Optimization & Evaluation of Data Center Network

by
Syed Ahmed Raza
saraza.z@gmail.com

Abstract

In recent years, technology is changing very rapidly and IT and Telecom infrastructure is expanding day by day. Especially With the emergence of cloud computing many business enterprises migrate their computing needs onto the cloud which results in enormous increase in the quantity of servers installed and the number of virtual machines running on them hence results in an increase in inter-server data traffic and an increase in bandwidth requirement for data center communication Network (DCCN). On the other hand High bandwidth, low latency and high availability requirements for real-time applications is enforcing Data networks to adopt the new concepts like Software Defined Networking (SDN), different network virtualization techniques and to optimize the network architecture (by modifying network topology & managing traffic flow through different techniques) which can handle & fulfill the future requirements.

Currently in traditional data centers, three tiers architecture is deployed which is based on Traditional Ethernet/fabric switches. The main drawbacks of this architecture from future point of view are network scalability, flexibility, latency, cabling, current management, power consumption and its control issues which are almost static up to certain extent. This research paper presents a flat novel data center Communication network (DCCN) architecture named as HFPFMAOS (Hybrid Flow based packet filtering, Forwarding and MEMS based all optical switching). This solution optimize the data center Communication network architecture in such a way that most of its issues or challenges is resolved up to certain extent in less CAPEX (capital expenditure) with little modification keeping most of its current network infrastructure. In HFPFMAOS OpenFlow (OF) enabled switches is deployed as TOR switches at Access layer to perform Flow based packet filtering and Forwarding and at aggregation layer along with traditional Switches, MEMS based all Optical Switching (MAOS) is deployed to provides highly scalable and dynamic data paths for switching of large Data traffic (Elephant Traffic) to control latency and congestion. While for the management and control of OpenFlow (OF) switches & MAOS Plane SDN controller is deployed. Deployment of SDN Controller and OpenFlow (OF) Switches decouples data forwarding plane from the control plane and provides us centralized intelligence of whole network that is helpful in better traffic

management, minimizing overheads, fastening updates and quick troubleshooting that reduces fault handling time in data center Communication network (DCCN).

Keywords

Data center communication Network (DCCN), HFPF, AOS (Hybrid Flow based packet filtering, Forwarding and MEMS based all optical switching), Software defined Networking (SDN), MEMS (Micro Electromechanical Switching), OpenFlow (OF).

Introduction

1.1 Background

A data center is a specialized facility which is used to premises different resources like servers, data storage systems and Telecommunication & networking systems to handle its IT needs. From the past few years with the rapid growth in information technology and internet usage, data center is playing a vital role in IT & data networks. Moreover with the advent of cloud computing, server virtualization, social media & mobile data, data center network is expanding exponentially in terms of no of server's deployment, storage and

intercommunicating network infrastructure. The main components of Data center IT network are (i) DCCN (Data center communication network) to handle different types of communication traffic (ii) HPCN (High Performance computing network) that includes servers which can be rack mount or chassis based (iii) SAN (Storage Area Network). Being backbone of data center, DCCN should have higher throughput, fault tolerance, energy efficiency and scalability, to accommodate increasing bandwidth needs. Traffic flowing through DCCN can broadly classified into three types;

- Traffic flows that stays within the data center.
- Traffic flows that is between data centers (i.e. from one data center to another data center).
- Traffic flows that is b/w data center and the end user via IP WAN or Internet.

According to cisco "Global cloud index forecast" [1] the total amount of traffic crossing the internet in 2016 will reach to 1.3ZB (Zeta Byte)/year and by 2019 it will reach to 2.0 ZB[2] while data center IP traffic grows approx. three times and reaches to approx. a total of 5.6ZB/year in 2016 and 10.4 ZB/year by 2019 which corresponds to a CAGR(compound annual growth rate) of 25% from 2014 to 2019.as shown in fig.1.1[2] The main component of this growth is cloud computing which is expected to be four-fifth of data center traffic by 2019[2].

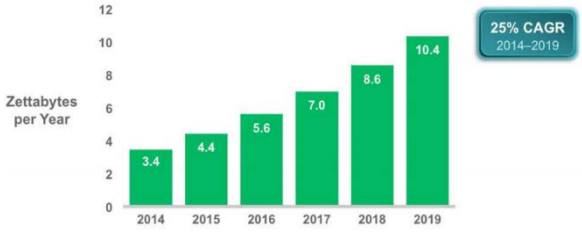


Fig 1.1: Global Data Center Traffic, 2014–2019 [2]

By this forecast 17.8% traffic flows till end user (in terms of web browsing, video streaming mobile data, VOD, email etc.) and only 6.8% traffic is from one data center to another data center (in terms of contents replication, exchanging database, data moving between clouds etc.) while 75.4% of the total traffic resides inside the data center b/w clusters also known as East-West traffic (in terms of storage, VM motion, database backup, authentication and development data) that which is presented in Fig. 1.2 as;

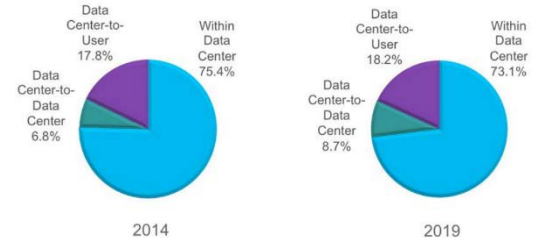


Fig 1.2: Global data centers traffic by destination [2]

From the above figure it is clear that majority of traffic exists within data center and out of which most of the traffic traverse between Edge Switches and servers. Traffic forecast between different layers of data center communication network is as shown in fig 1.3.

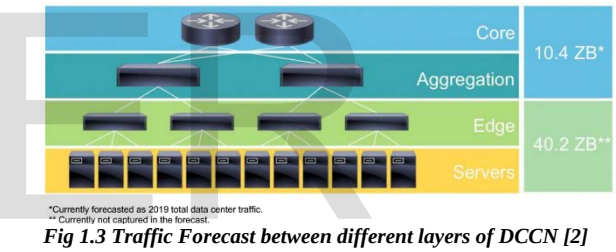


Fig 1.3 Traffic Forecast between different layers of DCCN [2]

1.2 Problem Statement

Globally in modern data center communication networks (DCCN), big data (Particle Accelerator, Airplane, Train, Hospitals, Oil Wells etc.)[2] along with cloud applications, server virtualization, video & mobile data traffic is growing very rapidly day by day. This growth requires unified network architecture (topology) and networking equipment which can handle & fulfill the performance requirements of multiple types of traffic with optimum mix of low cost, low latency, energy-efficiency& high performance. The issues or challenges which the DCCN will have to face are as;

- Limited Capacity between servers& high Oversubscription ratio as traffic move up in hierarchy through the layers of switches

- Congestion and high network latency issues badly effect delay sensitive applications and degrades overall network performance.
- Management of internal data-center traffic (i.e. east-west traffic between racks) which is almost 75% of the total traffic.
- Support to a variety of traffic patterns, which includes short persistent "mouse" flows and long persistent, high bandwidth "elephant" flows.
- Scalability, Agility and Efficient resource utilization issues as network grow.
- *Management, fault handling and troubleshooting in the network.*

1.3 Research Contribution

This research work propose a solution named HFPFMAOS (Hybrid Flow based packet filtering, Forwarding and MEMS based all optical switching) which optimizes DCCN architecture for catering its challenges up to certain extent with low capital expenditure. This solution will not only helpful in improving scalability issues, traffic management, Congestion control & utilization of available resources but also helpful in reducing network latency, power consumption and saving CAPEX.

HFPFMOAS employs OpenFlow(OF) at access switches which is performing Flow based packet filtering, Forwarding and MEMS based all optical Switching at aggregation layer provides switching of large persistent, high bandwidth data flow (elephant flow) along with traditional packet switching which is used for mouse flows. This will enable us to avoid normal data traffic route for specific flows and create dynamically high band width data paths directly between any OF (TOR) switches through MEMS switching plane for switching of that specific flows of elephant traffic, which results in reduced network latency due to all optical (OOO) switching and less chances of bandwidth bottlenecks. For centralized control, management and intelligence it is connected to SDN controller that keeps monitoring whole network all the time and maintains a centralized Topology database which helps us to dynamically create High bandwidth data path whenever and wherever it is required between OF Switches. Moreover it will reduce management overhead bytes on the network and helps us in quick fault diagnosis and its rectification.

For management, centralized control and performing different tasks we can install wireless network card in servers that provides a direct channel for communication between Server and SDN controller. This will be helpful in congestion notification ,link utilization updates, increased latency and scheduling different tasks like database backup, VM migrations from one server to other server, and creation of high bandwidth data paths through MEMS plane dynamically.

1.4 Research Methodology

Install Mininet in one virtual machine and Open DayLight controller in another virtual machine using Oracle VM VirtualBox manager. Build a reference data center network topology by deploying traditional switches in Mininet using MiniEdit, run the topology and generate multiple data traffic streams between different hosts using IPERF and check the network delay and its performance by observing their latency and packet loss before and after congestion. In second step deploy the OpenFlow (OF) switches in Access layer as TOR switches replacing traditional switches and connect them with remote SDN Open DayLight controller running in other virtual machine, capture the OF messages through Wire shark to verify the proper connectivity and functioning of OpenFlow protocol. Then generate the traffic with in the network between different Hosts using IPERF and check the network performance and delay. In last step MEMS switch is deployed at aggregation level along with traditional switches and connect this with OpenFlow (OF) Switches deployed at Access layer & ODL controller. Then again generated the traffic between different hosts and observe the latency and packet loss of multiple flows before eighty percent utilization of uplinks, after that add flow entries in the OF switches and directed the next flow through high bandwidth MEMS switch. Observe the network performance (i.e. latency, percentage of packet loss, data transfer rate etc.) and compare the results.

1.5 Software Used

Oracle VM VirtualBox manager, Mininet, Miniedit, ODL controller, Xming, Putty, Wireshark.

Data Center

2.1 Introduction

Typically a data center is a special facility conceived to house, manage, and support computing resources that are considered critical for one or more organizations.[3] A data center is a complex structure comprising of special building infrastructures, power backup arrangements, cooling systems, equipment cabinets, servers, main frames & High performance Computing(HPC) network to handle different types of traffic of communication, storage Area network (SAN), LAN & communication infrastructure with multiple networks (DCCN), structured cabling, application software, monitoring center and physical security systems. The illustration of a basic Data center is shown in fig 2.1 [3]

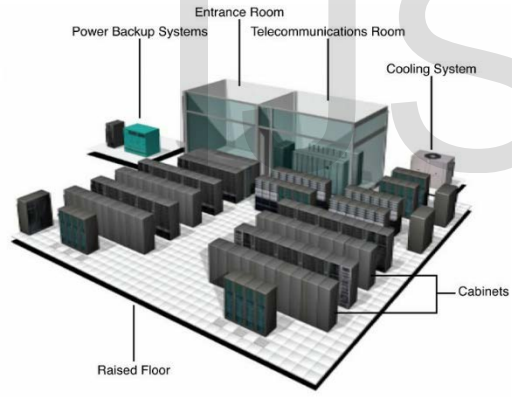


Fig. 2.1 Basic Data Center Diagram

Above diagram just show the small basic Data Center deployed in single room where as a real Modular Data center can spread over vast area and occupies several rooms spread over several stories or building, accommodating different infrastructures depending upon its criticality and its percentage of availability (i.e. Data center Tier).

2.2 Evolution of Data Center

The initial phase (Phase 1.0) of data center starts back in 1950's consisting of computer rooms containing main frame systems CPU and peripheral devices like storage, terminal and printers etc. These are centralized systems based on monolithic software architecture allowing a limited control over IT infrastructure and high resource utilization. In 1980's data center phase 2.0 starts with the popularity of client-server application model. In this time trends changes and servers takes the place of main frame computers which are comparatively small and can be accessed through applications installed on client PC's. In this phase Server which performs specific function is usually deployed near the client instead of Main IT infrastructure as to escape from high bandwidth cost. In 1990's when internet took the boom and web based applications utilization increases, this enforces that servers should be deployed centrally in a properly designed data centers. So in the era of 1990's the data center that originates from main frame computer room has gain importance. Data Center Evolution timeline is shown in fig 2.2 [3] as;

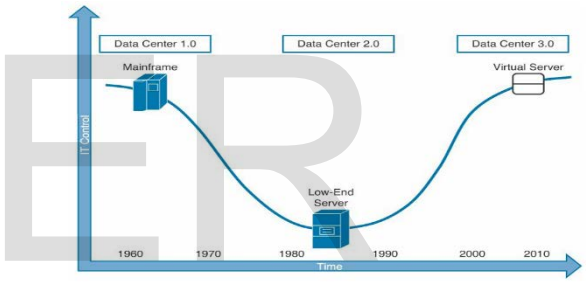


Fig. 2.2 Data Center Evolution Phases

Data center phase 3.0 starts around 2000, and as in this era technology changed very rapidly resulting in datacenter space, power and IT network start approaching to saturation so for expansion or to deploy new facilities require expansive budget. During such period Cisco IT study 2005 pointed out DC networks and servers were not fully utilized and on average only used at 20% of their capacity. The root cause of this situation were application silos with Discrete set of servers, network & storage resources. Later on various projects on network consolidation is done that offer new features, improves resource utilization, reduces no of network elements, processes and increase operational efficiency. Later on various virtualization technologies were introduced that provides many benefits like (i) consolidated structure in an isolated environment (ii) Aggregation of multiple resources into one shared resources (iii) simple operation procedures etc.

Data center network infrastructures are extremely interdependent of each other like (i) No of server/Rack depends on the Power distribution design (ii) Transport network depends on the No of installed servers in each rack .

2.3 Types of Data Centers

There are two main types of data centers [4][5]

- Private Data centers.
- Cloud Data centers.

2.3.1 Private Data Centers

Private data center usually refers to on premises hardware that provides computing services and stores data within an organization local network administrated by its IT department. Such kind of data centers is usually owned and operated by small private/public companies, government agencies and large companies providing technology products and services like Intel, Google, Microsoft, NASA etc.

2.3.2 Cloud Data Centers

Cloud data centers are also called Co-location/Managed Services provider. These are the data centers that is established and maintained to provide infrastructure and services to the third parties. It usually refers to off-premises computing that stores data on the internet and all its services is outsourced to third party cloud provider, who often have multiple data centers in several geographical locations for data protection, redundancy and high availability and is responsible for all its up gradation and maintenance. Main factors that forces companies to use cloud computing or built its own data centers are;

- Business needs.
- Data security.
- Equipment or infrastructure cost.

Building own data center is suitable for companies having enormous work load and running multiple types of application or offering multiple different types of services, as it has limited capacity and workload and have limited scalability. Whereas cloud computing is cost effective and offers more scalability but the subscriber has very less control and data security & also infrastructure is shared as well among multiple users.

2.4 Attributes of Data Center Communication Network (DCCN)

The main objective of DCCN is to provide data services and to transport the server's data to the clients or to others servers. Now a day from reliability, growth and efficiency point of view Data center communication network must possess these attributes

- **Availability:** network failure ability to quickly recover from them and mask their effects to the end user and connected devices.
- **Scalability:** Ease in Expansion as the network grows.
- **Flexibility:** Support design and deployment changes without adverse outcomes.
- **Efficiency:** Capable to fully utilize its available resources.
- **Predictability:** shows expected behavior during a fault and after its recovery.

Due to its layered architecture, high speed, low cost, simplicity, addressing flexibility, and extensibility, for data links Ethernet is the most common and popular protocol of DCCN and its interface range varies from 10Mbps to 100Gbps. However different limitations spurred to develop and adopt new virtualization technologies and to optimize data center network by deploying new networking techniques like MEMS all optical switching along with traditional packet switching as the traffic through DCCN grows.

2.5 Data Center Tiers

Tier is a standardized method which is used to define the percentage of availability or the uptime of a data center. Tier of a data center ranges from 1 to 4 and is useful for measuring.

a) Performance of a Data center b) Capital Expenditure (CAPEX) c) Return on Investment (ROI) Tier 1 is the simplest data Centre used by small business or shops. Tier 4 data center is very less prone to failure and is

considered as most robust. It is designed in a way that it's all subsystems have full redundancy (i.e. server & network links, storage, power, cooling etc.) and every section have different security zones which is controlled by different methods like biometric access control. Data center Tiers from 1 to 4 are as [6] [7];

- **Tier 1** = Guarantee 99.671% availability & components like (uplink and servers) have no redundancy.
- **Tier 2** = Guarantee 99.749% availability & components like (uplink and servers) have redundancy i.e. Tier 1 + redundancy.
- **Tier 3** = Guarantee 99.982% availability & have dual powered equipment's, multiple uplinks.
- **Tier 4** = Guarantee 99.995% availability & have dual powered equipment's. Also all components like uplinks, storage, servers, HVAC systems, and chillers etc. are fully fault-tolerant.

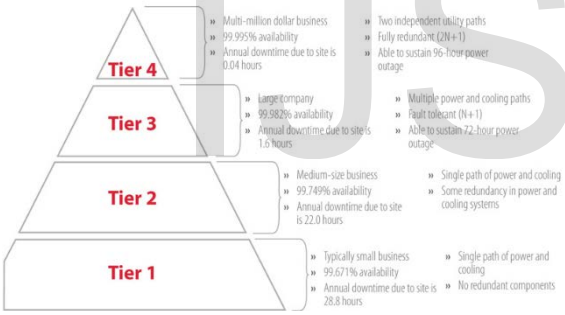


Fig. 2.3 Data Center Tiers

2.6 Design Factors for Data Center Network

While designing & deploying DCCN architecture we must have to consider following parameters [3].

- **Failure Impact & Application Resilience:** A failure in DCCN will affect all local and remote users to access their applications. In order to avoid traffic interruption (black holes) server-redundant interfaces should be connected to different access switches and network must have to capability to recover from any connection failure.
- **Server & Host connectivity:** Servers must be connected via redundant Ethernet connections.
- **Traffic Direction:** In DCCN most of the traffic lies within the data center i.e. b/w servers.
- **Agility:** It defines the ability that any service or application can be assigned to any server at any time as per requirement with in the data center network keeping proper performance Isolation and security between different applications.
- **Growth Rate:** It shows the increase in No of servers, switches and switch ports etc. with increase in customers or their data traffic, in order to avoid any congestion or bottleneck in the network topology.
- **Application Bandwidth Demand & Oversubscription Ratio:** These two aspects are very important and interrelate with each other while designing DCCN. Application bandwidth demand tells us that every application should be placed in such a way that proper resources can be allocated to it according to its bandwidth needs and oversubscription tells us No of outgoing ports or switch modules allocated for the specific number of hosts or Servers traffic etc. Oversubscription in an environment where multiple elements sharing a common resource and is defined as the ratio of assigned resources to each consumer to the maximum utilization by each consumer. In DCN it refers to amount of bandwidth that switches can efficiently offer to downstream devices at each layer e.g. if an access layer switch have 32 10Gig server connecting ports and 8 10Gig uplink ports then it has oversubscription ratio of 4:1 for servers upstream traffic. So through detailed analysis, fine-tuning, and testing we can set an oversubscription ratio that can fulfill the applications current and future demands.

2.7 Challenges for DCCN

Currently issues or challenges which the data centers communication network will have to face are as;

- Limited capacity between server's & high

Oversubscription Ratio as traffic move up in hierarchy through the layers of switches i.e. from TOR to Aggregation switch and from there to Core oversubscription ratio increases rapidly. Moreover, for communication between the server's located in different layer 2 domains, data must have to pass through layers 3 domain i.e. through Aggregation routers/switches and Border router/switches as a result link capacity between access and Border routers/switches is less than the total link capacity of servers connected to the access routers having oversubscription ratio of 10:1 to 80:1 [8]. Also these routers links is also carrying traffic traversing in and out of the data center so as a result bandwidth available over these links for server's communication b/w different parts of data center is limited.

- Congestion and high network latency. As in data center different applications are running generating different packets flows moving within and across the DCCN which is a multi-tier hierarchal structure consisting of multiple hops. In some cases bandwidth bottleneck is created at aggregation level where packets Incoming rate from multiple transmitters exceeds the packet handling rate of the receiver as receiver is running out of buffer space available to absorb packet flows. As a result of this congestion delay time increases and receiver starts dropping the packets which effects different application especially latency sensitive applications hence decreasing overall performance of the network.
- Management of internal data-center traffic (i.e. east-west traffic between racks) which is almost 75% of the total traffic and is due to separation of application servers, databases and storage which results in read/write, back-up and replication traffic traversing with in the data center. Moreover, parallel processing distributes the tasks and then assign these to multiple servers hence increases the internal traffic in DCCN.
- Support to a variety of traffic patterns, which includes short persistent "mouse" flows and long persistent, high bandwidth "elephant" flows. Mouse flows are usually related with bursty and latency-sensitive applications where as elephant flows usually incudes high bandwidth traffic due to VM (virtual-machine) migrations, data migrations and database backup, and Big data applications like Map-Reduce, Hadoop etc. Big data examples include Data generated by Particle Accelerator, Airplanes, Trains, Self-driving Cars, patients data in Health Care hospitals etc. e.g. In 2014 Onboard data produced by Aviation Boeing 787 is 40TB/hour and out of which 0.5TB/hour of data is Transmitted to data center. Usually most of the flows within a data center are mouse flows, but most of the data belong to a few elephant flows. Hence handling all type of traffic simultaneously keeping overall network delay within limit is a real challenge.
- Scalability, Agility and Efficient resource utilization issues as network grow. As no of Communication devices increases their management, fault handling and their troubleshooting also becomes difficult, not only this but it will also increase management overhead bytes on the network.
- Increase in Energy requirements and its consumption with the addition of more devices.

Currently in traditional data centers access layer comprises of server racks and each rack consist of no of servers which can be Rack mount server or Chassis based & TOR (Top of Rack) switch also known as Fabric interconnect to which all the servers of that rack is connected. This TOR switch of each rack is connected to EOR (End of Row) switch which is considered as the packet based aggregation layer of each cluster where cluster is a group of server racks. This aggregation layer on one side provides the connectivity between clusters of the data center over which East-West packet based Data traffic traverse and on other side it also provides connectivity with the core network as well over which North-South bound traffic traverse. Hence aggregation layer is the layer where problem comes as 75% traffic (East-West traffic) of the Data center traverse through it, resulting in the bandwidth bottleneck & increase in latency. Consequently, this will badly effect delay sensitive

applications and degrades overall network performance.

2.8 DCCN Topologies

DCCN is usually a layered architecture consisting of three layers (core, distribution and access) built to hosts servers provide different types of services to their clients and connectivity to their servers. A Data center communication network (DCCN) infrastructure is the backbone for its applications and should have these characteristics like reliability, scalability, low latency and enough bandwidth in order to avoid any congestion or bottleneck in the network. These characteristics mainly depend upon the network architecture in which the DCCN is laid and play a vital role in overall performance of Data center. According to cisco Common DCCN Architecture is shown in fig2.4 [9].

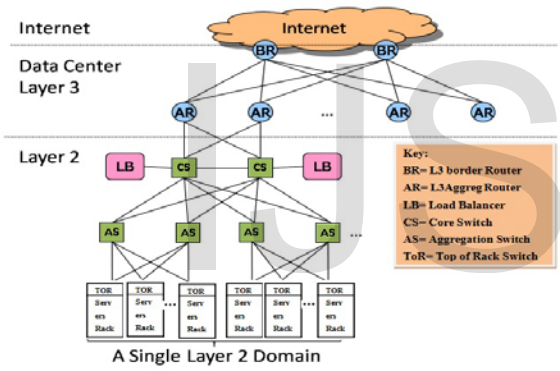


Fig 2.4 Common DCCN Architecture

The benefit of layered architecture is that it improves network resiliency, modularity and flexibility. In this architecture every layer performs different functionalities for distinct profiles like core layer includes routers (i.e. Aggregation router & Border Router) which will do forwarding decisions for both Ingress/Egress traffic using routing protocols. Aggregation router is used for routing between various layer 2 domains while core router is used for carrying traffic between aggregation routers and routing over the internet or WAN. Layer 2 domain consists of Core switches that provides highly flexible, scalable connections with multiple aggregation layer switches. This layer provides a central point for server IP subnets and act as a default gateway. It usually used to forward inter server traffic b/w multiple aggregation layer switches and state full network devices like server load balancers and firewalls are attached on this layer. Access layer usually consist of switches to which servers having a same IP subnet are attached and communicate with each other. It helps to improve network management and responsible to exchange any type of traffic like unicast, multicast or broadcast between servers.

In above figure at the bottom there is server layer mounted in server racks. These servers are running many Virtual machines which are assigned to different applications running inside the data center. An application is usually associated with multiple public IP's to handle and respond external client's requests coming from internet. For processing, these requests is distributed among the pool of servers by specialized hardware known as load balancer connected to the core switches in redundant fashion. The IP address which handle these requests is called VIP (virtual IP address) configured on the load balancers and the server IP addresses over which requests are forwarded are known as DIP's (Direct IP addresses)[10]. For each VIP there is list of DIP's configured on the load balancers over which incoming requests are forwarded for processing.
Typically there are 20 to 40 rack mount servers installed in one rack and these servers are connected to Top of the Rack (TOR) switch installed in the same rack over 1gbps link which can be electrical or optical. These TOR switches are connected with the two aggregation switches in redundant fashion. This aggregation layer is further connected to the Core layer switches. Below Layer 3 domain there is a layer 2 domain consisting of core Switches Aggregation switches and TOR switches connecting thousands of servers.

This three layered hierarchical design is most widely deployed in the world and suitable for small and medium sized data centers but seeing the current growth trends there is a need of new architecture which should be designed in order to meet the future challenges like scalability, cross-sectional bandwidth, power consumption and cost of networking equipment. By seeing these challenges researchers present new architectures for DCCN out of which major is described as;

2.8.1 Fat-Tree

This DCCN architecture is proposed by Al-Fareset al [11]. It uses the concept of fat-tree and aiming to

increase the scalability, end-to-end cross sectional bandwidth, backward compatibility and decrease cost and fault tolerance. This topology divides the overall infrastructure into different pods and the core. The pods consist of Servers, TORS (Access switches) and aggregation switches. These aggregation switches connects the Pods to the core switches and each port of the core switch is connected to one of the aggregation switch of different pod. It supposes that all of the switches are identical having K number of ports, hence have K number of pods each having K/2 number of Access and aggregation switches. Each aggregate switch in a pod is connected to all Access switches to which K/2 servers are connected. Hence if K=4 then there are 4 pods and 4 no of servers in each pod each having 4/2=2 edge and aggregation switches as shown in fig;

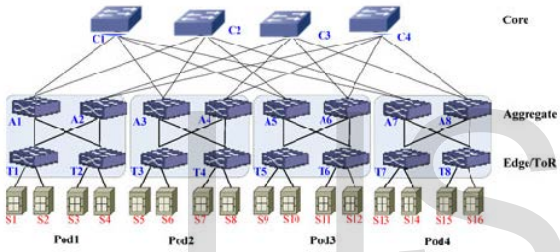


Fig 2.5 Fat-Tree topology architecture

This topology provides customized IP address scheme and multipath routing algorithm by customizing the data links in the DCCN architecture. The comparison of 3 tier and fat-tree is as shown in fig 2.6.

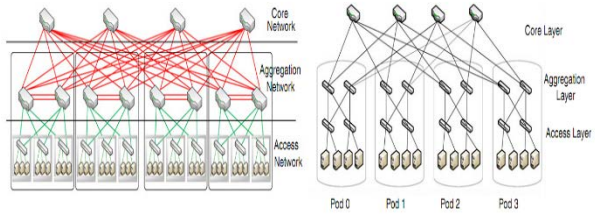


Fig 2.6 Comparision of 3 Tier and Fat-Tree topologies architecture

2.8.2 BCube

This topology [12] is proposed for modular data centers built inside the container offering simple installation and easy migration but scalability is low as designed for container data center. This architecture consists of layers of COTS (Commodity off the shelf) switches and servers which are responsible for packet forwarding. At level0 BCube architecture consists of multiple modules of BCube0 each having one switch with n number of ports connected to n no of servers. At level 1 there is a BCube1 module which consists of n switches to which n BCube0 modules or networks are connected. Each switch of BCube1 is connected to one server of BCube0 module.

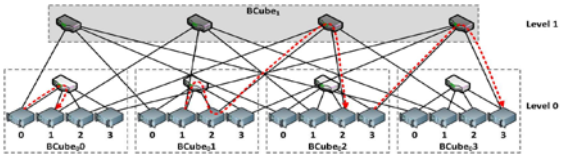


Fig 2.7 BCube Topology Architecture

In above fig server 0 of BCube00 is directly communicating to server 1 using its local switch, whereas server 1 of BCube01 forward its packet to server 1 through local switch which then forward these packets to its destination which is server 2 of BCube02 through upper level BCube1 switch. Also the communication between same level BCubes can take place through upper level BCube1 switch as server 3 of BCube02 is communicating with server 3 of BCube03 as shown in fig 2.7.

2.8.3 DCell

DCell is a recursively defined architecture which is proposed by Guo et al . This architecture is highly saleable and uses mini switches and servers for forwarding packets. Like BCube its main module is DCell0 consisting of single switch that is connected to n servers. Dcell1 consist of n+1 number of DCell0modules and each DCell0module is connected by one link each to the other DCell0modules through its servers as shown in fig 2.9.

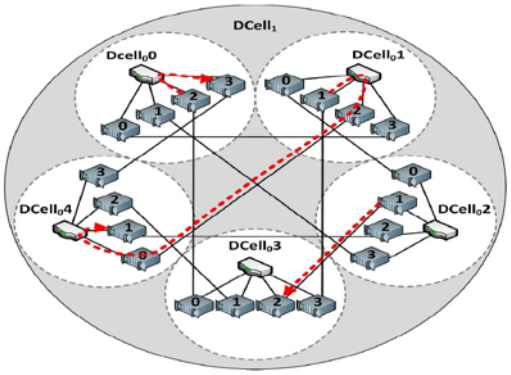
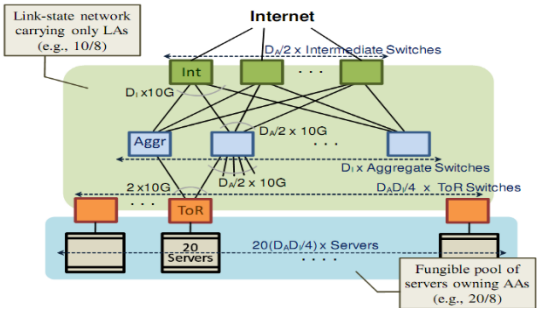


Fig 2.8 DCell Topology Architecture

Fig 2.9 VL2 Network Architecture built with LAs and AAs

2.8.4VL2

Virtual layer 2 (VL2) is a Fat-Tree based DCCN architecture aimed to provide flat automated IP addressing scheme facilitating placement of servers anywhere in the network without manual IP address configuration. This allows the provision that any service can be assigned to any server in the network. This architecture provides scalability supporting large scale data centers, high bandwidth between servers, Agility by dynamic allocation of servers to applications from shared pool resulting in efficient resource utilization and also provides ability to transparently migrate services or virtual machines to any server keeping same IP address & network configuration just like connected via LAN. It uses [14] address resolution based on end system to scale to large server pool, flat addressing scheme to place service instances anywhere in network just like a LAN (Local Area Network) where any Host with any IP address to any port on a switch can be connected & valiant load balancing to uniformly distribute traffic across the DCCN. To support agility VL2 uses a scheme of addressing which separates names of the servers named as AAs(Application specific address) from their location named as LAs(Location Specific Address) as shown in fig 2.10 [14]. IN VL2 switches and interfaces are assigned LAs and network operate using LAs(Location specific IP addresses), whereas applications uses AAs(Application specific IP addresses) which remains unchanged during VM migration independent of their location of migration. The Network infrastructure that include VL2 employs directory system for managing name to locator address mappings which is realized on servers for flexibility, dynamic re provisioning of service and context-aware server access control. A shim layer named as VL2 agent runs on every server that invokes directory systems resolution service.



Besides all these architecture there is also a need of architecture that not only provides backward compatibility with existing infrastructure and its architecture but can also handle current challenges which the DCCN are facing. In this thesis I proposed a new data center architecture HFPFMAOS which is not only backward compatible with existing architecture but also can accommodate its challenges up to some extent.

2.8.5 HFPFMAOS

HFPFMAOS is a data center communication Network architecture (DCCN) proposed in this thesis which stands for "Hybrid Flow based packet filtering, forwarding & MEMS based all optical switching". This architecture is proposed to cater different challenges which today's data centers are facing while keeping existing infrastructure. In this architecture I employed OpenFlow(OF) enabled switches at Access layer i.e. as TOR switches and MEMS based all optical switching (MAOS) plane consisting of MEMS switches (MS) at the aggregation layer along with traditional switches. This MAOS plane is connected with OF enabled TOR switches for dynamically provisioning High bandwidth data paths between the servers. For control and management, these OF switches and MAOS plane are connected with SDN controller over SSL/TCP. This will provide us centralized Intelligence of whole network by keeping an eye on all the flows passing through the OF enabled switches and helps us to dynamically create High bandwidth data path whenever and wherever it is required between the servers in the data center. Moreover it will reduce management overhead bytes on the network and helps us in quick fault diagnosis and its rectification. The architecture of this proposed solution is as;

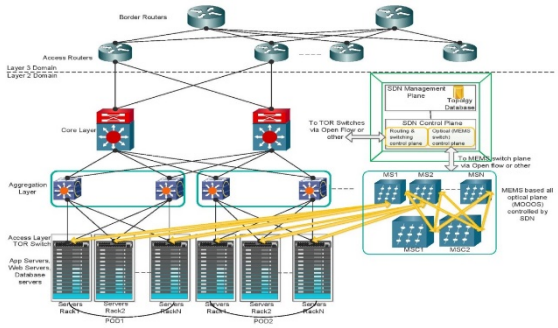


Fig 2.10 HFPFMAOS Architecture for DCCN

MAOS (MEMS all Optical Switching)

3.1 Introduction

Data center intercommunication Network is demanding a dramatic increase in its capacity as traffic is going to increase day by day exponentially. So all optical switching is best solution to get relief from electronic switched network bottlenecks. As optical fiber can offer very high bandwidth but it is limited by transmission capacity and electronic switching. Hence all optical switching can perform a vital role as switching is performed in optical domain.

3.2 OOO Switching (All Optical)

All Optical switching is an all optical networking technology in which the packet that enters the network travels in all optical form ingress point to egress point. In all Optical switching a circuit is established from ingress node to egress node by adjusting optical cross connects and data travel through this circuit in all optical form. It is an important breakthrough which can help us to solve the electronic switched network bottleneck as in it traffic is switched directly in optical domain instead of several optical-electrical-optical conversions.

3.3 Benefits of OOO Switching

The benefits of employing all optical switching in data center are as;

- Data centers usually employ POD based architecture resulting in poor computing resource utilization but using optical switching these compute resources can be shared between different PODs for maximum efficiency.
- Enhances revenue by provisioning new services quickly.
- Less power dissipation than conventional electrical switch.
- Create and reallocate capacity on demand.
- Incredible improvement in the operational efficiency of data center as applications runs smoothly by efficiently utilization of compute resources and by using 3D optical MEMS switching technology.
- Helpful in enhancing customer satisfaction by improving network efficiency, reliability.
- Improve quality of service & accelerate fault detection.
- Reduces CAPEX and operational expenses (OPEX) by provisioning easy up gradation to any higher optical data rates, improving network efficiency, reducing power consumption and by efficiently utilizing server and storage assets.

3.4 Applications of OOO Switching

Optical switching can be employed in wide range of applications which are as under;

- Optical switches can be employed in IP network for providing Optical circuit switching or optical packet switching.
- MUX & DEMUX: It is used in optical MUX (multiplexers) and DMUX (demultiplexers) to add and drop specific wavelengths from the composite signal consisting of multi wavelengths to avoid electronic operations. E.g. wavelength selectable switches (WSS).
- Fiber Restoration and Protection switching: It can be used for switching and restoring optical paths in case of any optical link failure having a switching time in ms (milli Sec). These are usually small switches.
- Signal Monitoring: It can be used for network monitoring and its management like it can be used for monitoring the optical power of a link for which WSS is usually used.

3.5OOO Switching Technologies

There are different switches that employ different all optical switching technologies which have different features and performance like Scalability, insertion loss, switching speed and crosstalk etc. A list of these is as [17].

- Micro Electromechanical system(MEMS) switching
- Liquid-crystal switching
- Bubble switching
- Electro-Holographic Switching
- Thermo-Optical switching
- Liquid-Crystals-in-Polymer Switching
- Acousto-Optic Switching
- Semiconductor-Optical-Amplifiers (SOA)Switching

Comparison of few of few OOO optical switching is given in this Table 3.1[17][18].

	MEMS	Liquid-crystal	Bubble	Electro-Holographic
Wavelength Range	1290 ~ 1625 nm	1525 ~ 1575 nm	1270 ~ 1650 nm	1310 ~ 1550 nm
Insertion Loss (IL)	1dB ~ 5dB	<1dB	<7.5dB	<3dB
Scalability	Higher scalability with 3D MEMS to thousands of ports but limited with 2D MEMS to about 64x64	low	Low (to about 100 ports)	Low
Switching Speed	1ms ~ 5ms	4ms	<10 ms	<30ns
Power Consumption & Technical Performance	Low power consumption less than 45W	Low power consumption about 50mW, Prefer for single mode instead of multimode, require polarization splitter, suffer from PMD, thermal fluctuation and moisture sensitivity	High power consumption about 25W, Wavelength dependent phase shift caused by bubble and hence amplitude dispersion in signal output, tradeoff between low loss and high isolation required.	Low power consumption, Polarization and wavelength dependent, hence dispersion and loss.
Cost & Packaging	Low cost & light weight	Costly & inconvenient	Low cost but some degree of precision required	Low cost & easily integrated with signal equalization & monitoring
Polarization Dependent Loss	<. 2dB	. 2dB	<. 3dB	
Maturity of Technology	Medium but Mature	Medium	Medium	Mature

Table 3.1 Comparison of different Optical switching technologies.

Hence from table it is obvious that MEMS technology have many advantages over other technologies like High scalability using 3D MEMS, low cross-talk& insertion-loss, independence of bit rate/wavelengths/modulation/polarization, switching speed in ms(some also claims in ns) and most important is low cost and low power consumption. Due to these reasons I choose 3D MEMS optical switching should be employed in Data center for Optical circuit switching.

3.6 Micro Electro Mechanical Systems (MEMS)

Micro electromechanical systems (MEMS) is a technology that combines computers with tiny mechanical devices such as sensors, valves, gears, mirrors, and actuators embedded in semiconductor chips[31]. It contains micro-circuitry on a tiny silicon chip into which some mechanical device such as a moveable micro-mirrors or a sensor has been installed that can reflect optical signal from input fibres to output fibres. By rotating or tilting the mirrors at different angles we can reflect the light beam from input port to any output port direction. Switching of light beam from one I/P port to other O/P port depends upon the tilt angle of mirror's in the I/P and O/P MEMS array. It consists of a pair of collimator arrays having I/O ports and a pair of MEMS mirror array arranged in a Z shaped layout.

3.7 Design and Principle

The basic structure of 3D MEMS optical switch consists of following component as shown in fig 3.1[16].

(i) **Input collimator:** An array of I/P fiber ports which guides the light from each I/P fiber to its I/P MEMS mirror array.

(ii) **Mirror matrix:** It is an array of MEMS I/P mirrors and an array of MEMS O/P mirrors.

(iii) **Output collimator:** An array of O/P fiber ports which couples the light from each O/P mirror to its O/P fiber.

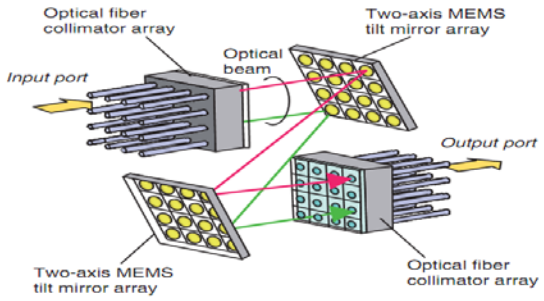


Fig. 3.1 Basic structure of a 3D MEMS optical switch

As in figure the MEMS optical switch consists of two optical fibre collimator arrays one for each I/P and O/P ports and two MEMS Tilt mirror arrays. Each mirror in the I/P and O/P MEMS array is dedicated for each port in the I/P and O/P collimator array. The optical beam coming from I/p port of the I/P collimator strikes to their dedicated mirror in the I/P MEMS mirror array from where it

is reflected towards one of the mirror of O/P MEMS mirror array depending upon its tilt angle. When the light beam reaches to one of the mirror of O/P MEMS mirror array it is again reflected towards one of the O/P port of the O/P collimator array. This optical path Connection from the I/P port to the O/P port of the collimator arrays depends upon the tilt angle of both the mirrors in the I/P and O/P MEMS mirror array. The angle at which the mirror needed to be tilt is denoted by θ. When Corner mirror of the MEMS input Array is at the angle of 0° then the input light will be deflected to the opposite corner mirror of the MEMS output Array as shown in fig 3.2.

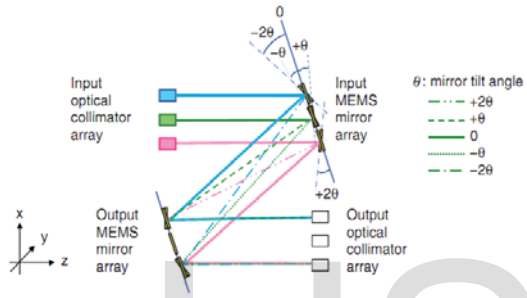


Fig 3.2 Conventional 3D MEMS optical switch.

Maximum tilt angle which is required to reflect the optical beam depends upon its position in the array and it increases as no of MEMS mirrors increases. To make the required maximum tilt angle constant for all mirrors in the array, a toroidal concave mirror can be used between the I/P MEMS mirror array and O/P MEMS mirror array to perform optical Fourier Transform. This will result in a W shaped layout providing a folded optical path as shown in figure 3.3.

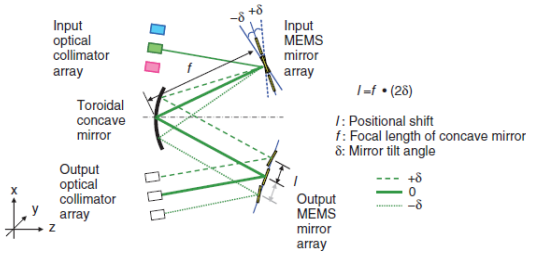


Fig 3.3 3D MEMS optical switch with Toroidal concave mirror.

Optical beam entering from the I/P collimator array is reflected by I/P MEMS mirror array and strike with the concave mirror at an incident angle which is determined by the MEMS mirror tilt angle δ. This Toroidal concave mirror performs a optical Fourier transform which converges the optical beam with a positional shift "/" towards output MEMS mirror array where the angle of each mirror is adjusted in such a way that the light beam is reflected into the desired output port of the collimator array. This shift "/" can be expressed in mathematical expression in terms of concave mirror focal length and the tilt angle of the MEMS mirror as;

$$/ = f \times 2\delta$$

It means that after using toroidal concave mirror the connecting output MEMS mirror is dependent on the tilt angle of the Input MEMS mirror instead of their position in the array. This W shaped layout providing a folded optical path results in a compact layout but it introduces off-axis aberration in X-Z plane which is reduced by using toroidal shape concave mirror. The calculated loss due to this aberration is less than 0.5dB.

The capacity of a switch or number of ports in the optical switch can be increased by producing large scale MEMS mirror arrays and other optical components but it increases fabrication complexities and makes that process slow. This problem is solved by arranging multiple small scale MEMS mirror and collimator arrays in a matrix form which makes fabrication process fast and less complicated e.g. 128 optical port collimator arrays and 128 port MEMS mirror array can be use in a 2X2 array form to obtain a capacity of 512 ports. This helps in producing high capacity optical switches and leads to ease in fabrication process with a small accumulated pitch error. The schematic diagram of 3D 512 MEMS optical ports switch with 128 ports in 2X2 array form is shown in fig 3.4.

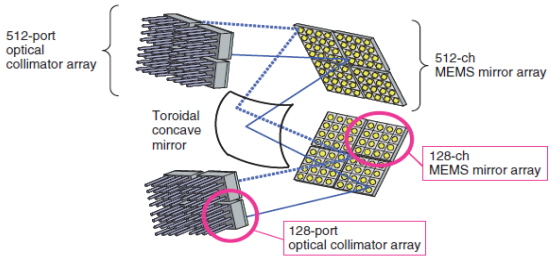


Fig 3.4 Schematic of 3D 512 MEMS optical ports switch

In above Fig 3.4 switch consists of four 128 ports MEMS mirror arrays arranged in a 2X2 matrix form. These arrays are mounted precisely in a ceramic package using multichip module technology. The 128 port MEMS mirrors are arranged in 2D in the array. Each mirror have four electrodes underneath it and has a gimbal structure which allow it to rotate electrostatically around two orthogonal axis resulting in a tilting angle range of 4.5° in any direction.

3.8 MEMS Mirror Structure

The mems mirror consist of two single crystal silicon substrates 1) Mirror substrate 2) Driving electrode substrate as shown in fig 3.5.

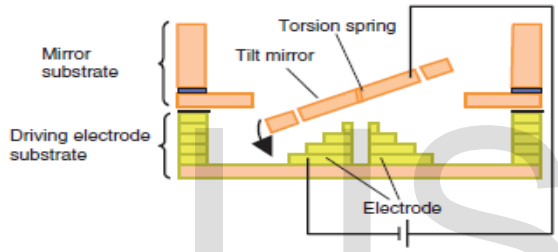


Fig 3.5 Cross Sectional Schematic of MEMS tilt mirror

These substrates are processed independently and is bonded each other by flip chip in such a way that air gap is formed between mirrors and electrodes. When we apply a voltage between electrodes and mirrors electrostatic force is generated which actuate the mirrors. Hence by applying a driving voltage to each electrode we can control the tilt angle of these mirrors.

3.8.1 Mirror Substrate

It consists of torsion springs and a mirror which are made up of single silicon crystal providing highly reliable movement to the mirror. The MEMS mirror which has a diameter of 600µm is connected on X axis to a gimbal ring and is supported by two folded torsion springs, while on the Y axis the gimbal ring is connected to the base by another pair of folded torsion springs as shown in Fig 3.6[16] These torsion springs have a high aspect ratio which is the ratio of spring thickness to its width. These springs have high rigorousness in Z direction with respect to torsion direction which stops the mirror from being pulled down and contact the electrodes. Hence optical beam can be reflected easily towards specific direction in 3D space by turning the mirror on the X axis and Y axis.

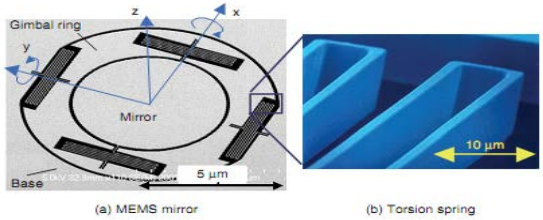


Fig 3.6 SEM snaps of MEMS mirror and High aspect torsion spring.

3.8.1.1 Fabrication Method of Mirror Electrode Substrate

The process flow for the fabrication of mirror electrode substrate include following steps
First of all by using lithography and dry etching mirror pattern is formed on SOI wafer (Silicon-on-Insulator) as shown in Fig 3.7 [16] step (a) then a coating of polyimide is spinned over the mirror pattern formed (b). In third step on the opposite side of the bulk Si the pattern for resistant mask(c) and mirror opening (d) is formed by dry etching. Next BOX (buried oxide) is removed with hydrofluoric acid which acts as an etching stopper (e), after that polyimide coating which protect the mirror from shocks during dicing is spinned again on the opposite side of the mirror coated surface (f). After dicing process it is exposed to oxygen plasma to ash off the polyimide layer (g). The mirror fabrication process by using this dry process is called "In-process sticking of the mirror". AS Au coating on the top surface of the mirror provides good reflectivity to optical beams so this coating is done on both side of the mirror. The optical characteristics of the switch consisting of MEMS mirror array depends upon the flatness of the mirror surface hence Peak-to-valley difference can be used to check the flatness of the mirror surface which is 0.05 µm in our case.

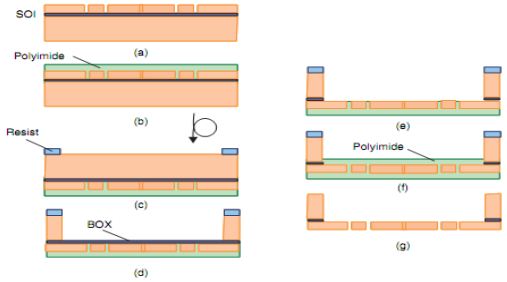


Fig 3.7 Fabrication flow process of mirror substrate

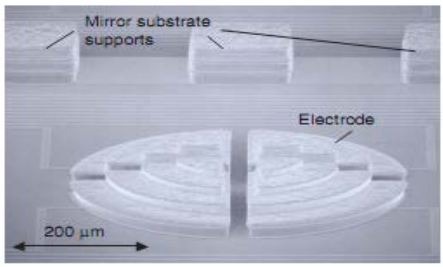


Fig 3.8 SEM snap of Electrodes

3.8.2 Driving Electrode Substrate

When we apply the voltage to the electrodes, electrostatic force is generated which is inversely proportional to the square of the gap between the electrode and the mirror. This electrostatic force causes the movement of mirrors.

3.8.2.1 Fabrication of Driving Electrode Substrate

In first step Si wafer is thermally oxidized and 0^{th} level interconnection is formed, then by evaporation metallic layer of AU/Ti is deposited over it. The Au layer act as a seed and Ti layer act as an adhesive for the successive electroplating as in fig 3.9step (a). In sec step polyimide coating is spinned over it as in (b) In third step by using lithography mirror substrate support and electrode are patterned over the first layer as in (c). While electroplating the thickness of the metal's varies with respect to the area to be plated because current densities on the wafer vary. So after electroplating to obtain similar electroplated area for each electrode, solid rectangular structures for mirror substrate support is formed as in fig 3.8. Now repeat the processes of polyimide coating, lithography and electroplating and we get thick multilevel electrodes embedded in polyimide as in (d, e, f). In last step polyimide is ashed away by exposing it to oxygen plasma. Hence after all these steps we get electrodes and mirror substrate supports whose height is over 80 µm ensuring enough space for tilting the mirror on the electrodes.

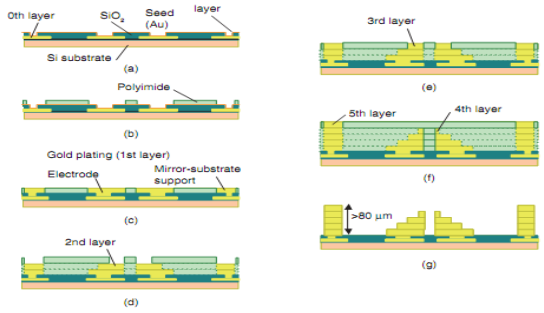


Fig 3.9 Fabrication flow process of Driving Electrode Substrate

3.9 MEMS Mirror Motion

The motion of the MEMS mirror array depends upon alignment accuracy during its bonding with the mirror substrate and driving electrode substrate. In case of any error in alignment during bonding process will cause misalignment in the centers of the mirror and electrodes and this will result in non-uniformity in the mirror motion, tilt direction and crosstalk between the tilt motion about X and Y axes. The relationship between applied voltage and mirror tilt angle is shown in Fig 3.10.

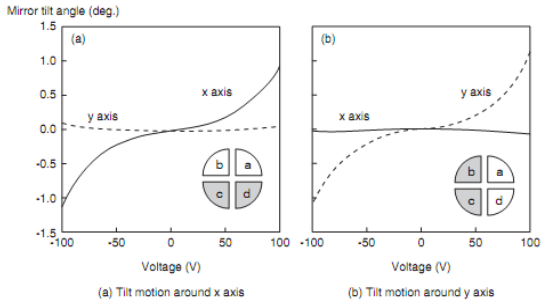


Fig 3.10 Relationship b/w Applied voltage & Mirror Tilt Angle

The graph shows the crosstalk for applied voltage and tilting angle between rotations along X axis and Y axis.As MEMS mirror array is the main functional components in the 3D MEMS optical

switch, hence its characteristics like reflectivity, aspect ratio, flatness, movement etc. directly affect the performance of the switch. The number of ports of the 3D MEMS switch is limited by the flatness and the size of the micro mirrors, as well as their fill factor and scan angle.

3.10 Key Advantages of MEMS Switches in DCCN

Key Advantages of using MEMS switches in data center is as;

- These photonic switches can switch large number of optical signals simultaneously hence in data center it can provide any-to-any connectivity between servers to transfer big data while offering a very low latency (millisec to nanosec depending upon manufacturer) which is very less than IP switches latency (usually in microsec) used in data centers.
- These switches are protocol and data rate independent so they can accommodate any data rate from 10Mbps to 100Gbps (currently maximum available) and beyond without any optics change.
- They can provide direct high capacity pure optical data paths b/w any TOR switches while bypassing the packet-based aggregation network to reduce network latency and to transfer the elephant traffic b/w servers when required.

SDN

4.1 Introduction

SDN stands for Software Defined networking. It is an emerging architecture that can be adaptable for designing and managing networks like DCCN (Data Center communication Network), CAN (campus area network) and service provider networks (ISP's) due to its dynamic approach, cost-effectiveness and manageability. The main objective of this architecture is to separate the control plane of the devices from its data/forwarding plane while provisioning programmability of its control plane. It presents a solution for controlling network and its management by centralizing and aggregating the control plane intelligence of the entire network infrastructure keeping its forwarding/data plane separate from it. Hence the network devices intact their data plane (switching fabric) but hand over their intelligence & control (routing, switching and forwarding decisions) to the controller which is a commodity server running NOS (network operating system). This allows the network administrator to dynamically configure and control the network devices through the applications programmed at the top of SDN architecture and makes the network highly flexible and easily manageable.

4.2 SDN Architecture

Currently in most network devices, control plane and the data plane exist on the same device or in other words data flow functions like switching, forwarding, routing and different protocols making these decisions all resides within that device. The main goal of SDN as defined in [7] is to "provide open user-controlled management of the forwarding hardware of a network element." Software defined networking (SDN) based architecture consist of following components also shown in Fig 4.1.

- Application Plane
- Control Plane
- Data Plane
- North Bound API's
- .South Bound API's
- East-West Protocols

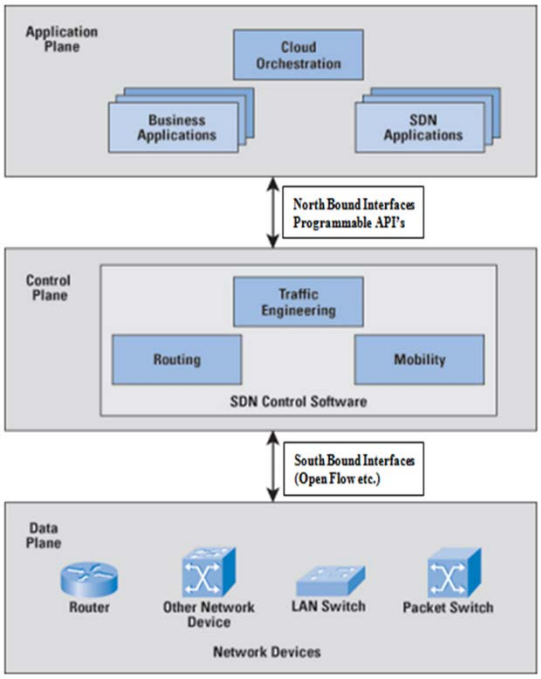


Fig 4.1: Software Defined Networking (SDN) Architecture.

4.2.1 Application Layer

It is the upper most layer of SDN (Software Defined Networking) architecture consisting of programmable applications that defines policies, control and logical operations to instruct the forwarding devices of the network. This layer retains network applications that assists the control plane in configuring network elements and can includes network features like forwarding schemes, manageability and security policies etc. Application layer can abstract the global view of all the network elements with the help of controller and can utilize that information for providing appropriate guideline to the control layer. In our design at application layer network monitoring applications like PRTG which continuously monitoring the link utilization of all the interfaces of the switch, management applications which provides CLI/ GUI of the network devices is running.

4.2.2 Controller or Control Plane

It is considered as brain of the network and is responsible to manage and programming the forwarding plane. The control plane provides an abstract view of all the network elements through topology database which uses LLDP and BDDP in our case. Building topology is critical for the smooth and accurate operation of other internal controller services like network configuration, host tracking, route planning, traffic switching, network monitoring and traffic engineering etc. It enables the network administrator to manage and define network operations (like applying custom policies/protocols implementation, routing) by dictating these operation to Network Elements like (Router, Switches) across the network. It consists of one or more than one software controllers which is used to communicate with the network elements of the forwarding plane. The open source controllers are Open day light (ODL), network operating system (NOX) C++ based, POX python based, floodlight, LOOM etc. but most active and supported by most vendors is ODL.

4.2.3 Data Plane

It is bottom most layer in SDN network architecture also known as infrastructure layer which comprises of forwarding Network Elements like (Router, Switches). Its main responsibilities include data forwarding, Statistics gathering & monitoring information. In our case it consists of OF switches at access layer and MEMS switches at aggregation layer.

4.2.4 North Bound API's

The software interfaces between the SDN applications and the software modules of the controller are known as the northbound APIs (application programming interfaces). The interface between the control plane and the application plane is usually called northbound interface.

4.2.5 South Bound API's

Standardized API's (Application Programed interfaces) through which SDN controller communicate with forwarding network infrastructure(like switches and routers) is known as south bound API. The protocol which SDN controller uses to manage and control the interface with various elements of NE's is known as south bound protocol which includes OpenFlow(OF), SNMP, BGP, PCEP etc. [18]. The first and most commonly used southbound interface protocol is OpenFlow (OF) which allows the SDN controller to manage and configure the switches.

4.2.6 East-West Protocols

These protocols are used to manage the interaction between various controllers in a multi-controller-based architecture. In a wider perspective, SDN presents a networking architecture in which Data traffic routing decisions is done outside of the physical switching hardware. Hence when the SDN along with PFFS and MOOOS is implemented in data center real network intelligence can be achieved. Hence to practically implement the concept of SDN into the communication network which is in our case is DCCN, there should be a logical architecture which is common in all network devices like routers, switches etc. and managed by SDN controller. Secondly there should be a secure standard protocol to communicate between SDN controller and network devices. Different vendor equipment may implement this logical architecture in different ways but from SDN controller perspective it functions like a uniform logical switch. AS OpenFlow (OF) fulfill both these requirements and is the First and most widely used interface b/w Infrastructure layer (Data plane) and control plane so it is explained.

4.3 Open Flow (OF)

OpenFlow(OF) is most common southbound interface of SDN architecture standardized by the Open Networking Foundation (ONF) which is nonprofit organization lead by the board of directors of seven renown companies that includes Microsoft, Google, Yahoo, Facebook, Verizon, NTT and Deutsche Telecom[23]. Initial version of OF is 1.0 which is developed by Stanford University but later on owned by ONF and its current version is 1.5 launched in December 2014.

It is an open standard communication protocol which explains the interaction of one or more than one control servers with SDN compliant switches. It provides a way to integrate multi-vendor switches to a single SDN controller and software based access till the flow table entries which directs routers and switches to guide the data traffic flowing through the network. An OF controller install the flow table entries in the OF compliance switches to forward traffic in accordance with these flow entries. Network administrator can use these flow tables to change data traffic flows and network layout. This protocol also provides management tools that can be used for packet filtering and to control topology changes. Almost all the well-known vendors like Cisco, HP, IBM, Brocade etc. offers routers and switches that support the OF protocol. OF or SDN compliance switches are of two types

4.3.1 Open Flow-Only Switches

These switches are also called Pure SDN switches or SDN-only switches which process all packets only by the OF pipeline and only support OF operation. In this switch all control functions like high-level Routing decisions will be done by a central controller and switch acts like a “dumb” device which is totally restricted to the data plane for forwarding data flows b/w different ports as dictated by SDN controller. OpenFlow-only switches architecture is shown in fig 4.3. The OF switch and SDN controller communicate with each other via OF protocol by exchanging messages like “Packet - IN”, “Packet - Out”, “Mod-forw-table”, “Get-Stats” etc.

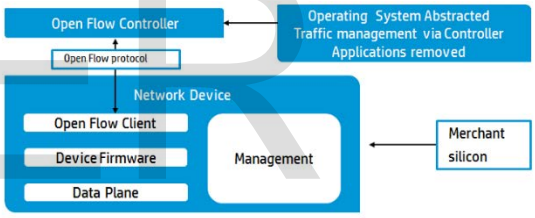


Fig 4.2 OpenFlow-Only Switch Architecture

4.3.2 OpenFlow-Hybrid switches

These switches support both OF operations (SDN) and traditional Ethernet switching operations like L2 Ethernet switching, L3 routing, VLAN isolation, ACL and QoS processing. These switches can classify data packets outside OF and then route that traffic either to normal pipeline or OF pipeline. Using hybrid switches network administrator can manage different types of traffic by configuring the SDN controller to discover and handle certain traffic flows like big Data while traditional routing & switching of rest of the traffic continues by different networking protocols configured on the network. Its architecture is shown in fig 4.3 as;

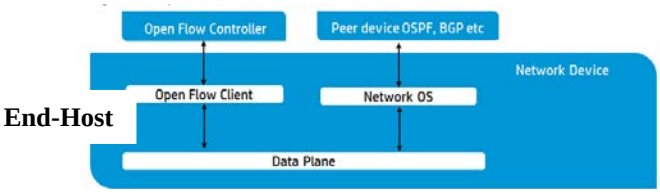


Fig 4.3 Hybrid Switch Architecture

4.3.3 OpenFlow (OF) Architecture

OpenFlow (OF) architecture consists of Infrastructure layer consisting of several OF compliance switches which are controlled by OF controller which can be one or more. OF switches communicate with the controller over a secure channel (secure socket layer) through a OF protocol. Each OF switch is further connected either to the other OF switch or Packet source and destination i.e. End Host or Server as shown in figure 4.4.

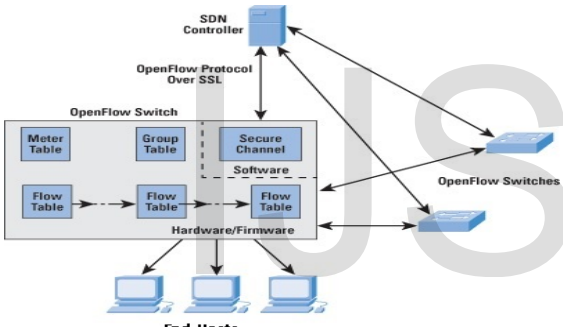


Fig 4.4 Basic Open Flow (OF) Architecture

Each OF switch can consists of multiple Flow tables to performs packet lookup and matching against flow entries in these tables to manage the packets flowing through it, Where a flow that could be a sequence of packets having same source port, same source/destination IP address or same MAC address or same VLAN (virtual local area network) tag or same I/P switch port, TCP/UDP port no etc.

In our proposed solution for DCCN OF switches is deployed in Access layer as TOR which is on one end is connected to the End-Hosts or Servers and on other end it is connected to Traditional switches and MEMS switches at Aggregation layer. ODL controller is deployed as a control plane which will communicate with OF switch through OF protocol over secure channel SSL / TSL (Transport Layer Security).

Logical architecture of OF switch consist of three tables (i) Flow Table (ii) Group Table (iii) Meter Table.

4.3.4 Flow Table

It is the basic building block of logical switch architecture which matches each incoming packet against a particular flow table which consists of multiple flow entries and specifies the action to be performed on that. This packet may pass through one or more flow table working in a pipeline manner. Using OF protocol a controller can add, delete and modify flow entries proactively or reactively (In response of incoming packet) from the flow tables of an OF switch as shown in fig 4.5.

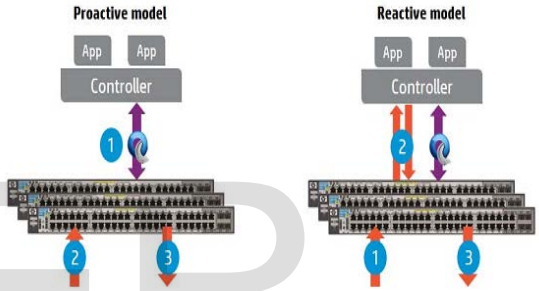


Fig 4.5 Proactive & Reactive Model of Flow Table Entries

In Reactive Model flow entries are built when something happens i.e. after arrival of packet it consult the controller and do action according to its instructions, whereas in Proactive model flow entries are built before something happens i.e. in advance and when packet comes it do not consult to controller. When first packet enters the switch OF agent software on the switch perform a flow table lookup in the ASIC in case of hardware, and in case of virtual switch in software flow table. If it is first packet so switch probably do not have any flow entry to match this packet, so it is a Table-Miss entry and as a default action switch forward this packet to controller as a “Packet-IN” massage which learns that a packet is arrive on a switch port, then controller reactively in response of it send a “Packet-Out” and “Flow-MOD” message to the switch to perform some action on the packet and create a flow table entry in its flow table. When packet arrive in the switch it starts matching the packet against the flow entries present in first flow table which might be proceed to other tables if no match found in first table. This matching is done in priority order means first matching entry have highest priority and is used in case of match

and instructions associated with that flow entry is carried out which is either actions set (modification, Packet Forwarding, or group table processing) or modify pipeline processing. This Pipeline processing permits the packet to be forwarded to succeeding table for further processing and its information b/w tables is communicated in the form of metadata as shown in fig 4.6.

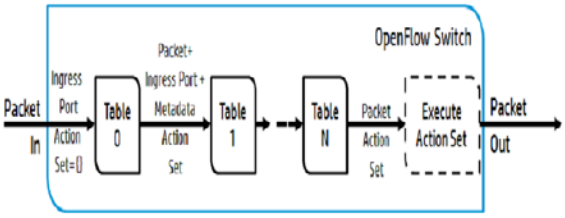


Fig 4.6 Multiple Table Pipeline processing

Pipeline processing pauses when instructions associated with the matching flow entry don't identify next table, in this situation packet is first modified and then forwarded. In case of no match the packet is marked as "Table-miss" flow entry which wildcards all match fields (all fields omitted) having a lowest priority of 0 and depending on "Table-miss" flow entry configuration, packet is forwarded towards controller through control channel by Packet-IN message or continue to next table or may drop. The flow chart of Packet processing is shown in fig 4.7.

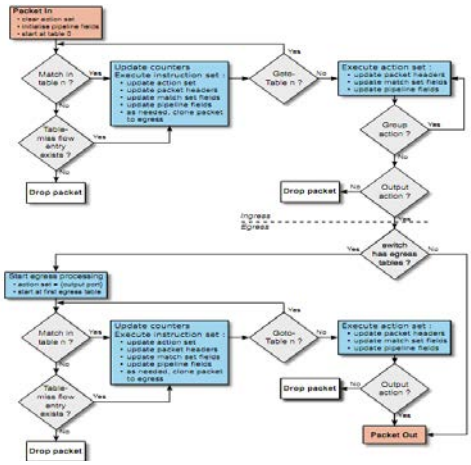


Fig 4.7 Flow Chart of Packet Processing through OF Switch

Every flow entry in a flow table consists of following components as shown in Table 4.1[32-+] as;

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie

Table 4.1 Flow Entry main components

4.3.4.1 Match fields

It is used for selecting packets whose field values matches match fields. In OF V1.5.1 there are 44 defined match fields which consist of input port, metadata of previous table, Eth SRC & DST addresses, Eth Frame types, Vlan ID, Vlan Priority, IP protocol, TCP/UDP SRC & DST, IPV4 SRC and DST,, ARP, ICMP, IPV6, MPLS, 802.1ah Provider backbone bridge and egress port etc. as shown in Table 2.4[32].

IN_PORT	IN_PHY_PORT	METADATA	ETH_DST	ETH_SRC	ETH_TYPE	VLAN_VID
VLAN_PCP	IP_DSCP	IP_ECN	IP_PROTO	IPV4_SRC	IPV4_DST	TCP_DST
TCP_DST	UDP_SRC	UDP_DST	SCTP_SRC	SCTP_DST	ICMPV4_TYPE	ICMPV4_CODE
ARP_OP	ARP_SPA	ARP_TPA	ARP_SHA	ARP_THA	IPV6_SRC	IPV6_DST
IPV6_FLABEL	ICMPV6_TYPE	ICMPV6_CODE	IPV6_ND_TAR	IPV6_ND_SLL	IPV6_ND_TLL	MPLS_LABEL
MPLS_TC	MPLS_BOS	PBB_ISID	TUNNEL_ID	IPV6_EXTHDR		

Table 4.2 Flow Entry Match Field

4.3.4.2 Priority

It shows relative priority of flow entries. For a specific flow table Priority and Match fields together make a unique flow entry identity.

4.3.4.3 Counters

Update when packets match. E.g. it includes No of bytes received, No of packets received and dropped etc.

4.3.4.4 Instructions

To modify the pipeline processing or the action set.

4.3.4.5 Timeouts

Maximum idle time for a switch before which flow is expired.

4.3.4.6 Cookie

These are the data value chosen by the controller and is opaque. It can be used by the OF controller to filter flow statistics entries, flow deletion and modification requests. These values are not used while packet processing.

4.3.5 Group Table

Flow table can direct its flows to Group Table to carry out various actions that can affect more than one flows. A group table consist of group entries which are identified its group identifier. Group entry main components in group table are as

Group Identifier	Group Type	Counters	Action Bucket

Table 4.3 Group Entry main components in a group Table

4.3.5.1 Group identifier

Each group entry is uniquely identified by this 32 bit identifier.

4.3.5.2 Group type

To support different group types marked with "Required" and "Optional".

4.3.5.3 Counters

Counters are used to count packets processed by a group.

4.3.5.4 Action buckets

It contains a list of actions to execute in order that will modify packets and forward them to a port. This bucket of actions always implemented in the form of set of action. A group entry can involve zero or more buckets except group type marked with "Required: Indirect" consist of only one bucket. In case if a group have no bucket then it will straight away drops the packets.

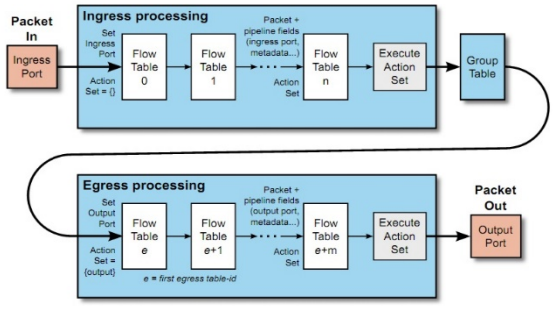


Fig.4.8 OpenFlow packets pipeline Processing

OF packet pipeline processing is done in two steps i.e. Ingress processing followed by Egress processing. First and main stage is Ingress processing, this is the stage in which packet enters the OF switch and can involve more than one Flow tables lookup, while second stage is Egress processing which starts after the determination of output port and may or may not involve flow tables.

4.3.6 Meter Table

It can carry out performance related actions on a flow. It consists of meter entries defining per-flow meter. Per-flow meter enable OF to implement QoS policies like rate limiting for a number of flows to a specific bandwidth, and DSCP (Differentiated service code point) based metering can classify packets into different groups according to their data rate. Meters are directly attached with flow entries instead of ports and after measuring, control the aggregate rate of all packet flows and can specify a meter action. Meter Entry main components are shown as;

Meter Identifier	Meter Bands	Counters

Table 4.4 Meter Entry main components in meter table

4.3.6.1 Meter Identifier

It is a 32 bit identifier which is used to uniquely identify each meter entry.

4.3.6.2 Meter Bands

It is a list of meter bands which are not in ordered form and defines the way to process the packets by specifying the rate of each band. Meter bands are used to define the meter behavior with packets for multiple ranges of meter rate measured which is calculated by all directing packets to that meter from all flow entries. A meter can consist of more than one meter bands but a packet is processed by only one meter band.

Band Type	Rate	Burst	Counters	Type specific arguments

Table 4.5 Main Components of Meter Band

4.3.6.2.1 Band Type

It defines the way by which packets processes.

4.3.6.2.2 Rate

It is a target rate for a Meter band. For the selection of meter band Meter uses it, usually the band for the lowest rate is applied.

4.3.6.2.3 Burst

Granularity of the meter band is defined by it.

4.3.6.2.4 Counters

When meter band processes packets it is updated.

4.3.6.2.5 Type specific arguments

Optional arguments for some band types. Meter bands ranking depends on the increase in the target rate, rank 0 is of the default band while from rank 1 configurable band starts. When the meter measured rate is exceed its target rate the packet is processed by only one meter band.

4.3.6.3 Counters

It is used to calculate number of packets processed by a meter. Different flow entries may use different meter, same meter or no meter at all in same flow table. With the use of different meters in a flow table, Disjoint set of flow entries can be metered independently.

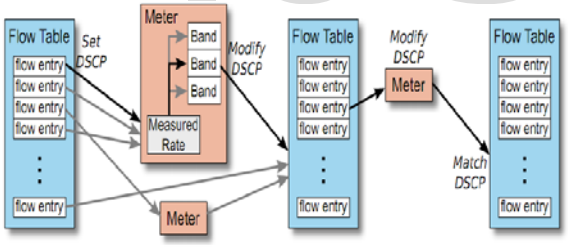


Fig. 4.9 Meters and Hierarchical DSCP metering.

4.3.7 OpenFlow Channel and its Protocol

Each OF switch is connected to the controller through an interface called OF channel. It is usually encrypted by TLS (Transport Layer Security) or SSL to provide secure OpenFlow OF channel but can also run directly over TCP. Controller uses it for managing and configuring switches, receiving events from them, sending packet out messages back to the switch etc., and the protocol which it uses for this purpose is called OF protocol. This protocol describes the massages that are exchanged between OF switch and OF controller and empowers the controller to direct the logical structure of the OF switch. These messages can be categorized into three types.

(i) Controller_to_Switch
(ii) Asynchronous
(iii) Symmetric.[20]

4.3.7.1 Controller_to_Switch Messages

These are the messages which are initialized by the controller to manage the logical state of the switch like configuration, flow table and group table etc. It also includes “Packet-Out” messages which are sent by the controller to the switch and as a response switch will forward the packet to one of its output port instead of dropping it.

4.3.7.2Asynchronous Messages

This type of messages is initiated by the switch which includes status messages to update the controller about the change in the switch state and network events. It also includes “Packet-In” message which is used by switch to send a packet to the OF controller if there is no match found in its flow table.

4.3.7.3 Symmetric Messages

These are the messages which can be initiated by either controller or the switch e.g. Hello messages which is exchanged between switch and controller at the time of connection establishment or Echo messages which is used to verify that device is properly working and to measure latency and bandwidth of connection between switch and controller.

Message	Description
Controller-to-Switch	
Features	Request the capabilities of a switch. Switch responds with a features reply that specifies its capabilities.
Switch Configuration	Set and query configuration parameters. Switch responds with parameter settings.
Flow Table Configuration	Configure/Modify behavior , property, flags of flow table
Modify State	Add, delete, and modify flow/group entries and set switch port properties.
Read-State or Multipart	Collect information from switch, such as current configuration, statistics, and capabilities.
Packet-out	Direct packet to a specified port on the switch.
Barrier	Barrier request/reply messages are used by the controller to ensure message dependencies have been met or to receive notifications for completed operations.
Role Request	Set or query role of the OpenFlow OF channel. Useful when switch connects to multiple controllers.
Bundle	Creation, closing, committing and discarding of bundles by controller
Set Asynchronous Configuration	Set filter on asynchronous messages or query that filter. Useful when switch connects to multiple controllers.
Asynchronous	
Packet-In	Transfer packet to controller.
Flow Removed	Inform the controller about the removal of a flow entry from a flow table.
Port Status	Inform the controller of a change on a port.
Controller Role Status	Inform the controller about its role Changing
Table status	Update to controller when table state changes
Request Forward	Update to other controller about modification in state of group and meters
Controller Status	Report to all other controller about change in controller status
Symmetric	
Hello	Exchanged between the switch and controller upon connection startup.
Echo Request/Reply	Echo request/reply messages can be sent from either the switch or the controller, and they must return an echo reply.
Error Messages	Notify controller of error or problem condition.
Experimenter	For additional functions.

Table 4.6 OpenFlow Messages

4.4 Benefits of SDN

SDN can be a promising technology for managing and providing solutions to various challenges encountered in Data center network. Using SDN technique administrator of the network do not have to implement different protocols and configure custom policies on every device in the network, he just have to defined all these on the centralized location from where all devices are controlled and directed by SDN controller.

Data centers are encountering scalability issues especially as the number of server and the number of VMs (virtual machines) running on them increases and later their need for migration (VM motion) increases. Virtual machine migration and MAC address Table updating requires a high bandwidth which increases latency in the network and hence decreases overall performance of the network and in traditional data center architectures user may experience interruption in accessing applications. Hence I propose that implementation of SDN along with "All optical switching" and other virtualization technologies like OTV can resolve this issue. AS by implementing SDN and All Optical Switching you can dynamically provide high bandwidth circuit b/w the two points to accommodate such traffic (elephant traffic). Whereas different virtualization techniques can be implemented as SDN applications like OTV which provides you a tunnel to exchange layer 2 traffic over the Layer 3

HFPFMAOS

5.1 HFPFMAOS Solution

In this thesis the solution I propose for DCCN challenges and issues is HFPFMAOS(Hybrid Flow based packet filtering, forwarding & MEMS based all optical switching). My proposal implementation consists of eight steps.

- Deployment of SDN controller & OF in access layer switches i.e. TOR switches and their connection with SDN controller over SSL.
- Deployment of MOOOS plane at Aggregation level and its connection with SDN controller.
- Building Topology Database.
- FPFSF (Flow based Packet Filtering and Forwarding).
- Outgoing Ports monitoring and calculation of Links utilization and congestion notification to SDN controller on reaching specific threshold.
- Flow Table lookup and Inspection of flow entries to identify the flows utilizing more bandwidth and notifying that flows source and destination.
- Creation of High bandwidth data paths between TOR switches corresponding to identified Source and destination
- Switching of that Specific flows traffic through High bandwidth link created through MOOOS plane, when that specific flow vanishes or come back to its normal data rate, switches it back to its route and tear down this High bandwidth link.

5.1.1 Deployment of SDN controller & OF in Access

In first step I will deploy SDN controller and enable OF in all access layer switches (TOR) while rest of the network remains same i.e. running traditional switches. Then I connect every OpenFlow OF enabled Access switch with SDN

controller over secure encrypted SSL (secure socket layer)/TCP connection. This connection is used for communication between SDN controller and OF switch through OF protocol. This connections is used by SDN controller and various north bound applications, like for initial handshaking messages(OFPT_HELLO,FEATURE_REQUEST_MESSAGE controller to switch & FEATURE_REPLY_MESSAGE from switch to controller), Topology discovery using LLDP (Link Layer Discovery Protocol)&BDDP (Broadcast Domain Discovery Protocol), building data base, Alert messages (PACKET IN to controller & PACKET OUT message from controller), instructions from controller, building link utilization updates, congestion notifications and link latency, traffic analyzation for determining source and destination of flows utilizing more bandwidth, for creation of high bandwidth data paths through MEMS switches and modification in flow entries of flow table to switch specific flows over all optical path between OF switches.

5.1.2 Deployment of MAOS (MEMS based all optical switching) Plane

We will deploy MAOS (MEMS based all optical switching) plane along with traditional switches at aggregation level. In this solution the existing packet based switching continue working along with newly suggested MAOS for providing dynamically high bandwidth data paths between servers for switching of elephant traffic whenever it is required. Moreover for centralized control, management of network elements (NE's) and traffic flow control, it is connected with centralized SDN controller. The aim of this deployment is to provide dynamically high bandwidth data paths b/w the servers along with normal data paths in order to support long persistent data flows, which results in less chances of bandwidth bottlenecks (i.e. point of congestion) as buffers will not overflow and data packets of Latency sensitive applications keep flowing through the network with normal latency. The architecture of my proposed solution HPMAOS (Hybrid Packet switching and MEMS based all optical switching) for DCCN is as shown in fig 5.1

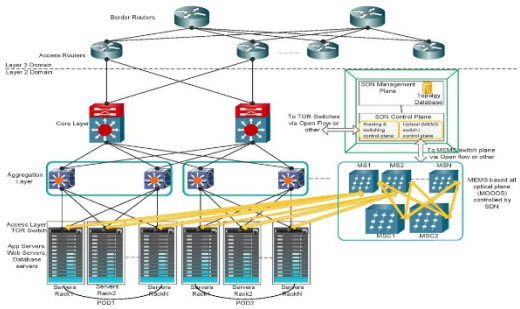


Fig 5.1 Architecture of HPMOOOS for DCCN

In above fig I added a MAOS plane at aggregation layer in parallel with Traditional switches where OF switches (TOR) are connected simultaneously with Aggregation switches and MEMS switches. Black lines shows active links of traditional packet switched network whereas Orange lines shows inactive high bandwidth TORS connections with MEMS switches in MAOS Plane to handle long persistent data flows.

5.1.3 Building Topology Database

In this solution HFPFMAOS, DCCN will be running both OF switches and traditional switches along with MEMS Switches i.e. hybrid network and employs OF protocol as South bound Protocol between SDN controller and OF switches. SDN controller build a topology data base of OF enabled devices in the network by discovering OF switches, available outgoing links and all the hosts attached with each OF switch. This building of topology database will involve three discoveries i) discovery of Switches ii) Discovery of links iii) Discovery of Hosts

5.1.3.1 Discovery of OpenFlow OF Switches

As part of initial configuration like "Bootstrapping" every OF enabled switch is assigned an IP address and configured with TCP port number & one IP address of Master controller and a range of IP addresses of slave controllers, as in the case of master controller connection failure OF switches can connect to slave controller. On startup OF switches will establish an encrypted and secure connection through TLS with the controller and is assigned a specific Switch ID depending upon its Pod ID and Rack ID i.e. OS1-1 (O→ OF, S→ Switch, 1→ pod no, -1→ Switch no) and a list of outgoing interfaces over which inter-racks,

inter-pod, b/w data centers and end client traffic is forwarded. These interfaces also consist of interfaces which connects each Access switch to MEMS switch, Interface through which OF switch is connected with the controller and interfaces to which Hosts are connected.

5.1.3.2 Discovery of Active Links

For Links discovery we can employees a OFDP (OpenFlow discovery protocol) which leverages L2 discovery protocol named BDDP (Broadcast domain discovery protocol)[22] to find indirect available multi-hop links (routes) between OpenFlow OF switches. This method is used by most famous Open Source controllers like ODL & Floodlight etc. This BDDP protocol is similar to L2 single hop discovery protocol LLDP (Link Layer discovery protocol used for discovering direct links b/w OF switches) with the difference in the destination MAC address and Ethernet type field values. BDDP uses broadcast address (ff:ff:ff:ff:ff:ff) in destination MAC address field instead of multicast address (01:80:c2:00:00:0e, or 01:80:c2:00:00:03, or 01:80:c2:00:00:00) used in LLDP, and uses Ethertype field value as 0x8999 instead of 0x88cc in case of LLDP. The frame structure of LLDP and BDDP is shown in Fig 5.1 and Fig 5.2.

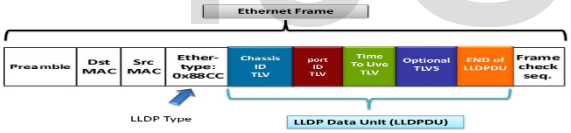


Fig 5.2 LLDP Frame Structure

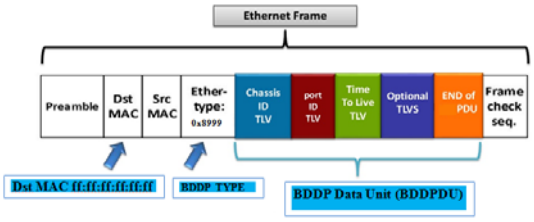


Fig 5.3 BDDP Frame Structure

OF controller uses OF protocol and as part of initial handshake send "OFPT_FEATURES_REQUEST" message to all switches to determine its features like identity, basic capabilities etc. and Switches in response of it send "OFPT_FEATURES_REPLY" message to controller. As a result controller gets all the information of the OF switches connected in the network like Switch ID, a list of active ports mapped with their respective MAC addresses in their administrative domain. After this the controller encapsulates a BDDP packet inside each "OFPT_PACKET_OUT" message for every active port of Each OF switch accepts the controller port as shown in fig 5.3.

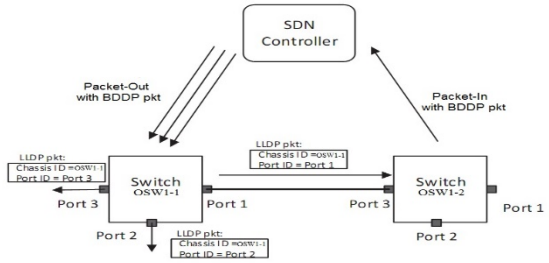


Fig 5.4 Links discovery Via OFDP

When OF Switches receives "OFPT_PACKET_OUT" messages, they will install the flow entries in their flow table and forward these BDDP packets through their respective ports with broadcast address (ff:ff:ff:ff:ff:ff) in the field of destination MAC address, Ethertype as 0x8999, TTL value 255, and appropriate chassis ID & Port ID indicated in their respective TLV fields of the BDDP message payload. This BDDP messages reaches to traditional switch which sees a broadcast address (ff:ff:ff:ff:ff:ff) in the field of destination MAC address then it decrement TTL value by 1 and immediately flood the packet out of all its ports. When these packets reaches to OpenFlow OF switches which sees Ethertype as 0x8999 they directly forward these packets to controller by encapsulating them in "OFPT_PACKET_IN" messages as shown in fig 5.5.

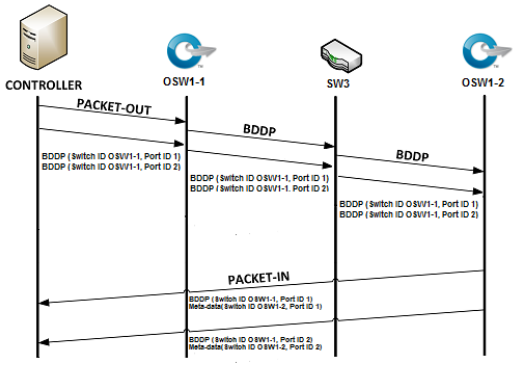


Fig 5.5 Messages b/w OpenFlow OF switches and controller for Links

These "OFPT_PACKET_IN" messages include the metadata of OpenFlow OF switches (like Port ID, Switch ID etc.) where BDDP packets are received. These messages are exchanged in one way between OpenFlow OF switches and their controller. Similarly these massages are exchanged in opposite direction as well. After receiving this "OFPT_PACKET_IN" messages controller is able to discover indirect multi-hop links between OpenFlow OF switches based on information present in BDDP and metadata and store them in their database to build network topology. Most important thing controller uses TTL value to count no of hops between OpenFlow OF switches. This topology discovery process is repeated periodically with a default value of 5 seconds. In this whole discovery process the total number of "OFPT_PACKET_IN" messages received by controller is twice the no of active links "L" available on the broadcast domain and can be calculated as;

$$\text{Total}_{\text{Rx OFPT\_PACKET\_IN}} = 2 \times L$$

The total number of "OFPT_PACKET_OUT" messages sent by controller can be calculated as;

$$\text{Total}_{\text{Tx OFPT\_PACKET\_OUT}} = \sum_{i=0}^{S} Pi$$

Where S is no of OpenFlow OF switches each with P active ports. In our reference topology there are 4 OpenFlow OF switches each having 2 active ports so total "OFPT_PACKET_OUT" messages sent by controller are 2+2+2+2=8. The total number of BDDPOFPT_PACKET_OUT messages send by controller to a switch can be reduced by adopting OFDPv2 [23] in which Port ID TLV field is set to 0 and will be ignored while in source MAC address field MAC address of the port has been set through which it is to be sent out. According to OFDPv2 the number of "OFPT_PACKET_OUT" message sent by controller is equal to one per switch and in total it is equal to the number of switches which is calculated as.

$$\text{Total}_{\text{Tx OFPT\_PACKET\_OUT}} = S$$

For connectivity with MAOS plane we don't need link discovery as one port of every OpenFlow OF switch is physically connected with one port of MEMS switch but it becomes active momentarily for dynamically creation of paths between any two points by the controller, and we will put its all port connectivity information and flows entries statically in topology database.

Switch ID	Port	Direct connected Port MAC address	Far End OSW ID & Port Rx "Packet_IN" MSG	Type	Status
OSW1-1	Eth 1	SW3 port0 MAC address	OSW1-2, Eth 1 ,1	Dynamic	Up
	Eth1	SW3 port0 MAC address	OSW2-1, Eth 1 ,3	Dynamic	Up
	Eth1	SW3 port0 MAC address	OSW2-2, Eth 1 ,3	Dynamic	Up
	Eth1	SW3 port0 MAC address	OSW1-2, Eth 1 ,3	Dynamic	Up
	Eth 2	SW4 port0 MAC address	OSW1-2, Eth 2 ,1	Dynamic	Up
	Eth 2	SW4 port0 MAC address	OSW2-1, Eth 2 ,3	Dynamic	Up
	Eth 2	SW4 port0 MAC address	OSW2-2, Eth 2 ,3	Dynamic	Up
	Eth 2	SW4 port0 MAC address	OSW1-2, Eth 2 ,3	Dynamic	Up
	Eth 3	MSW1 port 1		Static	Down
	Eth 4	Controller		Static	Up
	Eth 5				Down
	Eth 6				Down
	Eth 7	bb:bb:bb:bb:bb:bb		Dynamic	Up
	Eth 8	aa:aa:aa:aa:aa:aa		Dynamic	Up
OSW1-2	Eth 1	SW3 port1 MAC address		Dynamic	Up
	Eth 2	SW4 port1 MAC address		Dynamic	Up
	Eth 3	MSW1 port 1		Static	Down
	Eth 4	Controller		Static	Up
	Eth 5				Down
	Eth 6				Down
	Eth 7	cc:cc:cc:cc:cc:cc		Dynamic	Up
	Eth 8	dd:dd:dd:dd:dd:dd		Dynamic	Up

Table 5.1 Topology database

5.1.3.3 Discovery of Host

For available host discovery attached with the OpenFlow OF switches we employees two ways.

(i) Ports for which OpenFlow OF Switches do not send "Packet_In" message, is marked as Host port during link discover as BDDP "Packet_Out" message is sent out of all active ports of all switches

(ii) Host when connect to OpenFlow OF Switch send GARP message.

GARP is an advance notification mechanism in a broadcast domain to update the controller about host discovery and inserting MAC address flow entry in the flow table of OpenFlow OF switches (same like HSRP & VRRP[24] which uses GARP to update the MAC address table of L2 switches), update ARP table of other hosts before their ARP request, when a new host is connected to a switch or a host IP address or MAC address is changed due to failover or new NIC in case of VM motion. Gratuitous ARP is actually special ARP request packets in which both source and destination IP is the IP of the source (host) generating the gratuitous ARP request, destination MAC address field consist of broadcast MAC address (ff:ff:ff:ff:ff:ff)[25] and Ethertype field is set to 0x0806. A GARP request message consist of following parameters;

- Destination MAC address: FF:FF:FF:FF:FF:FF (broadcast)
- Source MAC address: Host's MAC address
- Source IP address = Destination IP address: IP address of the host sending GARP
- Type: ARP (0x0806)
- Example of GARP is shown in fig 5.6

```
Ethernet II, Src: PaloAlto_09:21:12 (00:1b:17:09:21:12), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
    Destination: Broadcast (ff:ff:ff:ff:ff:ff)
    Source: PaloAlto_09:21:12 (00:1b:17:09:21:12)
    Type: ARP (0x0806)
Address Resolution Protocol (request/gratuitous ARP)
    Hardware type: Ethernet (1)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
   [Is gratuitous: True]
    Sender MAC address: PaloAlto_09:21:12 (00:1b:17:09:21:12)
    Sender IP address: 10.66.24.67 (10.66.24.67)
    Target MAC address: Broadcast (ff:ff:ff:ff:ff:ff)
    Target IP address: 10.66.24.67 (10.66.24.67)
```

Fig 5.6 GARP Request Message

After completing GARP every host will ping each other to confirm connectivity and reachability with each other. At this moment all hosts have learnt each other MAC addresses and their respective IP addresses which can be seen from their ARP table and after discovery SDN controller suppress link discovery on these ports of OpenFlow OF switch to which hosts are attached to prevent BDDP packet propagation. This suppression is similar to BPDU guard security feature in traditional switches which prevent BPDU propagation on host connected ports enabled with "Portfast" feature [27].

IP Address	Physical Address
10.0.0.2 (Host 2 IP Address)	bb:bb:bb:bb:bb:bb (Host 2 MAC Address)

Table 5.2 ARP Table of Host 1connected with OSW1-1

IP Address	Physical Address
10.0.0.1 (Host 1 IP Address)	aa:aa:aa:aa:aa:aa (Host 1 MAC Address)

Table5.3 ARP Table of Host 2 connected with OSW1-1

5.1.4 Route-Tag assignment to Discovered routes & Forwarding Table Buildup

As the controller has a complete road map of all the network using a topology discovery and has a centralized database, so it has complete information about every outgoing interface of every OpenFlow(OF) switch and a list of all routes and destinaions (MAC/IP addresses) reachable through it. After Topology discovery SDN controller assign a Route-Tag for each discovered route. The main purpose of assigning a Route-Tag is to uniquely identify each learned routes and build a forwarding table consisting of these Route-Tag and all the destination (MAC/IP addresses) reachable through them. SDN controller install this forwarding table into OF switch which lookup this table for forwarding a flow towards outgoing interface based on Route-Tag. Traditional switches forward this Flow frame to the next hop depending upon Source addrees and destination address as their CAM(content addressable memory) table also have list of destination MAC addreses reachable across every link(port). When this Flow reaches to OpenFlow OF switch it again perform flow tables lookup for best match and after matching its destination address, forward this flow to the concerned Host port.

5.1.5 Flow Table built-up by inserting Flow enteries

When a flow comes to OpenFlow OF switch it match each incoming packet against a particular flow table or a group of flow tables which consists of multiple flow entries and specifies the action to be performed on that. This matching can be of any type like matching ingress port, Src/Dst MAC address, Src/Dst IP address, VLAN ID, TCP/UDP Port no etc. Here in this thesis I will use Hybrid Model of Flow Table Entries which is a combination of both proactive model and reactive modal to provide flexibility & support for various type of traffic passing through DCCN while providing low-latency for delay sensitive traffic. For normal traffic like web surfing, data transfer/ file transfer, peer to peer traffic I will use Reactive model of flow entries while for delay sensitive applications that requires low latency like audio/video call, live transmissions, banks transection data I will use Proactive model of flow entries. When traffic (L1 Flow/L2 Frame/L3 Pscket/L4 segment or packet etc) comes in the OpenFlow OF switch it searches its one or more flow tables for matching Source port/ L2 matching Src or Dest MAC address/ matching VLAN/ L3 matching Src or Dest IP address matching / L4 matching TCPorUDP Port-no respectively as shown in fig.

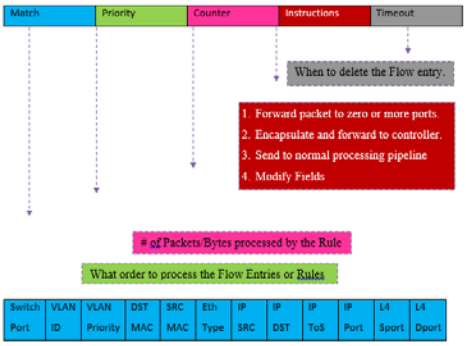


Fig 5.7 Anatomy of Flow Table Entry of OpenFlow OF

If it finds any flow entry that matches the fields of data traffic then it carry out the action specified by the flow entry or it is forwarded to group table to carry out multiple actions. Actions can be one or multiple which can include sending matching packets to the controller, flooding all ports, droping the packets, changing multiple header like IP, MACS, TCP/UDP ports and Pop, Push, Swap MPLS labels.Priority field is also critical as flow enteries are sorted out by priorities andin case of multiple flow entries matched flow entry with highest priority is used and others will be ignored. Few common actions are as;

- ALL: forwarding to all interfaces except incoming interface.
- Flood: send packets to all ports except the incoming port.
- Table: Perform actions in the flow table.
- Normal: Forward using traditional Ethernet.
- Local: send to its local networking stack.
- Controller: encapsulate packet and send to controller.

Moreover both flow table and group table consist of counters which is updated upon matching and passing of every flow. Counters can be of any type i.e. it can be Per Table, per Flow, Per Port or Per Que as Shown in table.

Per Table	Per Flow	Per Port	Per Queue
Active Entries	Received Packets	Received Packets	Transmit Packets
Packet Lookups	Receive Bytes	Transmitted Packets	Transmit Bytes
Packet Matches	Duration (Secs)	Receive Bytes	Transmit Overrun Errors
	Duration (nsecs)	Transmitted Bytes	
		Receive Drops	
		Transmitted Drops	
		Received Errors	
		Transmitted Errors	
		Receive Frame allignment error	
		Receive Overrun Errors	
		Receive CRC Errors	
		Collisions	

Table5.4 Types of Counters

In order to avoid conjestion SDN controller will continously get the Statistics of outgoing active links (Ports) by sending OFPMP_PORT_STATS command to OpenFlow OF switch which in reply send all the statistics of the requested ports i.e. number of Rx_packets, Tx_packets, Rx_bytes, Tx_bytes, duration_sec (alive duration), dropped packets and Tx/Rx errors. Based on the replied statistics controller calculate the link utilization of all the desired ports. The formula[28] to calculate the Link utilization is as;

$$\text{Input utilization} = \frac{\Delta \text{ifInOctets} \times 8 \times 100}{(\text{number of seconds in } \Delta) \times \text{ifSpeed}}$$

$$\text{Output utilization} = \frac{\Delta \text{ifOutOctets} \times 8 \times 100}{(\text{number of seconds in } \Delta) \times \text{ifSpeed}}$$

- ΔifInOctets: The Δ (or difference) between two poll cycles of collecting the snmp ifInOctets object, which represents the count of inbound octets of traffic.
- ΔifOutOctets: The Δ between two poll cycles of collecting the snmp ifOutOctets object, which represents the count of outbound octets of traffic.
- IfSpeed: the speed of the interface, as reported in the snmpifSpeed object.

Fig 5.8 Formula to calculate Link Utilization

Controller maintains a specific link utilization threshold for all the outgoing interfaces it crosses the specified limit it consult its flowtable to filter flows with outgoing interface crossing a specific threshold. Out of these flows it check which flow have maximum counter value i.e max bytes processed (Received or transmitted) and check that flow source and destination address. After determination of Source and destination of that flow SDN controller will consult its topology database to find ports on the source and destination OpenFlow OF switch, and its corresponding ports on the MOOOS plane. After specifying ports SDN controller then check the availability of ports and creates a Highband dat pathpath through the MEMS Plane.

In casace of no match then it is a table miss entry so as a default action it is Punt to the controller which then ïnstruct the switch to perform some actions and insert flow entry by sending

"Packet_Out" messages or "Flow_Mod" message. "Flow_Mod" message consist of multiple things like Buffer ID, Timeout, Actions, Priority etc Also flow entry can be a permanent or for a specific time which is of two types "Idle timeout" and "Hard Timeout". Idle timeout means that's if there is no matching flow entry request in that specific time remove the entry and Hard timeout is max residing time of an entry in the flow table no matter it is alive matching entry if, Hard timeout is deactivated if it is set to 0. For Example in our reference topology diagram after host discovery through GARP Host 1 send a HTTP request to Host2 (say web server), since it is a TCP conversation so it starts with SYN message. Host 1 send a SYN message to OSW1-1 switch, when switch receive this packet it check its flow table since it is first packet so probably there is no flow entry matching with the packet This is called table miss flow entry. So as a default action switch punt this TCP packet to controller by encapsulating it in "Packet_IN" message. This Packet_IN message includes complete TCP packet or its Buffer ID (say Buffer ID =250 location where switch buffers the entire TCP message). So controller will perform couple of actions which may includes sending "Packing_Out" message or "Flow_Mod" message back to the switch, where "Packet_Out" message includes the complete encapsulated TCP packet or the reference buffer ID where switch stores this packet and instructions from the controller to the switch to do with the specific packet.In case of "Packet_Out" message controller instruct the switch OSW1-1 to simply forward the TCP SYNpacket reference with buffer ID=250 out of its port8 to Host2. The "Flow_Mod" message instruct the switch to install a new flow entry in its flow table. The flow entry helps the switch to know what to do in future, if similar packet arrives at the switch based on matching its fields and masks. This message instruct the switch that any TCP request from the IP or MAC of Host1 to the IP or MAC of Host4, send all to the Port 8. It also instruct the switch that TCP message which u buffered at BufferID = 250, release that packet from buffer and apply the actions in this message as well. H4 in reply send a SYN/ACK packet to the switch when this messages is received by switch then there is no flow entry in the switch from Host2 to Host1 so again it is a table miss so switch encapsulte this SYN/ACK packet into "Packet_In" message and send it to the controller with a reference buffer ID (say BufferID=251 . Controller in reply send a Packet_Out message and a Flow_mod message to the switch which instruct the switch to add up a flow entry in the flow table and perform some action which is forwarding the SYN/ACK message to port7. After all of this rest of the conversation between Host1 and Host2 would not go to the controller as switch have flow enteries into the flow table to guide the switch what have to do with the packet. Like ACK messages and HTTP reply is directly forwarded by the switch as shown in below figs5.9 (a) & (b).

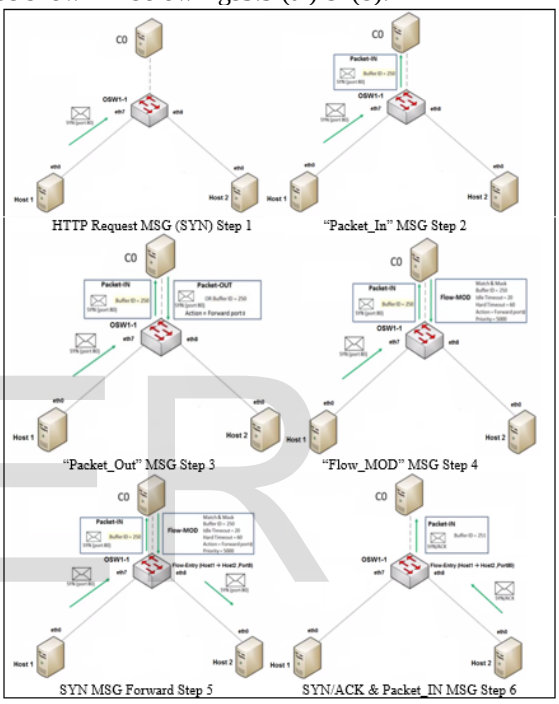


Fig 5.9(a) Open Flow messages for HTTP request

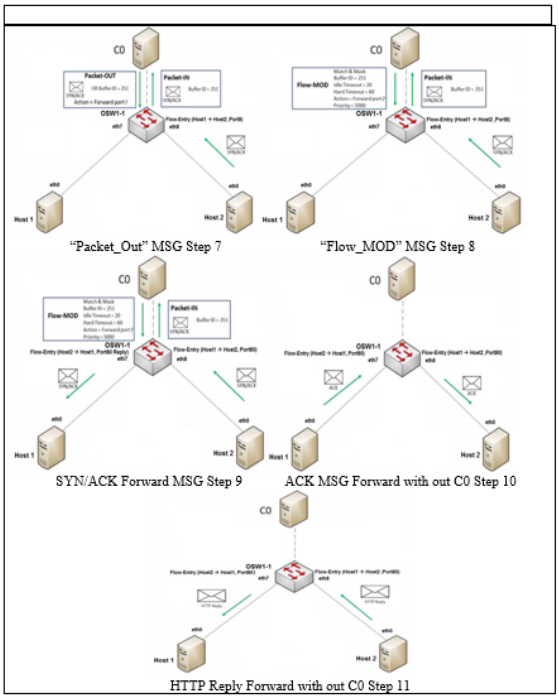


Fig 5.9(b) Open Flow messages for HTTP reply

The OF switch flow entries in its flow table will look like as shown in Table 5.5.

I/P Port	Src MAC	Dst MAC	VLAN ID	VLAN PCP	VLAN type	Ether_T	SRC IP	DST IP	IP Prot	IP ToS	L4 SPort	L4 DPort	Priority	Action	Counter
contr	*	ff:ff	*	*	*	0x8999	*	*	*	*	*	*	10	Local	11
*	*	*	*	*	*	0x0806	*	*	*	*	*	*	10	Local	2
*	*	aa:aa	*	*	*	*	*	*	*	*	*	*	80	Port 7	50
*	*	bb:bb	*	*	*	*	*	*	*	*	*	*	80	Port 8	60
*	*	dd:dd	*	*	*	*	*	*	*	*	*	*	80	Port 1,2	60
*	*	*	10	*	*	*DST routing	10.0.0.9/8		*	*	*	*	67	Port 1	20
*	*	*	20	*	*	*DST routing	10.0.0.16/8		*	*	*	*	67	Port 2	15
*	*	*	*	*	*	*	*default route		*	*	*	*	50	Contr,Port1	100
Port7	aa:aa	bb:bb	20	0	0x0800	10.0.0.1	10.0.0.2		0x06	*	38661	80	67	Port 8 TCP	20
Port8	bb:bb	aa:aa	20	0	0x0800	10.0.0.2	10.0.0.1		0x06	*	80	38661	67	Port 7 TCP	20

Table 5.5 Flow Entries in Flow Table of OF Switch OSW1-1

First entry tells the switch that if any packet comes from controller port check its Ether_type if it is 0x8999 broadcast it out of all the switch ports except the port from where it comes as it is BDDP message and update the counter. Sec entry tells the switch that don't care all the fields if the Ether_type is 0x0806 send it all the switch ports as it is GARP request and update the counter . Third, forth and fifth line is L2 matching and forwarding it tells the switch that if don't care other fields if the packet comes specfied DMAC carry out specific action identified by Action field and update the counter. Sixth and seventh line tell ip routing of packets to specific port depending upon its destination IP. Eighth line is the default route and last twoflow entries are TCP flow entry of HTTP request and reply as shown for our reference topology.When the long persistent high bandwidth data flow also known as "elephant flow" start flowing along with majority of short flows also known as "Mouse flows", this creates point of constriction and bandwidth bottleneck on the links between the TOR switch and the Aggregation layer switch. This High bandwidth data flows which can be due to VM motion or data base backup or other applications like Hadoop etc. causes the switch buffers to overflow as a result switch processes the packet slowly which increases network delay and badly effect latency sensitive applications. As now a days on a server number of virtual machines are running for different kind of applications and due to some reason one application running on a virtual machine in Server Rack 1 start taking high resources like bandwidth and CPU etc. effecting other applications running on the same server. This enforce administrator to move this virtual machine to other server with enough resources for this application causing VM motion to Server Rack 2 of same POD, this creates bandwidth bottleneck and increases latency on the network links b/w server & TOR switch and Aggregation switch as switch buffers starts overflowing and shown in fig 5.10

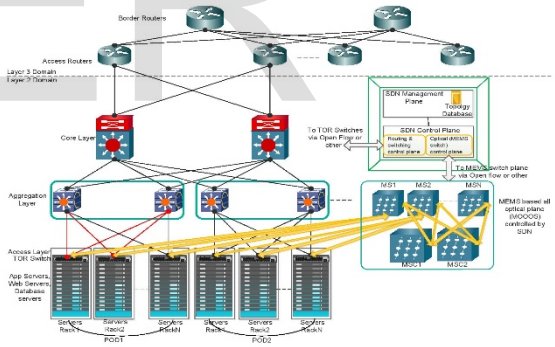


Fig 5.10 Data Flow and Links with Congestion

In above fig 5.10 links having red color shows congestion on active link of packet switched network through which packet trying to traverse. Switches notifies about the congestion and increased latency in the network to its management/controlled plane as shown in fig 5.11

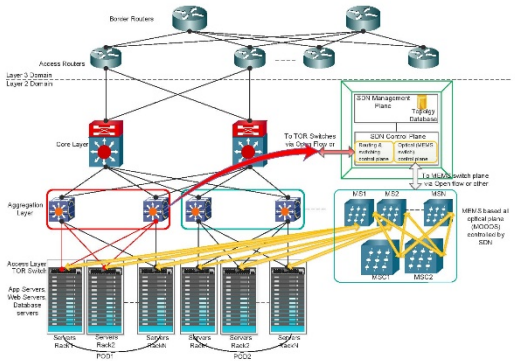


Fig 5.11 Switches Notifying about congestion to Control Plane

The management/ Control plane in response inspect the packets for determining the source and destination of packets flow, consult its topology database and identifies the points b/w which high bandwidth data paths is required to create. As a result Optical control plane notifies the MAOS control plane consisting of MEMS based optical switches that create the high bandwidth data path in parallel with traditional packet switched data paths by electrostatically adjusting and repositioning micro-mirrors which reflects optical beam from the I/P collimator array port to the O/P collimator array port of MEMS switches corresponding to Source & Destination of data flow. Now the elephant data can flow through this direct, less delayed and high bandwidth path hence relieving the network from congestion and improving its latency as shown in fig 5.12.

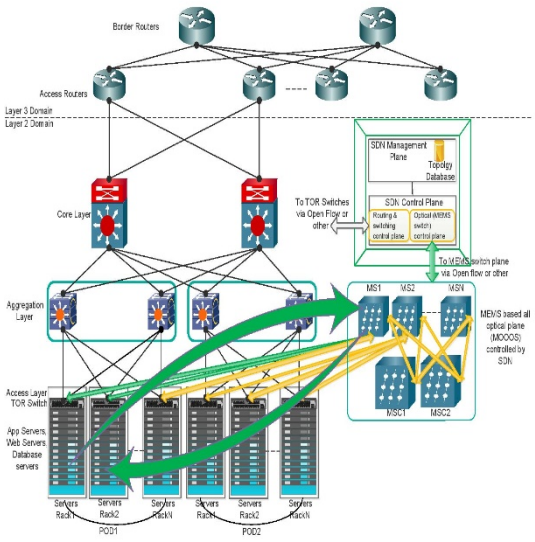


Fig 5.12High data rate optical path through MEMS switches

In this figure traffic is flowing through both packet switched network and also through temporary created high bandwidth path represented by Green colour through MEMS based switches. When the high persistent flow ends up and traffic flow again come to its normal level then control plane tear down this temporary establised optical path and reallocated again when ever and where ever is need.

Software Implementation

6.1 Reference Network Topology

For software implementaion of my proposal first I considered a reference network topology diagramwhich is given as

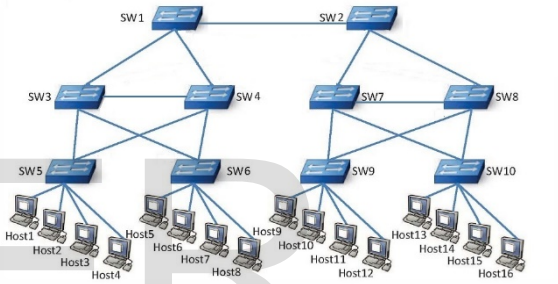


Fig 6.1 Reference network topology for our simulation test

This reference network topology consist of two pods i.e POD 1 and POD2 and each POD suppose to be consist of two access switches, two aggregation switches and eight hosts or servers. POD1 consist of two access (TOR) switches named as SW5 & SW6, two aggregation switches named as SW3 & SW4 and eight hosts named Host1 to Host8 that generates traffic. POD2 also consist of two access (TOR) switches named SW9 & SW10, two aggregation switches named SW7 & SW8 and eight hosts named Host1 to Host8.

6.2 Tools/Software used

The software's which I used to layout the reference network topology and implementing my proposed solution named HFPFMAOS (Hybrid Flow based packet filtering, Forwarding and MEMS based all optical switching) are

- Oracle VM VirtalBox
- Mininet
- ODL controller

- Wireshark
- IPERF

6.3 Preparation of Software

Following steps are required in the preparation of this setup are;

- First downloaded and install Oracle VM Virtual Box that will run my Mininet virtual machine and Open Day Light virtual machine.
- Second make a virtual machine on Virtual Box and download the Mininet virtual machine image and mount it over this newly created virtual machine and install Mininet.
- Third make another virtual machine on Virtual Box and download the ODL controller setup and install it.

To layout and emulate the reference topology I have used the Mininet and its GUI application MiniEdit which is running on one virtual machine on Virtual Box. To access Mininet over SSH and run Miniedit I need X forwarding for which I have used Putty and XMing. For Controlling OpenFlow virtual switches and MEMS switches I have used the external SDN based OpenFlow controller which is Open DayLight (ODL Beryllium). I have used the IPERF to generate data traffic from Hosts/Servers and measuring various performance parameters like link bandwidth, network delays etc. For capturing data packets and checking different protocol messages over various interfaces across the network topology I used Wireshark.

Mininet: It is an open source software emulation tool for creating various custom topologies with number of Hosts/Servers, virtual Switches that can support open flow, interconnecting links and SDN based OpenFlow controller, all running on a single machine. Mininet is a highly flexible Python based network emulator that runs over the Linux operating system. It allows me greater flexibility to create and run various custom topologies, Hosts/Servers to send/receive data packets and network links with any specified bandwidth, and delay etc.

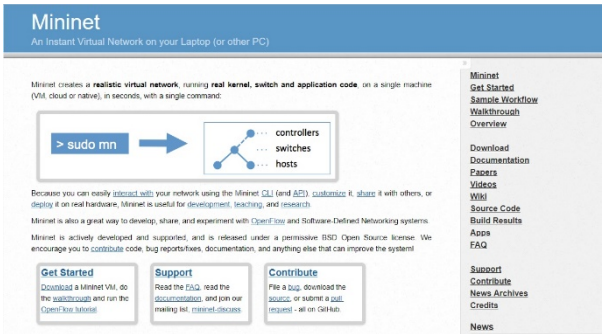


Fig 6.2 Mininet

MiniEdit is an application of Mininet which empowers you to create and run various custom network topologies over Graphical User interface. MiniEdit version which I used for creating my network topology is '2.2.0.1'.

Fig 6.3 MiniEdit

Open DayLight is also an Open source SDN based community project announced in April 2013 which is aimed to propose and promote SDN based North bound API's (Application program interface) and is hosted by LINUX Foundation. It provides flexible, pluggable and modular controller platform that will increase the adoption of Software defined networking techniques. ODL controller is a java based software controller which supports Open Flow and can be deployed on any OS (Operating system) supporting java. It consists of various dynamically pluggable modules to carry out various according to your requirement. ODL is based on web based API named Representational State Transfer (REST) and OSGI (Open services gateway initiative). It consists of separate plug-Inns for supporting different protocols like BGP, OpenFlow 1.0, OpenFlow 1.3, etc. at the south bound interface which are dynamically connected to the SAL (service abstraction Layer) which in turn presents the services of North bound modules to the physical devices underlying in the topology. It is backed by Telecom Industry leading

organizations like, Brocade, Cisco, HP, RedHat, IBM, Juniper, VMware and Microsoft etc. Till now there are six different releases of ODL controller named as Hydrogen, Helium, Lithium, Beryllium, Boron, and Carbon. The first one is Hydrogen which is launched in Feb 2014 and Carbon is the latest which will be launched in June 2017, here in this thesis I am using Beryllium which was launched in Feb 2016.

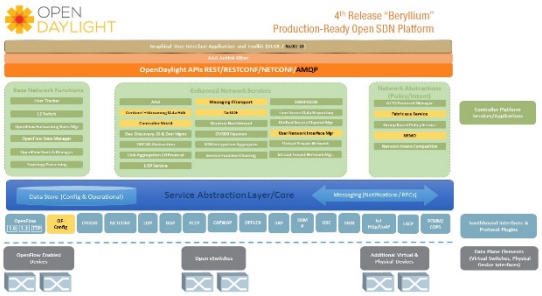


Fig 6.4 Open Day Light

IPERF is a test tool that is used to generate TCP/UDP traffic between the host and to the measure the throughput/bandwidth and quality of the available links carrying the network traffic. The bandwidth can be measured with an IPERF TCP test, whereas the quality of the available links can be measured by pinging from one host to another host and observing latency (round trip time) and by measuring jitter and packet loss from UDP Test. It can be easily installed on any LINUX/UNIX or MS windows OS. For IPERF testing one host should be server and other should be client as shown in fig

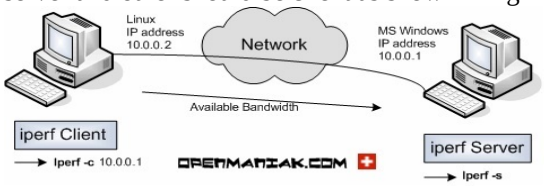


Fig 6.5 IPERF

WireShark is a famous Open Source network analysis tool that is used to capture real time packets for analyzing network protocol and their messages and troubleshooting.

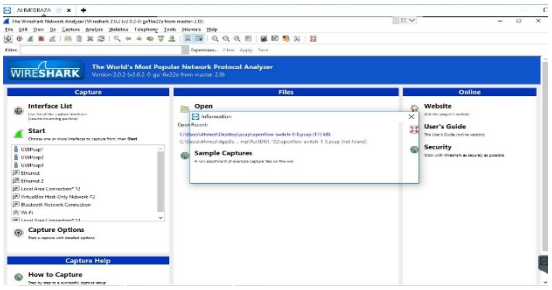


Fig 6.6 WireShark

6.4 Practical Implementation

The practical implementation of HFPFMAOS is done in three steps.

STEP1:Building reference Network topology in Mininet with Traditional switches

i. Download & Install Mininet on one virtual machine in Virtual Box.
ii. Download and install putty.
iii. Start Xming in host operating system for X-forwarding.
iv. Access the Mininet (IP address: 192.168.56.101) via SSH through putty from Host OS.
v. Run the MiniEdit.py file from Mininet.
vi. Buildup the reference network topology in Miniedit by deploying traditional switches.
vii. Run the topology in Miniedit.
viii. For loop avoidance bring down the redundant links between TOR switches and aggregation switches.
ix. Make three Hosts H1, H3 and H4 as UDP streaming servers using IPERF and H11, H13, H16 as UDP streaming clients.
x. Start a ping of 500 Bytes from H13 to H2 to observe network latency.
xi. First Generate traffic Between H1 and H16 for 120sec and check the network performance by observing total data transmitted in fixed intervals of 5sec, total % of packet loss and observe latency of ping.
xii. After 15 sec generate another traffic stream for 120sec Between H3 and H11 in parallel with the first one and check the network performance & network latency.
xiii. After 15 sec Generate third traffic stream for 120sec Between H4 and H6 in parallel with the first one and check the network

performance by observing total data transmitted, % of packet loss for each stream and net observing network latency through ping.

Links	Bandwidth	Delay
TOR	4	10
TOR	10	5
Agrreg	20	3

Table 6.1 Topology Hypothesis

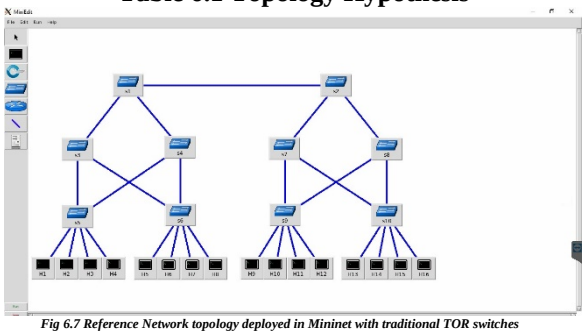


Fig 6.7 Reference Network topology deployed in Mininet with traditional TOR switches

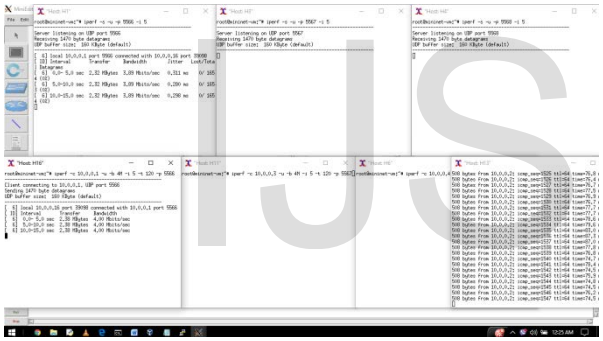


Fig 6.8 UDP traffic generation b/w client H16 and Server H1

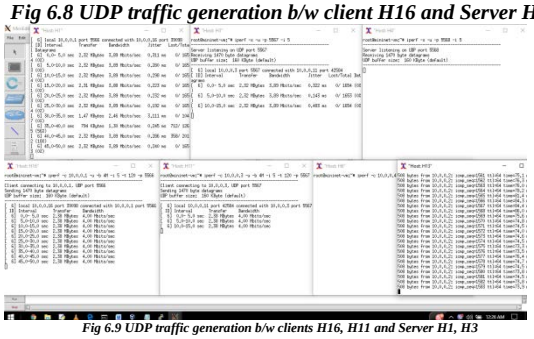


Fig 6.9 UDP traffic generation b/w clients H16, H11 and Server H1, H3

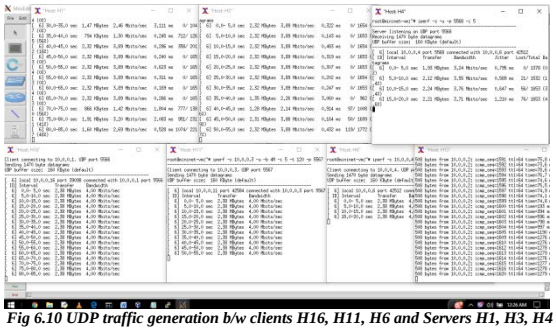


Fig 6.10 UDP traffic generation b/w clients H16, H11, H6 and Servers H1, H3, H4

STEP2: Replace TOR Traditional switches with OF switches

xiv. Download & install SDN ODL controller in second virtual machine in Virtual Box.

xv. Start the ODL controller.

xvi. Replace the TOR traditional Ethernet switches with OF switches in access layer and connect them with the remote ODL controller over the TCL.

xvii. Run the topology in miniedit.

xviii. For loop avoidance bring down the links b/w OF switches and aggregation switches.

xix. Add the Flow entries in the flow table of OF switches for packets filtering, forwarding discovering links and switches connected to OF switches.

xx. Make three Hosts H1, H3 and H4 as UDP streaming servers using IPERF and H11, H13, H16 as UDP streaming clients.

xxi. Start a ping of 500 Bytes from H13 to H2 to observe latency.

xxii. First Generate traffic Between H1 and H16 for 120sec and check the network performance by observing total data transmitted in fixed intervals of 5sec, total % of packet loss and observe latency through ping.

xxiii. After 20 sec generate another traffic stream for 120sec Between H3 and H11 in parallel with the first one and check the network performance & network latency.

xxiv. After 45 sec Generate third traffic stream for 120sec Between H4 and H6 in parallel with the first one and check the network performance by observing total data transmitted, % of packet loss for each stream and net observing network latency through ping.

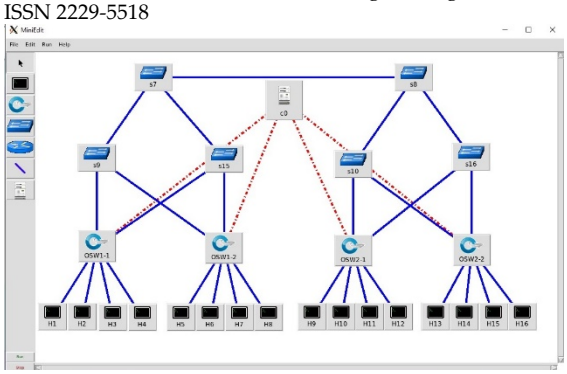


Fig 6.11 Network topology with OF switches in Access as TOR switches

In this topology Host H1~~ H16 are servers installed in their respective racks and connected each connected with OF switches and Traditional switches while C0 is the SDN controller which is ODL having IP address 192.168.56.102 and connected with each OF switch over standard OF port no 6633 via TCP. H1, H2, H3 & H4 are considered as servers installed in rack which is connected with TOR switch(OSW1-1), andH5, H6, H7 & H8 are connected with TOR switch(OSW1-2), whereas H9, H10, H11& H12 are connected with TOR switch(OSW2-1) and H13, H14, H15& H16 are connected with TOR switch(OSW2-2). Each host from H1~~ H16 is assigned an IP address of 10.0.0.1/6 to 10.0.0.16/6 respectively whereas each OF switch is also assigned an IP address which are OSW1-1=10.0.1.1/16, OSW1-2=10.0.1.2/16, OSW2-1=10.0.2.1/16, OSW2-2=10.0.2.2/16. Link between Hosts and OF switches is of 4Mb/s with delay of 10msec, whereas links between OF switches and traditional switches is of 10Mb/s with delay of 5msec and between all traditional switches is of 20Mb/s with delay of 3msec as shown in below figs.

Host, Nodes, Links and interfaces verification can be done by these commands

mininet>nodes
mininet>net
mininet>dump

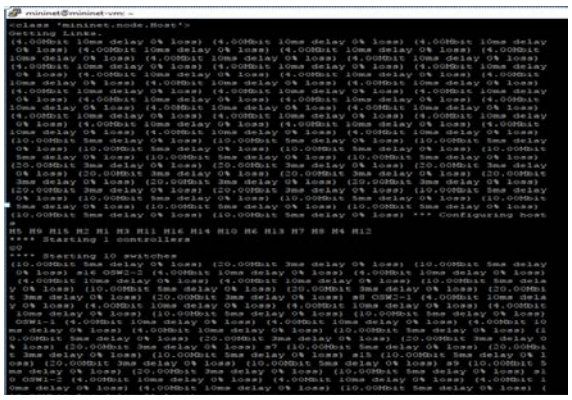


Fig 6.12 Links& Switches created

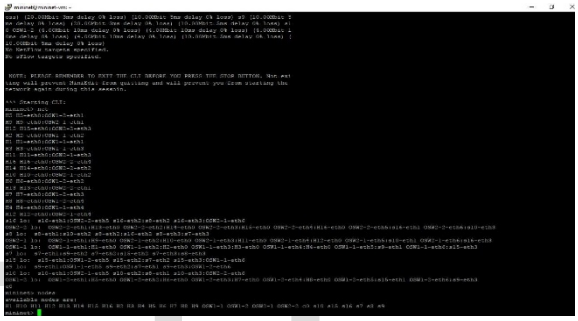


Fig 6.13 Network and Nodes verification

After that I added the flow Entries in the flow table of OF switches OSW1-1, OSW1-2, OSW1-3 & OSW1-4 for my proposed solution HFPFMAOS which are as;

For Host Discovery and ARP Table

mininet>pingall

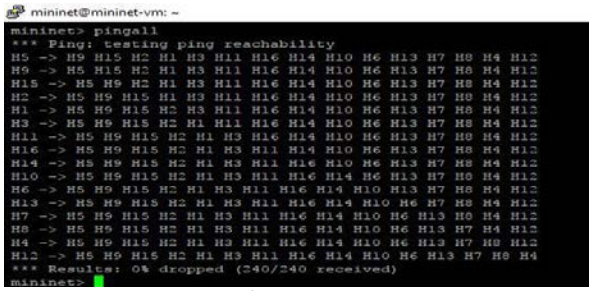


Fig 6.14 Pingall command for All Host Discovery and Reachability

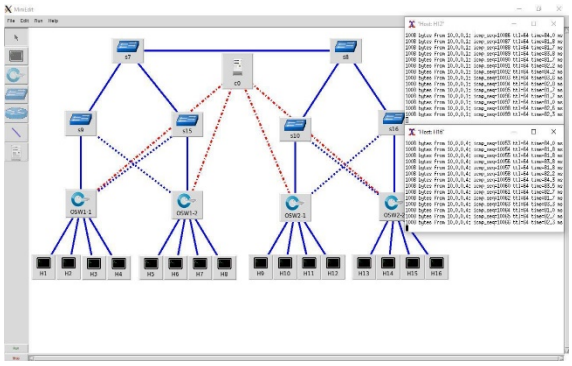


Fig 6.15 Host Reachability test via miniedit

For Links Discovery GARP Requests:

mininet>shovs-ofctl add-flow OSW1-1
dl_type=0x806,nw_proto=1,dl_dst=ff:ff:ff:ff:ff:ff,a
ction=flood
mininet>shovs-ofctl add-flow OSW1-1
dl_type=0x806,nw_proto=1,dl_dst=ff:ff:ff:ff:ff:ff,a
ction=flood
mininet>shovs-ofctl add-flow OSW1-1
dl_type=0x806,nw_proto=1,dl_dst=ff:ff:ff:ff:ff:ff,a
ction=flood
mininet>shovs-ofctl add-flow OSW1-1
dl_type=0x806,nw_proto=1,dl_dst=ff:ff:ff:ff:ff:ff,a
ction=flood

For Links Discovery BDDP Requests:

mininet>shovs-ofctl add-flow OSW1-1
priority=100,dl_type=0x8999,dl_dst=ff:ff:ff:ff:ff:ff
,action=controller:65535
mininet>shovs-ofctl add-flow OSW1-2
priority=100,dl_type=0x8999,dl_dst=ff:ff:ff:ff:ff:ff
,action=controller:65535
mininet>shovs-ofctl add-flow OSW1-3
priority=100,dl_type=0x8999,dl_dst=ff:ff:ff:ff:ff:ff
,action=controller:65535
mininet>shovs-ofctl add-flow OSW1-4
priority=100,dl_type=0x8999,dl_dst=ff:ff:ff:ff:ff:ff
,action=controller:65535

For Enable Layer 2 Forwarding

mininet>sh ovs-ofct1 add-flow OSW1-1
action=normal
mininet>sh ovs-ofct1 add-flow OSW1-2
action=normal
mininet>sh ovs-ofct1 add-flow OSW2-1
action=normal
mininet>sh ovs-ofct1 add-flow OSW2-2
action=normal

ARP Table of Hosts

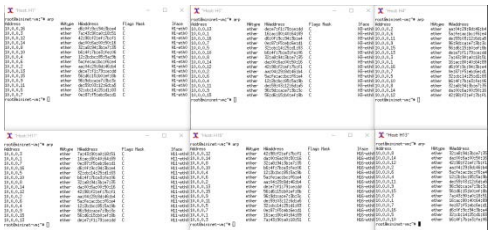


Fig 6.16ARP table entries of Hosts

Wireshark OpenFlow messages b/w OSW & C0

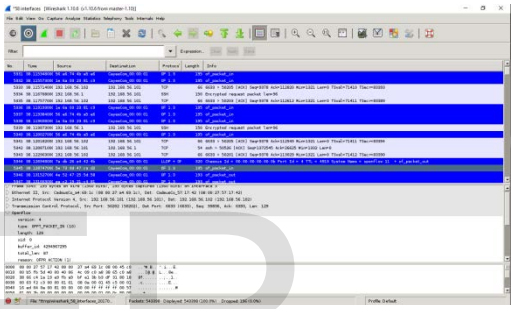


Fig 6.17OpenFlow "Packet_in" message to C0

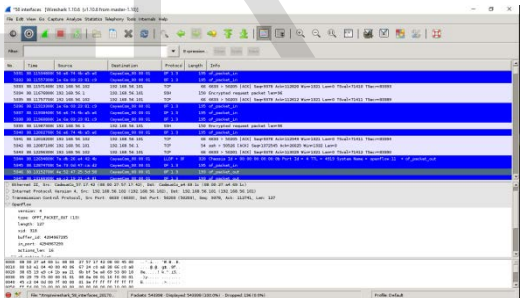


Fig 6.18 OpenFlow "Packet_out" message from C0

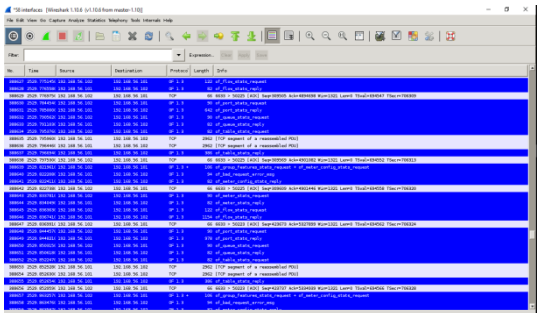


Fig 6.19 OpenFlow messages between OSW & C0

OpenDay Light controller GUI screen Shots:

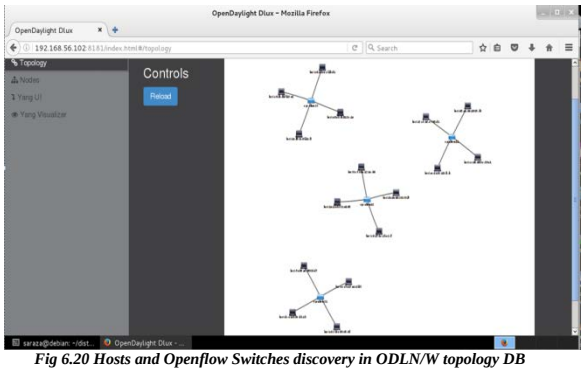


Fig 6.20 Hosts and Openflow Switches discovery in ODLN/W topology DB

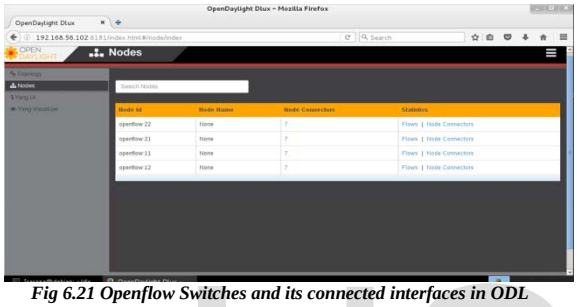


Fig 6.21 Openflow Switches and its connected interfaces in ODL

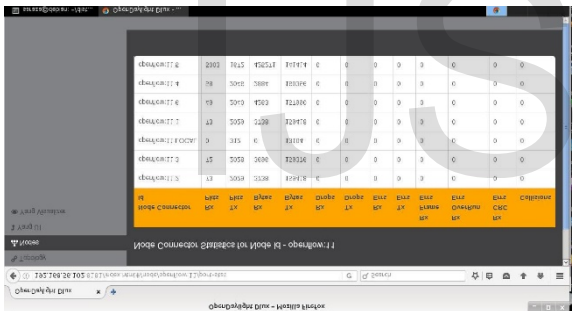


Fig 6.22 OSW11and its connected interfaces

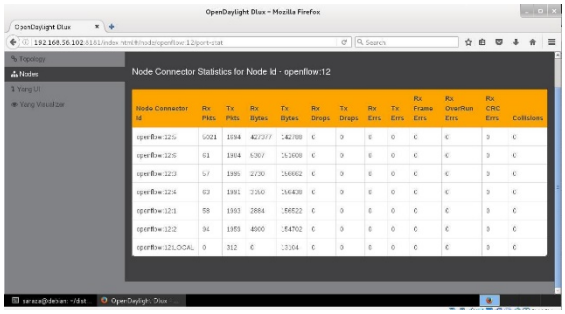


Fig 6.23 OSW12and its connected interfaces

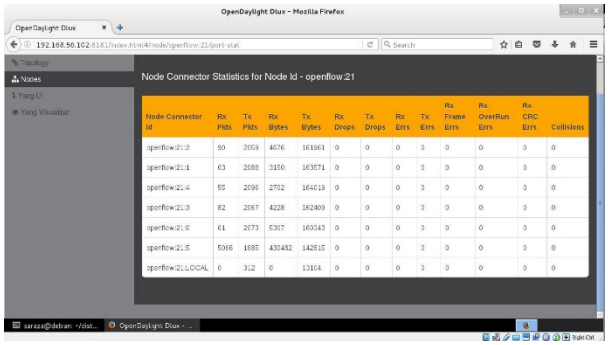


Fig 6.24 OSW21and its connected interfaces

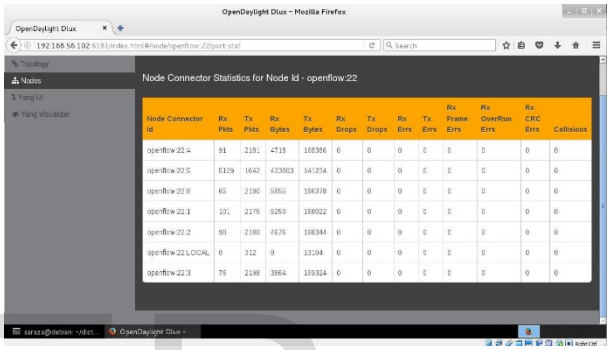


Fig 6.25 OSW22and its connected interfaces

Traffic Generation by Iperf

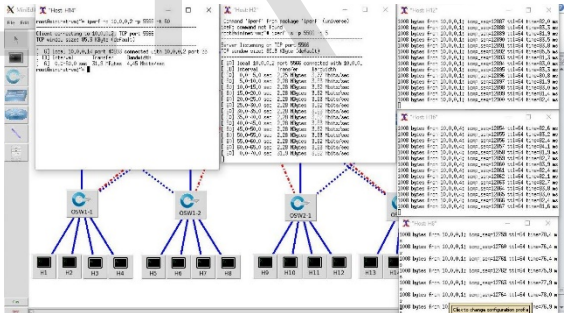


Fig 6.26 Traffic Generation and Latency (Delay)

For traffic generation I made H1, H3 and H4 as IPERF servers and H16, H11 and H6 as their corresponding clients. Traffic generation from clients H16,H11 and H6 is done in three phases and analyze the network topology performance at each step by checking the bandwidth, packet loss and latency through ping of 500Bytes between Hosts H2 and H13.

In first phase 4Mbps UDP data stream is generated for a period of 120sec from clientH16 over a UDP port 5566 which is received on the server H1 over the corresponding port, then I checked the amount

of data transferred data during 5sec interval is 2.32MBytes and percentage of packet loss which is 0% throughout mean while also check the ping delay between H2 and H13 which is around 83ms as can be observed in below figure6.23.

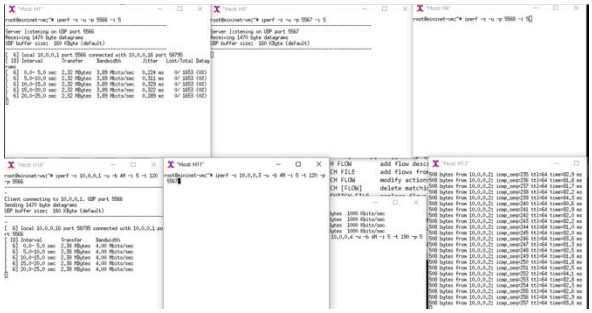


Fig 6.27 UDP traffic generation b/w client H16 and Server H1

In second phase after 20seconds another 4Mbps UDP data stream is generated for a period of 120Sec from client H11 over a UDP port 5567that received on the server H3 over the corresponding port, then I checked the amount of data transferred during 5sec interval and percentage of packet loss for each transfer meanwhile also check the effect on ping delay between H2 and H13 as now total link utilization of the link between OSW11 and S9 is now reached to 80% of available bandwidth. After 20 sec when second data stream is started then H3 server keep receiving 2.32Mbytes with no packet loss while there is a significant reduction in amount of data received by the H1 which is 1.16MB and 1.57MB during 20-30 sec and there is a packet loss as well which is around 26% which is due to buffering limitation, also meanwhile the ping delay also enhanced till 99.7msec from 83msec. But it becomes normal and settles for next interval30-35secwhich can be observed in below figure 6.24.

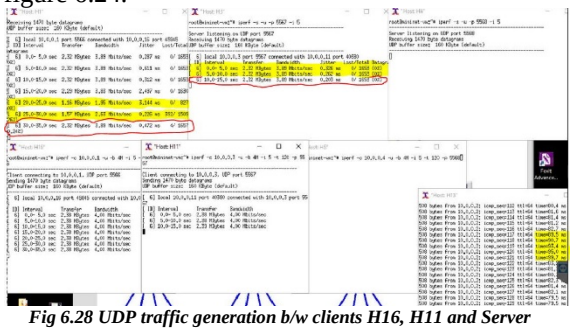


Fig 6.28 UDP traffic generation b/w clients H16, H11 and Server H1,H3

In Third phase again after few seconds another 4Mbps UDP data stream is generated for a period of 120secfrom client H6 over a UDP port 5568that received on the server H4 over the corresponding port. This stream created the congestion over the link between OSW11 and the S9 switch as now the total data transfer rate through this leg is exceed then the total bandwidth. I checked the amount of data transferred during 5sec interval and percentage of packet loss for each transfer and found that there is a significant amount of reduction in total amount of data transferred during 5 sec intervals for each stream and also there is huge percentage of data loss as well in each stream, meanwhile ping delay between H2 and H13 also increases drastically to 1283ms from just 83mswhich can be seen in below figure 6.25.

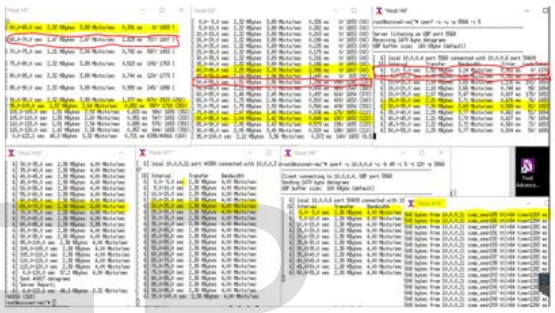


Fig 6.29 UDP traffic generation b/w clients H16, H11, H6 and Server H1, H3, H4

STEP3: Network topology in Mininet with MEMS

In second step to deploy the concept of MEMS switching I add up another Openflow switch named MEMSW1 in aggregation layer along with traditional switches and connect it with OF switches(TOR). To introduce the concept of MEMSI let the links bandwidth with MEMSW1 too high i.e. 1000Mb/s and delay is 1msec. To show all optical switching I make the buffer size and throughput (speedup) of the links equal to bandwidth of the link i.e. 1000Mb/s.
For Step 2 procedure is as;

i. Repeat all procedure from i) to viii) described in step 1 with addition of MEMSW1 and its links during topology creation.
ii. Add the Flow entries in the flow table of OF switches in such a way that normally all data between different legs will traverse through traditional switches and bypass/avoid MEMSW1.

iii. Make three Hosts H1, H3 and H4 as UDP streaming servers using IPERF and H11, H13, H16 as UDP streaming clients.

iv. Start a ping of 500 Bytes from H13 to H2 to observe latency.

v. First Generate traffic Between H1 and H16 for 120sec and check the network performance by observing total data transmitted in fixed intervals of 5sec, total % of packet loss and observe latency through ping.

vi. After 20 sec generate another traffic stream for 120sec Between H3 and H11 in parallel with the first one and check the network performance & network latency.

vii. ODL controller will keep monitoring link utilization of interface b/w OSW11 and S9 when it reaches 80% of its utilization it will pass the next stream through MEMSW1.

viii. After 45 sec Generate third traffic stream for 120sec Between H4 and H6 in parallel with the first one and check the network performance by observing total data transmitted, % of packet loss for first two streams and observing network latency through ping. Moreover also check the total bytes transmitted and latency for third stream passing through MEMSW1.

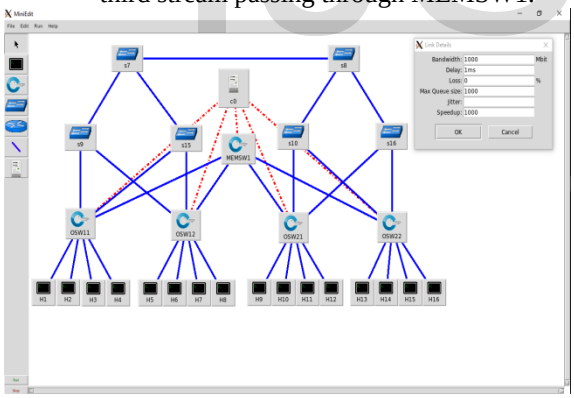


Fig 6.30 Network Topology with MEMS

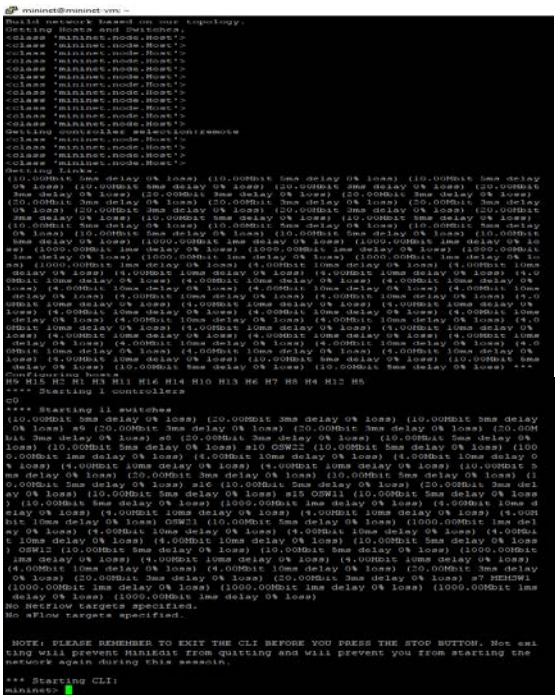


Fig 6.31 Links& Switches created with MEMSW

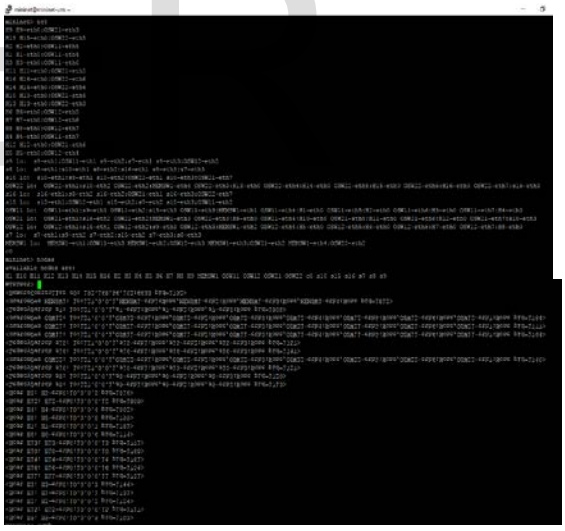


Fig 6.32 Network and Nodes verification

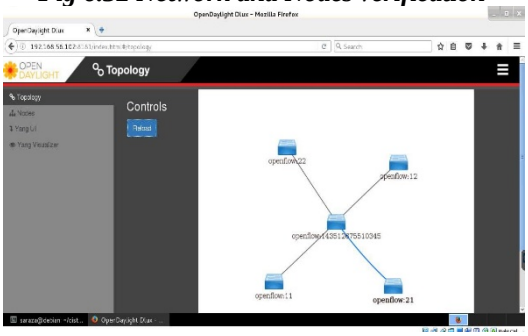


Fig 6.33 ODL Open Flow switches discovery with MEMSW1

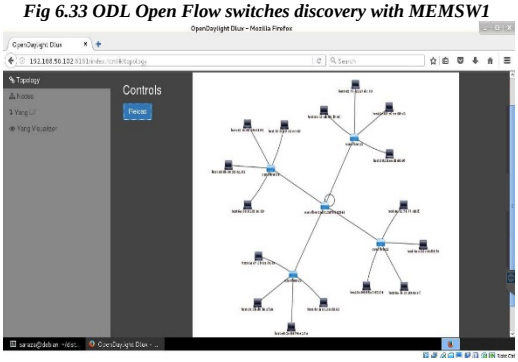


Fig 6.34 ODL Hosts and Open Flow switches discovery with MEMSW1

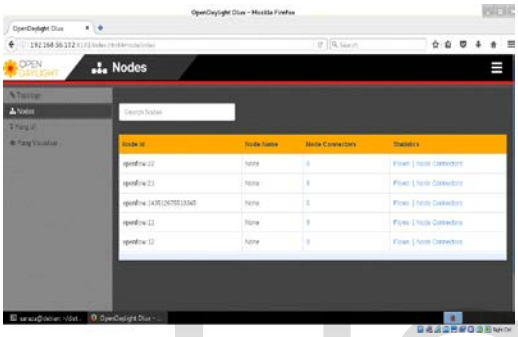


Fig 6.35 ODL OF nodes with no of connected interfaces

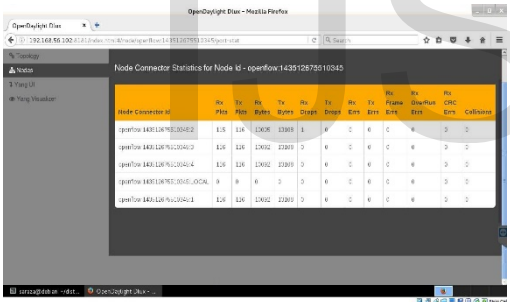


Fig 6.36 ODL MEMSW1 connected interfaces flow statistics

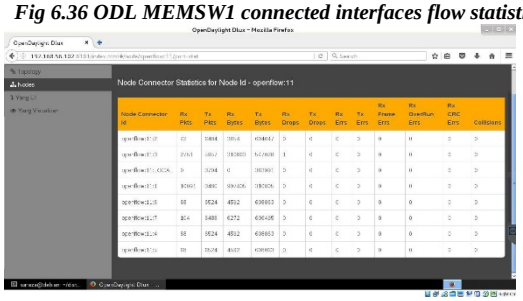


Fig 6.37 ODL OSW11 connected interfaces flow statistics

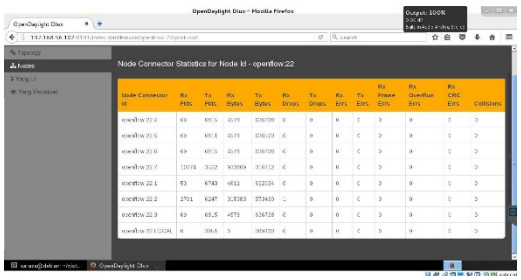


Fig 6.38 ODL OSW22 connected interfaces flow statistics

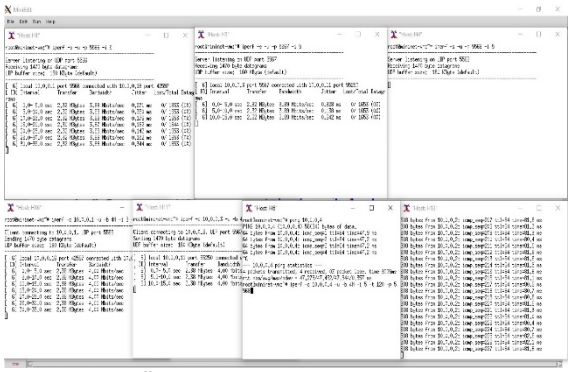


Fig 6.39 UDP traffic b/w H16, H11 & H1, H3 and Ping delay b/w H13 & H2.

In above fig we can observe that total bandwidth utilized by UDP traffic between H16 & H1 and H11& H3 is around 8Mbits/sec which is approximately equal to 80% of total available band width through OSW11. In this fig there is also a ping of 508bytes between H13 & H2 which follows a normal path having a delay of around 82msec while there is another ping between H6 & H4 which is through MEMSW1 having a delay of 47msec which is very less than normal ping delay and approx. 90% of this delay is due to link delay between Host and OSW switches. In above fig first I generated 4Mb/sec UDP stream between H16 & H1 and after 20sec generate another stream between H11 & H3 and observe there is no much effect on delay in ping between H13 & H2 and also there is no packet loss in either stream. As till now the total outgoing link utilization from OSW11 reaches to almost 80% so next stream which I will generate will be between H4 & H6 and is also of 4Mb/sec, so we pass that through MEMS layer (i.e. through MEMSW1) as shown in Fig 6.40.

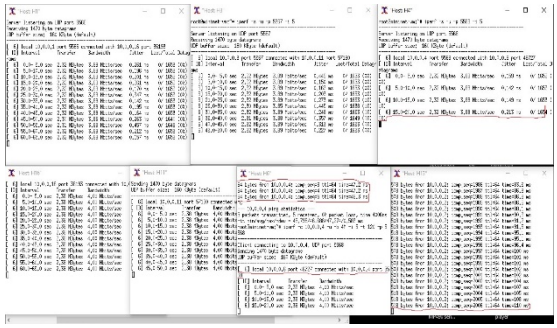


Fig 6.40 UDP traffic through MEMSW1 b/w H4 & H6 and Ping delay.

In above figure 6.40 we can see that the generation of third stream between H4 & H6 has no effect on other streams and there is no packet loss in either stream. As the third stream is passing through MEMS and does not follow the route as first two streams was following.

6.5 Simulation Results Summary, Comparison & Analysis:

The simulation results of all three data forwarding techniques Ethernet Switching, HFPF & HFPFMOAS is summarized in the below Table.

Three Data Forwarding Tecqniques Comparison

Data packets Forwarding Technology/ Topology	Data Stream Sequence	4Mbps UDP traffic generated from Sources (Clients) to Destinations (Servers) with a 2.38MBytes tranferred in 5se							
		H16 (Client) ---> H1 (Server)			H11 (Client) ---> H3 (Server)			H6 (Client) ---> H4 (Ser	
		Bytes Received (5 sec)	Bandwidth	Datagrams Lost	Bytes Received (5 sec)	Bandwidth	Datagrams Lost	Bytes Received (5 sec)	Bandwidth
Ethernet Switching	First stream	2.32MBytes	3.89Mbps	0%	0		0	0	
	Sec stream started	794Kbytes	1.30Mbps	56%	2.32MBytes	3.89Mbps	0%	0	
	After 10 sec Two stream	2.32MBytes	3.89Mbps	0%	2.32MBytes	3.89Mbps	0%	0	
	Third Stream	1.60MBytes	2.69Mbps	40%	2.32MBytes	3.89Mbps	7%	2.21MBytes	3.71Mbps
HFPF	First stream	2.32MBytes	3.89Mbps	0%	0		0	0	
	Sec stream started	1.57Mbytes	2.63Mbps	26%	2.32MBytes	3.89Mbps	0%	0	
	Two streams	2.32MBytes	3.89Mbps	0%	2.32MBytes	3.89Mbps	0%	0	
	Third Stream	1.52MBytes	2.64Mbps	35%	1.52MBytes	2.56Mbps	13%	2.09MBytes	3.5Mbps
HFPFMAOS	First stream	2.32MBytes	3.89Mbps	0%	0		0	0	
	Sec stream started	2.30Kbytes	3.87Mbps	0%	2.32MBytes	3.89Mbps	0%	0	
	Two streams	2.32MBytes	3.89Mbps	0%	2.32MBytes	3.89Mbps	0%	0	
	Third Stream	2.32MBytes	3.89Mbps	0%	2.32MBytes	3.89Mbps	0%	2.32MBytes	3.89Mbps

Table 6.2 Simulations Results summary

From above summary table it can be observed that using traditional switches when second data stream is generated, initially there is a no of datagrams lost and there is dropage in Bytes received as well for few seconds who vanished after. But when third stream is generated the network delay drastically increases due to congestion and there is a continuous dropage of data packets in all streams. However using HFPF there is some improvement in the percentage of lost datagrams and number of bytes received during 5 sec interval when second and Third data stream is generated. But by the addition of MOAS plane with HFPF there is a significant amount of improvement in network delay at each step of data stream generation, also now there is no loss of datagrams as well. Hence this show that HFPFMOAS can be very beneficial for improving network performance and congestion issues in data center communication network.

6.6 Code for Topology Creation:

```
#!/usr/bin/python
from mininet.net import Mininet
frommininet.node import Controller,
RemoteController, OVSController
frommininet.node import CPULimitedHost, Host,
Node
frommininet.node import OVSKernelSwitch,
UserSwitch
frommininet.node import IVSSwitch
frommininet.cli import CLI
from mininet.log import setLogLevel, info
frommininet.link import TCLink, Intf
fromsubprocess import call

defmyNetwork():

net = Mininet( topo=None,
build=False,
ipBase='10.0.0.0/8')

info( '*** Adding controller\n' )
   c0=net.addController(name='c0',
controller=RemoteController,
ip='192.168.56.102',
protocol='tcp',
port=6633)

info( '*** Add switches\n')
   s5 = net.addSwitch('s5', cls=OVSKernelSwitch,
failMode='standalone')
   OSW2-2 = net.addSwitch('OSW2-2',
cls=OVSKernelSwitch)
   OSW1-2 = net.addSwitch('OSW1-2',
cls=OVSKernelSwitch)
   s7 = net.addSwitch('s7', cls=OVSKernelSwitch,
failMode='standalone')
   s16 = net.addSwitch('s16',
cls=OVSKernelSwitch, failMode='standalone')
   s10 = net.addSwitch('s10',
cls=OVSKernelSwitch, failMode='standalone')
   s15 = net.addSwitch('s15',
cls=OVSKernelSwitch, failMode='standalone')
   s9 = net.addSwitch('s9', cls=OVSKernelSwitch,
failMode='standalone')
   s8 = net.addSwitch('s8', cls=OVSKernelSwitch,
failMode='standalone')
   OSW1-1 = net.addSwitch('OSW1-1',
cls=OVSKernelSwitch)
```

```
    OSW2-1 = net.addSwitch('OSW2-1',
cls=OVSKernelSwitch)

info( '*** Add hosts\n')
    H3 = net.addHost('H3', cls=Host, ip='10.0.0.3',
defaultRoute=None)
    H16 = net.addHost('H16', cls=Host,
ip='10.0.0.16', defaultRoute=None)
    H14 = net.addHost('H14', cls=Host,
ip='10.0.0.14', defaultRoute=None)
    H10 = net.addHost('H10', cls=Host,
ip='10.0.0.10', defaultRoute=None)
    H6 = net.addHost('H6', cls=Host, ip='10.0.0.6',
defaultRoute=None)
    H13 = net.addHost('H13', cls=Host,
ip='10.0.0.13', defaultRoute=None)
    H11 = net.addHost('H11', cls=Host,
ip='10.0.0.11', defaultRoute=None)
    H7 = net.addHost('H7', cls=Host, ip='10.0.0.7',
defaultRoute=None)
    H8 = net.addHost('H8', cls=Host, ip='10.0.0.8',
defaultRoute=None)
    H4 = net.addHost('H4', cls=Host, ip='10.0.0.4',
defaultRoute=None)
    H12 = net.addHost('H12', cls=Host,
ip='10.0.0.12', defaultRoute=None)
    H5 = net.addHost('H5', cls=Host, ip='10.0.0.5',
defaultRoute=None)
    H9 = net.addHost('H9', cls=Host, ip='10.0.0.9',
defaultRoute=None)
    H15 = net.addHost('H15', cls=Host,
ip='10.0.0.15', defaultRoute=None)
    H2 = net.addHost('H2', cls=Host, ip='10.0.0.2',
defaultRoute=None)
    H1 = net.addHost('H1', cls=Host, ip='10.0.0.1',
defaultRoute=None)

info( '*** Add links\n')
    OSW1-1H1 = {'bw':4,'delay':'10ms','loss':0}
net.addLink(OSW1-1, H1, cls=TCLink , **OSW1-
1H1)
    OSW1-1H2 = {'bw':4,'delay':'10ms','loss':0}
net.addLink(OSW1-1, H2, cls=TCLink , **OSW1-
1H2)
    OSW1-1H3 = {'bw':4,'delay':'10ms','loss':0}
net.addLink(OSW1-1, H3, cls=TCLink , **OSW1-
1H3)
    OSW1-1H4 = {'bw':4,'delay':'10ms','loss':0}
net.addLink(OSW1-1, H4, cls=TCLink , **OSW1-
1H4)
    OSW1-2H5 = {'bw':4,'delay':'10ms','loss':0}
net.addLink(OSW1-2, H5, cls=TCLink , **OSW1-
2H5)
    OSW1-2H6 = {'bw':4,'delay':'10ms','loss':0}
net.addLink(OSW1-2, H6, cls=TCLink , **OSW1-
2H6)
    OSW1-2H7 = {'bw':4,'delay':'10ms','loss':0}
net.addLink(OSW1-2, H7, cls=TCLink , **OSW1-
2H7)
    OSW1-2H8 = {'bw':4,'delay':'10ms','loss':0}
net.addLink(OSW1-2, H8, cls=TCLink , **OSW1-
2H8)
    OSW2-1h9 = {'bw':4,'delay':'10ms','loss':0}
net.addLink(OSW2-1, h9, cls=TCLink , **OSW2-
1h9)
    OSW2-1h10 = {'bw':4,'delay':'10ms','loss':0}
net.addLink(OSW2-1, h10, cls=TCLink ,
**OSW2-1h10)
    OSW2-1h11 = {'bw':4,'delay':'10ms','loss':0}
net.addLink(OSW2-1, h11, cls=TCLink ,
**OSW2-1h11)
    OSW2-1h12 = {'bw':4,'delay':'10ms','loss':0}
net.addLink(OSW2-1, h12, cls=TCLink ,
**OSW2-1h12)
    OSW2-2h13 = {'bw':4,'delay':'10ms','loss':0}
net.addLink(OSW2-2, h13, cls=TCLink ,
**OSW2-2h13)
    OSW2-2h14 = {'bw':4,'delay':'10ms','loss':0}
net.addLink(OSW2-2, h14, cls=TCLink ,
**OSW2-2h14)
    OSW2-2h15 = {'bw':4,'delay':'10ms','loss':0}
net.addLink(OSW2-2, h15, cls=TCLink ,
**OSW2-2h15)
    OSW2-2h16 = {'bw':4,'delay':'10ms','loss':0}
net.addLink(OSW2-2, h16, cls=TCLink ,
**OSW2-2h16)
s16OSW2-2 = {'bw':10,'delay':'5ms','loss':0}
net.addLink(s16, OSW2-2, cls=TCLink ,
**s16OSW2-2)
s10OSW2-1 = {'bw':10,'delay':'5ms','loss':0}
net.addLink(s10, OSW2-1, cls=TCLink ,
**s10OSW2-1)
s15OSW1-2 = {'bw':10,'delay':'5ms','loss':0}
net.addLink(s15, OSW1-2, cls=TCLink ,
**s15OSW1-2)
s9OSW1-1 = {'bw':10,'delay':'5ms','loss':0}
net.addLink(s9, OSW1-1, cls=TCLink ,
**s9OSW1-1)
    s9s7 = {'bw':20,'delay':'3ms','loss':0}
net.addLink(s9, s7, cls=TCLink , **s9s7)
    s7s15 = {'bw':20,'delay':'3ms','loss':0}
net.addLink(s7, s15, cls=TCLink , **s7s15)
    s8s10 = {'bw':20,'delay':'3ms','loss':0}
net.addLink(s8, s10, cls=TCLink , **s8s10)
    s8s16 = {'bw':20,'delay':'3ms','loss':0}
net.addLink(s8, s16, cls=TCLink , **s8s16)
```

```
    s7s8 = {'bw':20,'delay':'3ms','loss':0}
net.addLink(s7, s8, cls=TCLink , **s7s8)
    OSW1-1s15 = {'bw':10,'delay':'5ms','loss':0}
net.addLink(OSW1-1, s15, cls=TCLink ,
**OSW1-1s15)
s9OSW1-2 = {'bw':10,'delay':'5ms','loss':0}
net.addLink(s9, OSW1-2, cls=TCLink ,
**s9OSW1-2)
s10OSW2-2 = {'bw':10,'delay':'5ms','loss':0}
net.addLink(s10, OSW2-2, cls=TCLink ,
**s10OSW2-2)
    OSW2-1s16 = {'bw':10,'delay':'5ms','loss':0}
net.addLink(OSW2-1, s16, cls=TCLink ,
**OSW2-1s16)
    OSW1-1s5 =
{'bw':10000,'delay':'1ms','loss':0,'max_queue_size':
10000,'speedup':10000}
net.addLink(OSW1-1, s5, cls=TCLink , **OSW1-
1s5)
    OSW1-2s5 =
{'bw':10000,'delay':'1ms','loss':0,'max_queue_size':
10000,'speedup':10000}
net.addLink(OSW1-2, s5, cls=TCLink , **OSW1-
2s5)
s5OSW2-1 =
{'bw':10000,'delay':'1ms','loss':0,'max_queue_size':
10000,'speedup':10000}
net.addLink(s5, OSW2-1, cls=TCLink ,
**s5OSW2-1)
    OSW2-2s5 =
{'bw':10000,'delay':'1ms','loss':0,'max_queue_size':
10000,'speedup':10000}
net.addLink(OSW2-2, s5, cls=TCLink , **OSW2-
2s5)

info( '*** Starting network\n')
net.build()
info( '*** Starting controllers\n')
for controller in net.controllers:
controller.start()

info( '*** Starting switches\n')
net.get('s5').start([])
net.get('OSW2-2').start([c0])
net.get('OSW1-2').start([c0])
net.get('s7').start([])
net.get('s16').start([])
net.get('s10').start([])
net.get('s15').start([])
net.get('s9').start([])
net.get('s8').start([])
net.get('OSW1-1').start([c0])
net.get('OSW2-1').start([c0])

info( '*** Post configure switches and hosts\n')
    OSW2-2.cmd('ifconfig OSW2-2 10.1.2-2')
    OSW1-2.cmd('ifconfig OSW1-2 10.1.1.2/16')
    OSW1-1.cmd('ifconfig OSW1-1 10.1.1.1/16')
    OSW2-1.cmd('ifconfig OSW2-1 10.1.2.1/16')
print "Dumping host connections"
dumpNodeConnections(net.hosts)
print "Testing network connectivity"

    #defperfTest():

    # if user test argument is active then pick the
correct test

net.pingAll()
net.pingAll()
print("Test bandwidth b/w H1 and H2....." );
    #H1, H2 = net.get('H1', 'H2')
    #net.iperf((H1, H2)

    #print ("Test bandwidth b/w H1 and
H6....." );
    H1, H6 = net.get('H1', 'H6')
net.iperf((H1, H6))

    #print( "Test bandwidth b/w H1 and
H10....." );
 H1, H10 = net.get('H1', 'H10')
net.iperf((H1, H10))

    #print( "Test bandwidth b/w H1 and
H15....." );
    H1, H15 = net.get('H1', 'H15')
net.iperf((H1, H15))

    # also argument for generating traffic

    # arugment for stat analysis

CLI(net)
net.stop()

if __name__ == '__main__':
setLogLevel( 'info' )
myNetwork()
```

Conclusion and Future Work

7.1 Conclusion

In this research work, I successfully implemented my proposed solution named HFPFMAOS (Hybrid

Flow based packet filtering, Forwarding and MEMS based all optical switching) using OpenFlow tools like Mininet, OpenDayLight SDN controller and IPERF. This solution is very helpful inefficiently utilizing data center network resources by managing East-West data center traffic and supporting a variety of traffic patterns (i.e. mouse flows and Elephant flows) which will eventually reduce congestion and improves network latency in the traditional data center communication network. This solution modifies the traditional data center network architecture in such a way that it will resolve most of its challenges with little modification in low CAPEX keeping its current infrastructure. It leverages the concept of SDN by employing OF switches in Access and MEMS in aggregation which enables you to control East-West data flows between servers and create dynamically very high bandwidth data paths between TOR OF switches (Access layer) to avoid congestion, ease traffic flows and maintain QOS throughout its network. As SDN is an emerging technology which is becoming popular very rapidly and adoptable due to its flexibility and great potential and this solution also employees it so it can be very helpful to meet current and future challenges which traditional or current data centers are facing. Secondly this solution employee MEM switching which is already matured and deployed in optical backbone transport network equipment's like DWDM for switching of extremely high data rates, so using MEMS switches at aggregation layer which can create dynamically very high bandwidth optical paths of any data rate between any TOR switches which can solve many issues like scalability issue. So this solution with a blend of MEM switching and SDN can be very beneficial in meeting current and future requirements of data center communication network.

7.2 Future Enhancement

As a future enhancement we can employee Multi-pathing and dynamic load balancing over its links between OF switches and traditional switches as here for loop avoidance I intentionally make the sec link down. Secondly we can test this idea by implementing it on other data center topology architecture and check its performance for the deployment of new data centers build up. Thirdly we can employ automatic switching of data flows utilizing high bandwidth over the MEMS path or of new flows in case the utilization of available bandwidth reached to 80% of total bandwidth.

References

[1] Cisco Global Cloud Index: https://cisco.com/c/en/us/solutions/collateral/ service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.pdf

[2] https://www.cisco.com/c/dam/m/en_us/service-provider/ciscoknowledgenetwork/files/547_11_10-15-DocumentsCisco_GCI_Deck_2014-2019_for_CKN_10NOV2015 _.pdf

[3] eBook: Cisco Data-center-virtualization-fundamentals-understanding-9781587143243

[4] http://www.graybar.com/applications/data-centers/types

[5] http://www.buusinessn ewsdaily.com/4982-cloud-vs-data-center.html

[6] http://www.cyberciti.biz/faq/data-center-standard-overview/

[7] http://www.graybar.com/applications/data-centers/tiers

[8] https://www.microsoft.com/en-us/research/publication/the-cost-of-a-cloud-research-problems-in-data-center-networks/.

[9] Data center Load balancing data center services: https://learningnetwork.cisco.com/docs/DOC-3438

[10] http://research.microsoft.com/en-us/um/people/dmaltz/papers/DC-Costs-CCR-editorial.pdf

[11] M. Al-Fares, A. Loukissas, and A. Vahdat, Fat Tree: "A scalable, commodity datacenter network architecture," in ACM SIGCOMM.

[12] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: a high performance, server-centric network architecture for modular data centers," in ACM SIGCOMM.

[13] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "DCell: A scalable and fault-tolerant network structure for data centers," in ACM SIGCOMM.

[14] "VL2: A Scalable and Flexible Data Center Network" https://pdfs.semanticscholar.org/74a2/0105bca9430c41fd67cec635445b734961eb.pdf

[15] "3D MEMS Optical switch with toroidal concave mirror" https://www.ntt-review.jp

[16] "High-yield Fabrication Methods for MEMS Tilt Mirror Array for Optical Switches" by Joji Yamaguchi †, Tomomi Sakata, NobuhiroShimoyama, https://www.ntt-review.jp

[17] "State of the Art of Optical Switching Technology for All-Optical Networks" http://home.deib.polimi.it/

[18] "Comparison between Optical Switching Technologies" http://mapyourtech.com

[19] https://www.sdxcentral.com/wp-content/uploads/2015/11/2015_SDxCentral_-SDN_Controllers-Report_Cisco_FINAL.pdf

[20] http://www.opendaylight.org/project/technical-overview

[21] "HP OpenFlow Protocol Overview" http://www1.hpe.com

[22] http://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-59/161-sdn.html

[23] "Software-Defined Networking: State of the Art and Research Challenges" https://arxiv.org/ftp/arxiv/papers/1406/1406.0124.pdf

[24] "Current Trends of Discovery Topology in OpenFlow-based Software Defined Networks" by L Ochoa Aday - 2015 http://upcommons.upc.edu/

[25] "Efficient Topology Discovery in Software Defined Networks" https://www.research gate.net/

[26] http://networkengineering.stackexchange.com/questions/7713/how-does-gratuitous-arp-work.

[27] "Gratuitous ARP" https://learningnetwork.cisco.com

[28] https://live.paloaltonetworks.com/t5/Management-Articles/Trigger-a-Gratuitous-ARP-GARP-from-a-Palo-Alto-Networks-Device/ta-p/61962

[29] Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures http://www.internetsociety.org/sites/default/files/10_4_2.pdf

[30] "Calculate Bandwidth Utilization" http://www.cisco.com/

[31] "MEMS Micro electromechanical systems" "http://internetofthingsagenda.techtarget.com/definition

[32] https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/OpenFlow OF/OpenFlow OF-switch-v1.3.1.pdf

[33] https://www.mininet.org