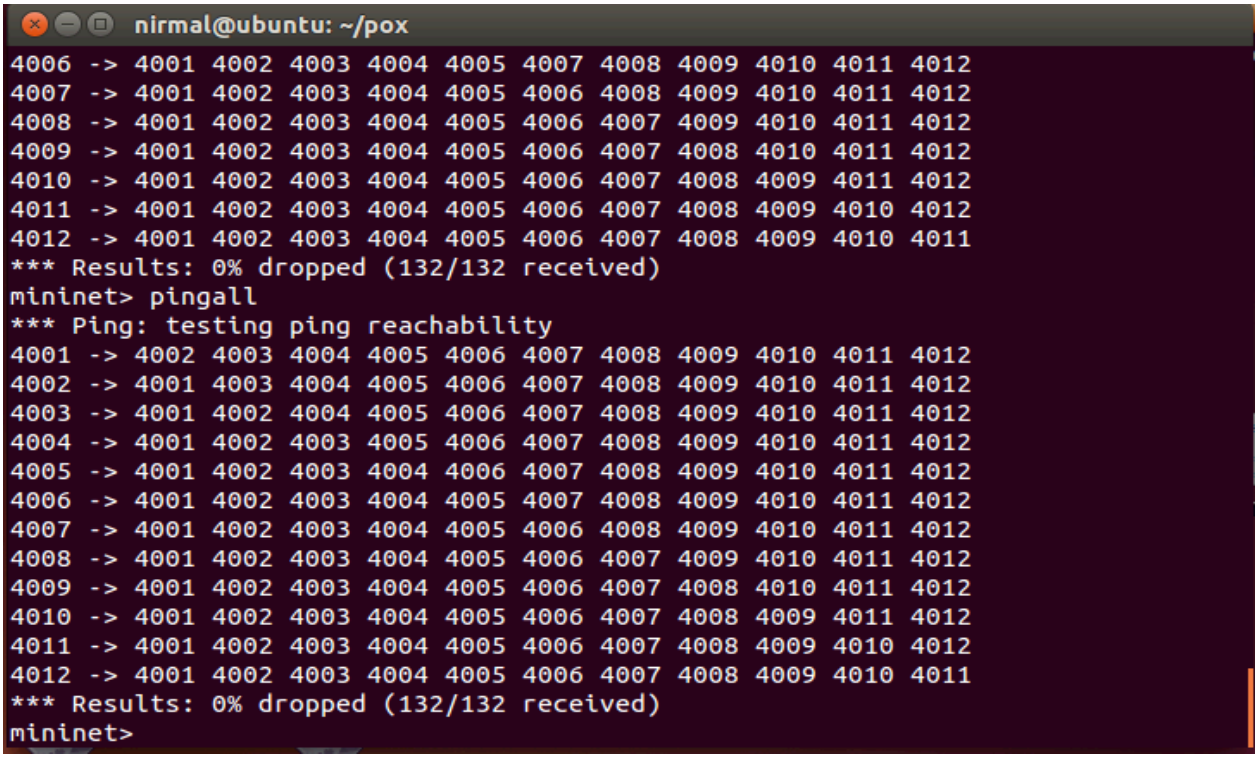

Figure 17 :- pingall output on mininet cli

Mininet> pingallfull

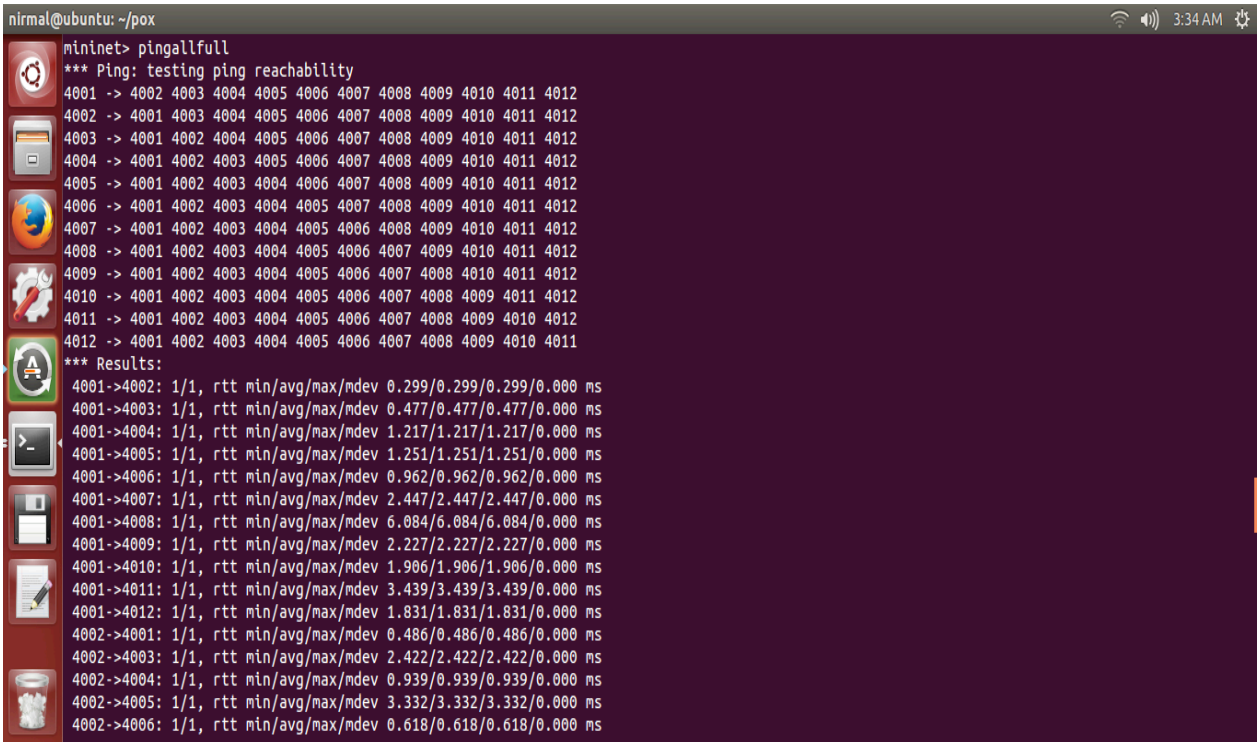

Figure 18 :- pingallfull output on Mininet cli

The outcome is that each host in the network pings every other host in the network to test reachability and response time. The average response time can be considered as average delay between the source and destination.

5.2.3 TRACEROUTE

Traceroute is the direct command and does not require any options or parameters. It can be defined as a "computer network diagnostic tool for displaying the route (path) and measuring transit delays of packets across an Internet Protocol (IP) network. The history of the route is recorded as the round-trip times of the packets received from each successive host (remote node) in the route (path); the sum of the mean times in each hop is a measure of the total time spent to establish the connection. *traceroute* proceeds unless all (three) sent packets are lost more than twice, then the connection is lost and the route cannot be evaluated. Ping, on the other hand, only computes the final round-trip times from the destination point."

How does traceroute work –

It sends the UDP packets in sequence by default to the destination host. But we can also alter it with TCP or ICMP Echo Request. This scenario is for Linux but it is a little different in windows, traceroute uses ICMP Echo Request preferably in place of UDP. But again, you can manually assign with TCP or UDP. The metric for traceroute is hop limit which is also known as time-to live (TTL) value which shows the behavior of the intermediate routers between host 1and host 2. Every time the traceroute packet arrives at the access point in the network, it will reduce the TTL value by one and route it forward towards the destination. In this case, as soon as the TTL value for any packet reaches zero that packet is discarded and the error message is sent back to the same route from which it was coming. Common TTL value in Windows is 128 and in Unix-based OS, it is 64.

Usually to start with the packet forwarding, any OS starts with the TTL value as 1 and gradually increasing it to the respective value according to the OS. These packets are dropped at first router in the starting. Second packet will be dropped at second router. This is how we find out the TTL between the two hosts and the whole route between them is traced.

Mininet> Xterm <node name>

Xterm # traceroute <destination IP>

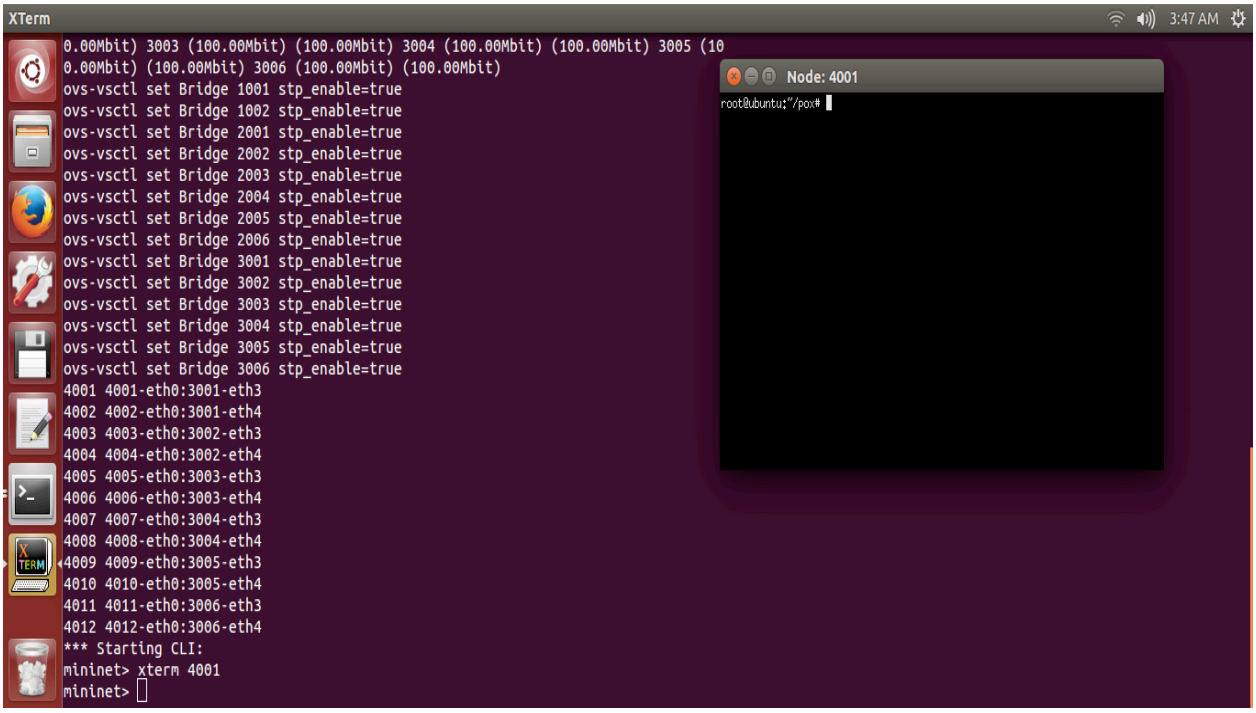

Figure 19 :- Initiating traceroute from Host 0

5.3 ALGORITHM EVALUATION

To evaluate the results, we run two experiments in data center networks based on different environment and compare the results based on bandwidth utilization of core links. In the First experiment ECMP is used in a fat tree topology and results are evaluated and in the second experiment SDN based environment is created and results of both the experiment are compared. Traffic is passed on the flows in host destination pairs using Iperf is used to generate the traffic and bandwidth is evaluated on the links. In the second experiment a threshold is set and when the traffic exceed the threshold controller dynamically selects the flows.

5.3.1 FLOW CHART OF TWO EXPERIMENT

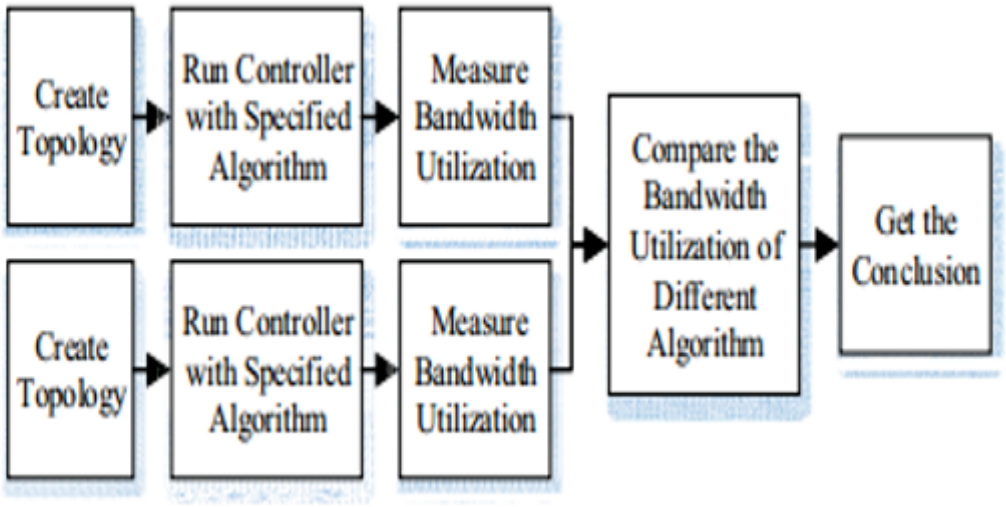

Figure 20 :- Flowchart for both the Experiments

5.3.2 FIRST EXPERIMENT

Topology of First Experiment

The possible bandwidth utilization is calculated as shown in TABLE 1. To evaluate the results we calculate the bandwidth utilization of eight flows. The table and graph shows the bandwidth utilization of flows in fat tree topology when ECMP is used.

DATA FLOW	**4001 - 4009**	**4001 - 4010**	**4002 - 4009**	**4002 - 4010**	**4003 - 4009**	**4003 - 4010**	**4004 - 4009**	**4004- 4010**
BANDWIDTH UTILIZATOIN	**0.4**	**0.2**	**0.4**	**0.2**	**0.4**	**0.1**	**0.4**	**0.1**

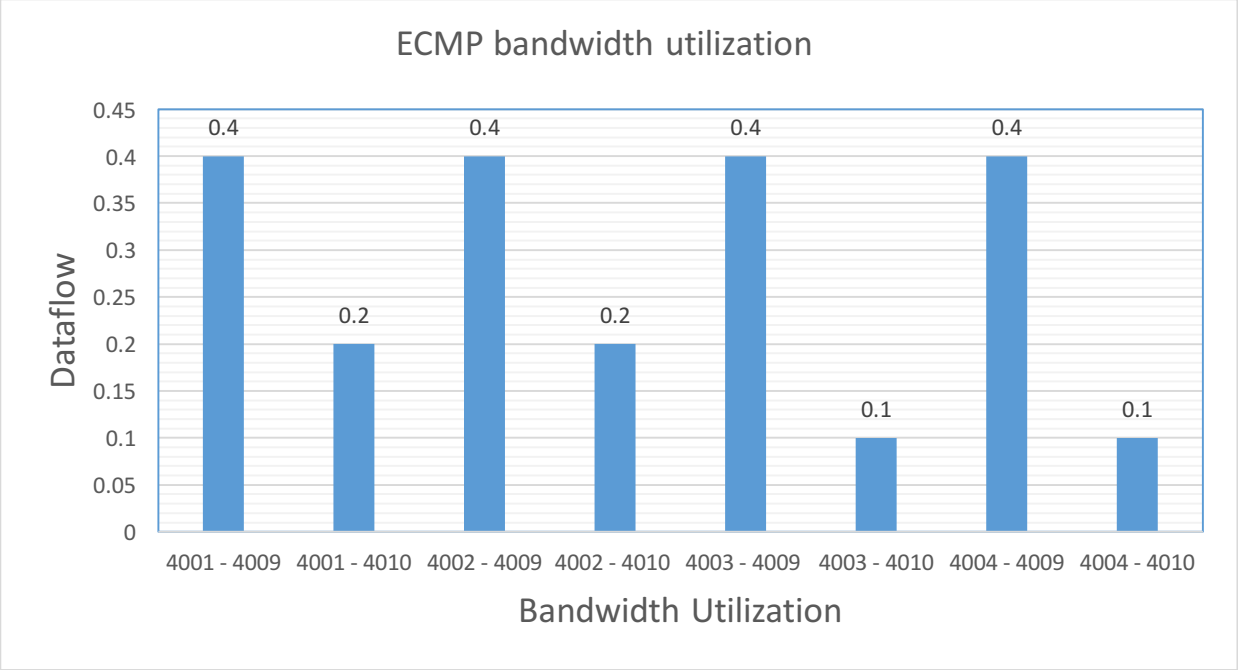

Figure 21 :- Table for possible bandwidth utilization of each access flow.

5.3.2 SECOND EXPERIMENT

Topology of Second Experiment - Flow Chart of SDN Based ECMP.

DATA FLOW	DATA FLOW	BANDWIDT UTILIZAION
4001 - 2001	4002 – 2002	0.4
2001 – 1001	2002 – 1001	0.4
1001 – 2005	1001 – 2005	0.4 + 0.4 = 0.8
2005 - 4009	2005 - 4010	0.4

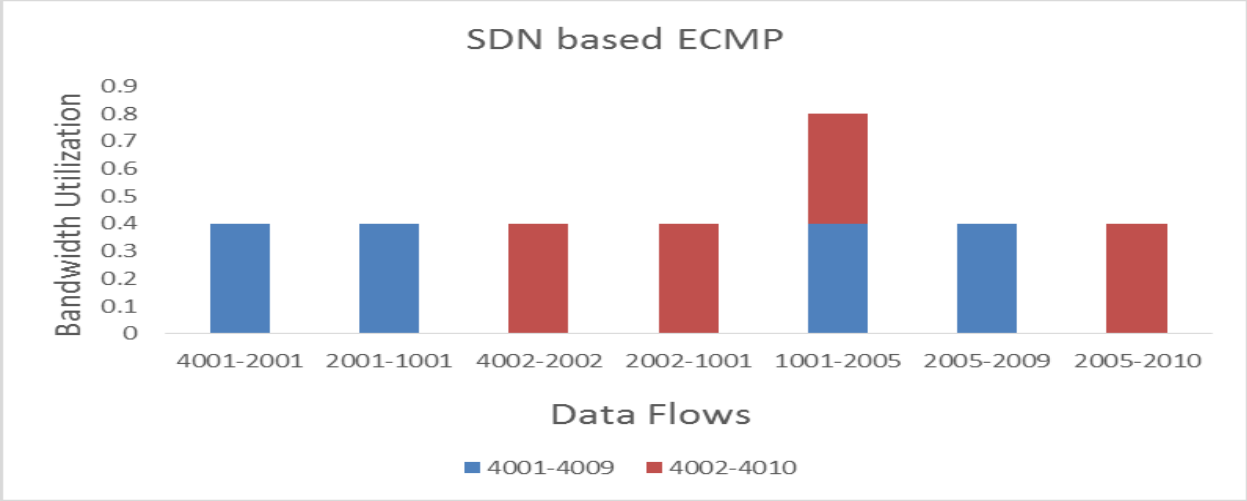

Figure 22 :- Table for possible bandwidth utilization in SDN based ECMP.

From the above graph and table it is concluded that the two flows 4001 - 4009 and 4002 – 4010 are forwarded through the same core switch 1001. So from 1001 core switch both the path are forwarded through the same path from switch 1001 – 2005. So the possibility is that bandwidth utilization of that core link from 1001 – 2005 exceeds the set threshold. The controller than activates and directs the flow to less utilized path.

5.3.3 FLOW CHART OF SECOND EXPERIMENT

The flow above states that first a fat tree topology is created in mini-net and ECMP algorithm is started. Flows are forwarded as per ECMP and bandwidth on these flows is recorded using the iperf tool. When the mini-net topology is executed at the same time on another terminal a pox controller is started. The results are evaluated based on the bandwidth utilization of core links. When threshold set exceeds the controller is activated and direct the flows to less utilized paths. If not than normal flow forwarding continues.

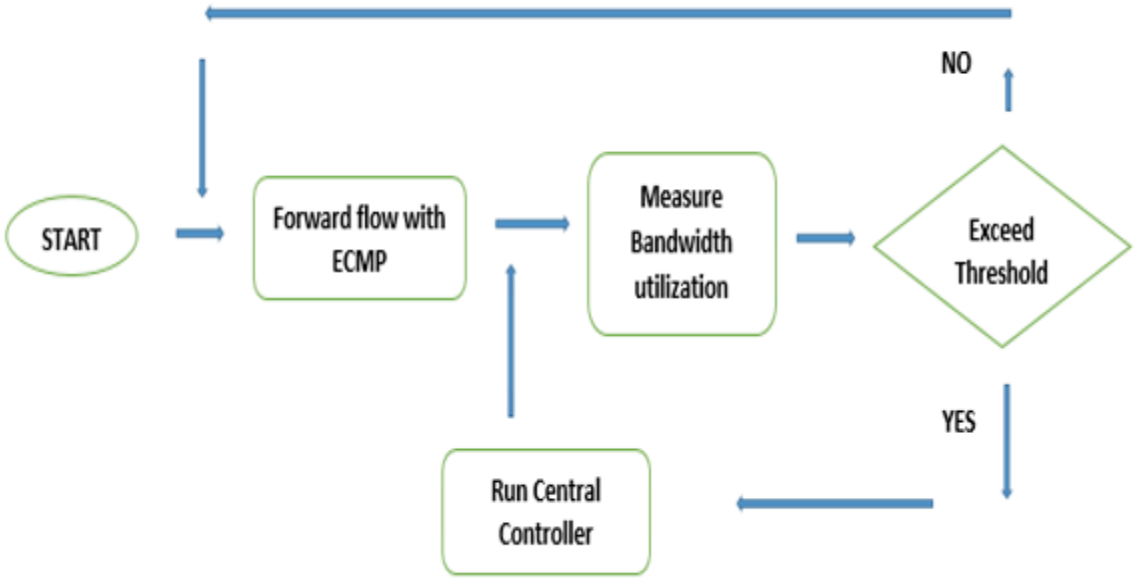

Figure 23 :- Flowchart for second Experiment (SDN based ECMP).

CHAPTER 6 CONCLUSION

CONCLUSION -

The limitations of data center topology are overcome by fat tree with multi-path routing with increased bandwidth and fault tolerance. To efficiently utilize bandwidth and traffic load we suggest a multi-path routing scheme with efficient load balancing capabilities. As a practical solution we suggest a use of centralized scheduler or a controller which manages the network information and forwards flows based on bandwidth utilization of core links. The controller dynamically select flows and makes optimized load balancing decisions when adding new flows to the network. With the use of ECMP in data center networks throughput of topology can be increased but the problem with ECMP is lack of flexibility and dynamic scheduling when selecting flows on equal cost paths. With the increase in the development of SDN we propose a solution which will outperform the issues of ECMP. The work proposed shows that the central controller will dynamically adjust flows when the bandwidth utilization of core links exceeds the threshold which increases the throughput and network performance. This also proves the fact that SDN based environment is suitable for data center networks. Thus it proves that with help of centralized controller it is possible to improve the throughput of data center networks. Experiment results show that our proposed work with the used SDN controller consistently outperforms ECMP in fat tree topology.

DIVISION OF WORK

DHARMESH BHANUSHALI

Chapter Number	Section Number	Title
Chapter 1	1.1	History of Data Center Technology
	1.2	Modern Data center Design
Chapter 2	2.1	Fat Tree Topology
	2.2	Architecture of Fat Tree Topology
	2.3	Network Utilization
	2.4	Multi – Path Routing
	2.5	ECMP
Chapter 3	3.1	What is SDN
	3.2	Need For Centralized Control
	3.3	POX Controller
	3.4	Open v Switch
	3.5	Implementing ECMP in SDN
Chapter 4	4.1	Goals of Project
Chapter 5	5.2	Traffic Design and Management

NIRMAL KARIA

Chapter Number	Section Number	Title
Chapter 1	1.1	History of data center topology
	1.2	Topology
	1.3	Routing
	1.4	Requirements of Data center
	1.5	Modern Data center Design
Chapter 4	2.5	Tools
	2.6	Mini-net Topology Creation
	2.7	Working of Fat Tree Architecture code
Chapter 5	5.1	Simulation and Evaluation
	5.3	Algorithm Evaluation
Chapter 6		Conclusion

REFERENCES

1. Open Networking Organization, https://www.opennetworking.org
2. An Introduction to Software Defined Networking, https://www.youtube.com/watch?v=H_3Lk6XbWw0&spfreload=10
3. Are We Ready for SDN? Implementation Challenges for Software-Defined Networks by Sakir Sezer, Sandra Scott-Hayward, and PushPinder Kaur Chouhan, CSIT, Queen's University Belfast Barbara Fraser and David Lake, Cisco Systems Jim Finnegan and Niel Viljoen, Netronome Marc Miller and Navneet Rao, Tabula http://ieeexplore.ieee.org.libaccess.sjlibrary.org/stamp/stamp.jsp?tp=&arnumber=6553676
4. Are We Ready for SDN? Implementation Challenges for Software-Defined Networks by Sakir Sezer, Sandra Scott-Hayward, and PushPinder Kaur Chouhan, CSIT, Queen's University Belfast Barbara Fraser and David Lake, Cisco Systems Jim Finnegan and Niel Viljoen, Netronome Marc Miller and Navneet Rao, Tabula http://ieeexplore.ieee.org.libaccess.sjlibrary.org/stamp/stamp.jsp?tp=&arnumber=6553676
5. Software Defined Network – Architectures: 2014 International Conference on Parallel, Distributed and Grid Computing; Nishtha Department of Computer ScienceHimachal Pradesh University, ShimlaShimla, India-171005, http://ieeexplore.ieee.org.libaccess.sjlibrary.org/stamp/stamp.jsp?tp=&arnumber=7030788
6. Cloud Computing Impact on Data Centers, http://www.techrepublic.com/blog/the-enterprise-cloud/how-cloud-computing-will-impact-the-on-premise-data-center/
7. Cloud Computing Wikipedia, https://en.wikipedia.org/wiki/Cloud_computing
8. Data Center Wikipedia, https://en.wikipedia.org/wiki/Data_center

9. 5 Reasons why a modern data center is required for digital world, http://www.gartner.com/newsroom/id/3029231
10. Do You need a Data Center, http://www.emersonnetworkpower.com/documentation/en-us/solutions/cio-topics/documents/executive%20briefs/executivebrief_do%20you%20need%20a%20new%20data%20center_fa.pdf
11. Cloud Computing and Evolution of Data Centers, http://www.techrepublic.com/blog/the-enterprise-cloud/cloud-computing-and-the-evolution-of-the-data-center/
12. Cloud Versus Data Centers, http://www.businessnewsdaily.com/4982-cloud-vs-data-center.html
13. Scale out networking in the data Center, http://www.computer.org/csdl/mags/mi/2010/04/mmi2010040029-abs.html
14. Camcube, Rethinking the Data Center Cluster, http://www.cs.vu.nl/~kielmann/hpdc-trends-2012/costa.pdf
15. Multi-tiered Network Architecture, https://www.digitalspyders.com/us/network-architecture.html
16. Symbiotic Routing in Future Data Centers, http://david.choffnes.com/classes/cs4700sp15/papers/sigcomm10-camcube.pdf
17. A Cost Comparison of Data Center Network Architectures, http://homes.cs.washington.edu/~arvind/papers/datacenter_comparison.pdf
18. A Comparative Analysis of Data Center Network Architectures, http://www.seas.gwu.edu/~guruv/iccngn14.pdf
19. Flattened Butterfly: A Cost Efficient Topology for High-Radix Networks, http://www.cs.berkeley.edu/~kubitron/cs258/handouts/papers/ISCA_FBFLY.pdf
20. Camcube: A Novel data Center, http://trilogy-

project.org/fileadmin/publications/Presentations/2010_07_02_Multipath_transport/rowston-camcube.pdf

21. OpenFlow Table Patterns, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/OpenFlow%20Table%20Type%20Patterns%20v1.0.pdf
22. OpenFlow Switch Specification, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf
23. OpenFlow Wikipedia, https://openflow.stanford.edu/display/ONL/POX+Wiki
24. Chapter 4: SDN Controller 'Software Defined Networks An Authoritative Review of Network Programmability Technologies" by Thomas D. Nadeau, Ken Gray.
25. L3 Learning in POX Controller, https://haryachyy.wordpress.com/2014/06/03/learning-pox-openflow-controller-imitating-l3/
26. A Cost Comparison of Data Center Network Architectures, http://homes.cs.washington.edu/~arvind/papers/datacenter_comparison.pdf
27. Cisco-Fundamentals of Spanning Tree Protocol, http://www.ctwebfactory.com/ccna/html/b3.html
28. Fat-Tree Architecture Wikipedia, https://en.wikipedia.org/wiki/Fat_tree
29. Mininet Simulation, http://www.brianlinkletter.com/mininet-test-drive/
30. Spanning Tree Protocol Wikipedia, https://en.wikipedia.org/wiki/Spanning_Tree_Protocol
31. Software Defined Networks (EWSDN), 2013 Second European Workshop on, Issue Date: 10-11 Oct. 2013, Written by: Teixeira, J.; Antichi, G.; Adami, D.; Del Chiaro, A.; Giordano, S.; Santos, A.
32. M. Al-Fares, A. Loukissas, and A. Vahdat, ``A scalable, commodity data center network

architecture," in Proc. ACM SIGCOMM, 2008, pp. 63_74.

33. Hailong Zhang School of Information Engineering, Communication University of China, Beijing, China Xiao Guo ; Jinyao Yan ; Bo Liu ; Qianjun Shuai - SDN Based ECMP Algorithm for data center networks.
34. Yi-Chi Lei ,Kuochen Wang and Yi- Huai - Hsu ,Department of Computer Science, National Chiao Tung University, "Multipath Routing in SDN based Datacenter ", 2015
35. Software Defined Networks (EWSDN), 2013 Second European Workshop on, Issue Date: 10-11 Oct. 2013, Written by: Teixeira, J.Antichi, G.Adami, D.Del Chiaro, A.Giordano, S.; Santos,A.
36. M. Al-Fares, A. Loukissas, and A. Vahdat, ``A scalable, commodity data center network architecture," in Proc. ACM SIGCOMM, 2008, pp. 63_74.
37. Hailong Zhang School of Information Engineering, Communication University of China, Beijing, China Xiao Guo ; Jinyao Yan ; Bo Liu ; Qianjun Shuai - SDN Based ECMPAlgorithm fordatacenternetworks.