Survey of SDN Traffic Engineering (Load balancing and Energy Saving)

Introduction

The increase of network resources and users have put a lot of burden on the current network infrastructure causing congestion and affecting the network performance. Congestion occurs when the resources are insufficient or when the traffic is not efficiently utilized, causing overutilization on some links while underutilization on others. The solution would have been to increase the capacity of the network by adding more resources and expanding the infrastructure, but this is not easy and practical as it comes with high cost. The easier and immediate solution would be to reduce congestion by using balanced traffic distribution schemes [2].

IP Networks use IGP (Interior Gateway Protocol) protocols such as OSPF/ISIS which aggregate traffic to same destination through same path. They do not focus on distributing the load on the network, causing scenarios where even in the presence of underutilized paths there is congestion in the network. Multi-Protocol Label Switching (MPLS) was introduced primarily to solve this problem by supporting Traffic Engineering, which allowed it to move traffic from congested path to a less congested path. However, MPLS adds a lot of complexity and overhead in the network.

On the other hand, power consumption has been on a steady rise with the advancement in technology. Datacenters are required to provide sufficient capacity in terms of bandwidth and redundancy to meet the requirements of today's applications and users.
The switches and links in a datacenter are mostly active and consuming energy, they are mostly underutilized, some estimates state that only 30-40% of network devices and links are utilized most of the time. Even though the utilization of the resources varies with time the power consumption is fixed and independent to traffic variations. This necessitates the need of considering methods of saving energy.

Software Defined Network (SDN), is a new network paradigm in which the control and data plane are separated, and forwarding decision is made centrally by a logical controller. The controller can perform complex programming and compute the best path and forward to the switches, which simply act as forwarding devices.
Using the Controller, SDN can easily identify traffic and provide fine grained Traffic engineering dynamically without added complexity and it can be used to deploy energy aware routing algorithms in an efficient and dynamic manner [17].

In this project, we will survey some Traffic Engineering (figure 1) and Energy Saving Techniques in datacenter environment. Section 1, provides some of the IP based Traffic Engineering mechanisms available for the classic network architecture and discusses their challenges. Section 2, describes an overview of SDN architecture and discuss various elephant flow and mice flow load balancing techniques. Section 3, describes various Energy saving techniques, with focus on Minimizing and Maximizing MLU (Minimum Link Utilization). Finally, Section 4 concludes our project.

1 IP Based Traffic Engineering

Traffic Engineering was introduced in ATM to solve the congestion problem caused by the increase in demand for multimedia service such as voice, data and video, but IP quickly took over due to the simplicity and introduction of IP-QOS [1].

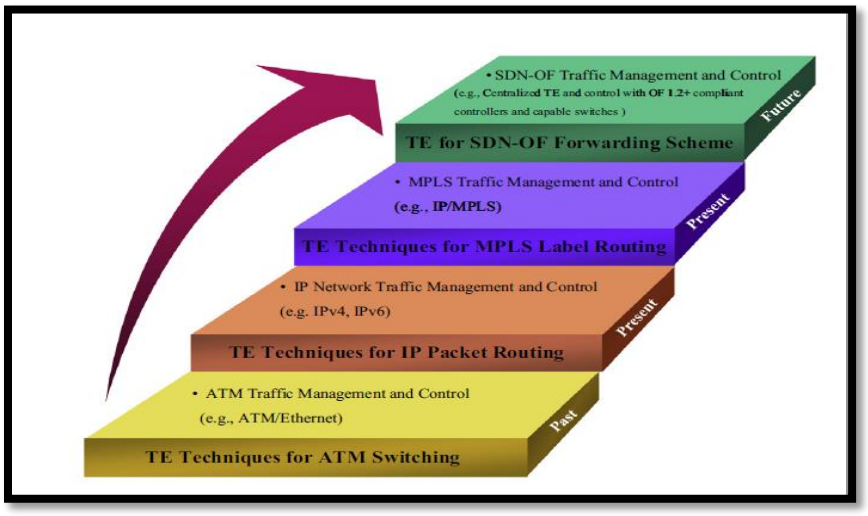

Figure 1: Evolution of traffic engineering [1]

Traffic engineering is very important in today's IP networks and is required to optimize the network performance and traffic delivery by making sure that the traffic is distributed evenly in the network without overloading or overutilizing any of the links capacity and ensuring spare capacity for future demands. In IP networks QOS and failure protection is also considered part of Traffic engineering thus the TE solution must consider how to provide QOS and better protection from failures. So far, the available IP based Traffic engineering solutions use shortest path algorithm for routing based on Link weight and equal cost load balancing which splits traffic evenly on multiple paths with the same cost [1].

IP Networks use Interior Gateway Protocols (IGP) like OSPF and ISIS which use shortest path algorithm to compute traffic without consideration for other factors like available bandwidth and delay etc. They therefore aggregate traffic to same destination through same path causing congestion on that link despite existence of underutilized links as seen in figure 2. To optimize the utilization of resources in the IP network we will thus require Traffic Engineering to be able to control how traffic flows through the network to optimize resource utilization and network Performance and minimize congestion [2].

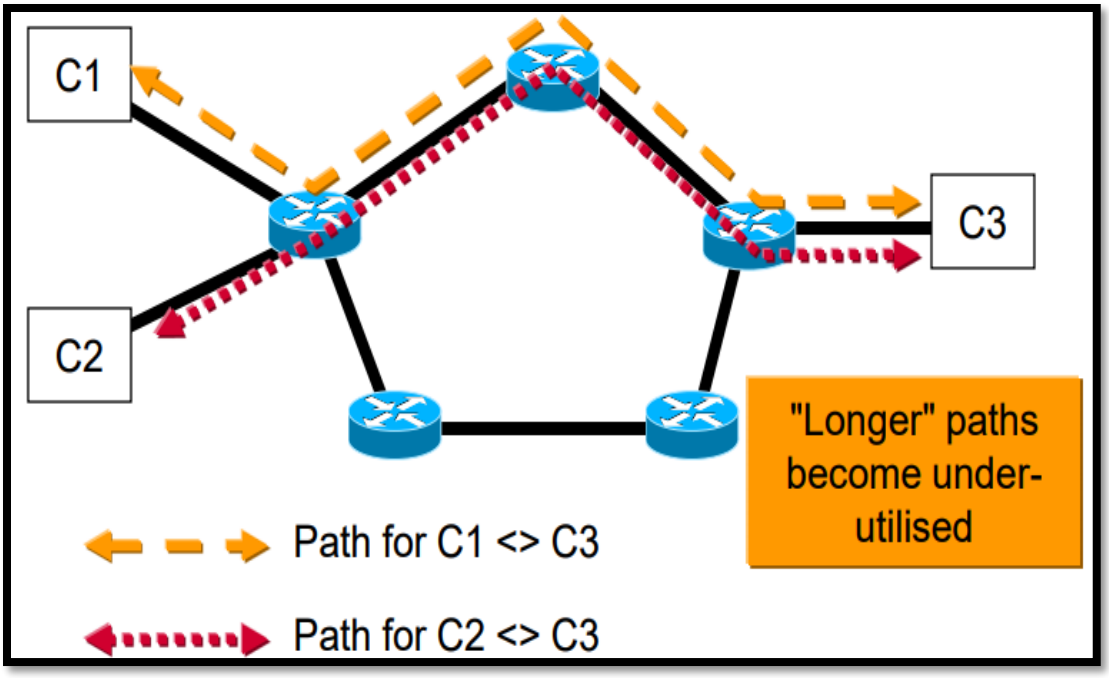

Figure 2: Fish Eye Problem caused by IGP [5]

1.1 IGP Link Weight for Traffic Engineering

Normally IGP protocols use link weights to select the shortest paths, these weights are exchanged between all IGP routers to provide a complete view of the network, which are then pushed into the routing table for routing traffic [3]. If the value of the link weight is calculated such that it is resilient to traffic fluctuations particularly while taking a global view of the network, we can achieve near optimal traffic engineering [3]

However, the link weight would need some sort of intervention to be modified either by an operator or program, changing the link weight of a single link can easily affect the entire network so care needs to be taken before its changed, and it is not ideal to change the link weight multiple times, to avoid network disruption [3]. Thus, selecting good link weights requires having an accurate and timely view of the current network topology and traffic, this information can be obtained from SNMP traps, logs, databases, billing information, the customers downloads, routing tables and interface status as well. **IP link weight for Traffic Engineering lacks the flexibility required to support modern day traffic engineering [1][3].**

1.2 The equal-cost multi-path routing

Today's IP network are vast in size and may contain several hundreds of routers switches and hosts, they are connected together and use IGP protocols to communicate, the IGP protocols may be broken down into multiple areas to provide better and easier management of the network, these networks may have multiple shortest paths to each other, The IGPs are not specifically designed with distributing traffic between multiple paths, and they lack the means to distribute the traffic arbitrarily bases on specific requirements [1].

One of the most common IGP protocols, OSPF provides shortest- path-first routing with simple load balancing by Equal-Cost Multi-Path (ECMP).
ECMP is primarily based on the Hash function, which works to divide the hash space into equal-size partitions corresponding to the outbound paths, and forwards the packets based on their destination information [1]. It splits the traffic evenly across all the available next hops along the set of shortest paths to achieve fair load balancing. However, even if the traffic is equally distributed, it may not always achieve optimal load balancing.

Consider the scenario in Figure 3 below, Available bandwidth between switches is 1.5Gbps, if B is sending a 0.5Gbps of data to destination D. When A tries to send 1Gbps of data to the same destination D at the same time, it has two paths, by using ECMP the traffic from A to D will be equally spread across the two paths (0.5Gbps up and 0.5Gbps down). However, since B is already sending 0.5Gbps traffic to D, the common links will be utilized at 66% of their capacity [4].

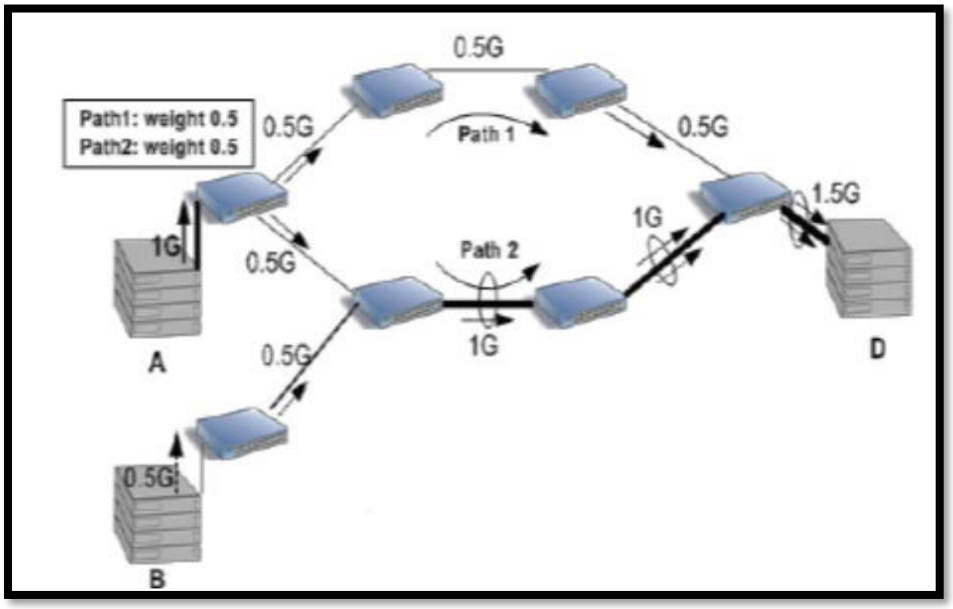

Figure 3: Traffic Load distribution using ECMP [4]

1.3 MPLS-based traffic engineering

To overcome the limitations of IP, Multi-Protocol Label Switching (MPLS) was introduced, its primary objective is to provide Traffic Engineering, Unlike IGP protocols where the routing decision is made by the protocols, MPLS allows for explicit routing of traffic between a source and destination and it allows for splitting of network traffic in an arbitrary way (unequal cost load balancing). Furthermore, it also controls traffic flow in a network and can move traffic away from the shortest path fast(SPF) calculated by IGP to a less congested path.
It does this by setting up Labeled switching paths (LPS) and controls how the LSP's are recovered in case of a failure [1].

In MPLS data forwarding is based on locally significant 20-bit labels rather than the IP address, which are assigned and distributed between MPLS routers using Label distribution protocol(LDP) [5]. A packet entering the MPLS cloud is assigned a MPLS label by the ingress router(Push), and the label is swapped as it moves from one router to the other within the cloud, and the packet is either removed at the penultimate hop or the egress router(pop) and then forwarded back as an IP packet into the IP network.

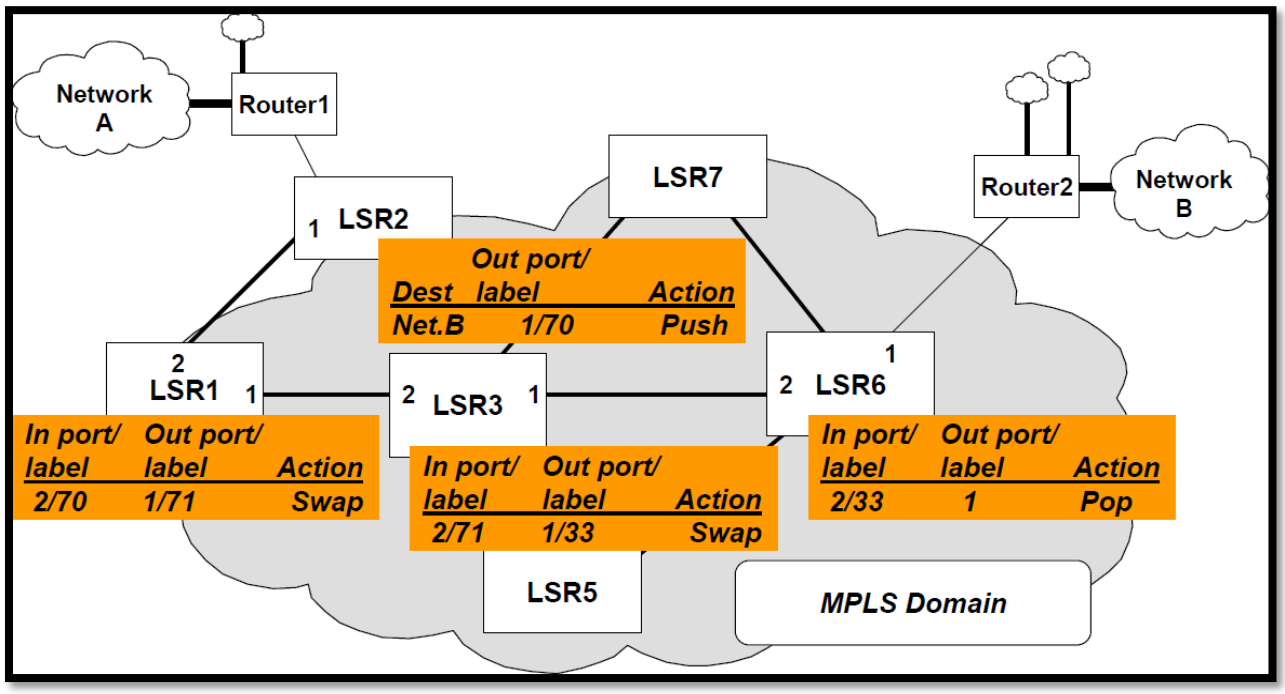

Figure 4: MPLS Label operation [5]

For Traffic Engineering path calculation, MPLS uses IGP enhancements such as OSPF-TE protocols to distribute relevant information such as available bandwidth, attribute flags and administrative weight, bases on these constraints MPLS calculates the constraint based shortest path (CSPF). After which it uses the signaling protocol RSVP-TE to setup, maintain and teardown the explicit paths called Label Switching paths (LSP) [5].

In MPLS the paths that are created are called LSP Tunnels, and are maintained by RSVP protocols, for traffic to be forwarded down these tunnels, the network could use three methods such as: static routing, policy-based routing or autoroute. Since the main purpose for MPLS is TE, unequal cost load sharing can be performed between multiple tunnels, to do this we need to first define the sharing ratio and add it to the forwarding table. QOS information is carried in the IP packets, but since MPLS does not read the IP packets, it must carry QoS information in the Experimental-LSP(E-LSP) or Label-LSP(L-LSP) to be able to support QOS. MPLS can combine DiffServ and Traffic engineering to provide Traffic engineering with QOS, this is called DiffServ aware traffic engineering [5].

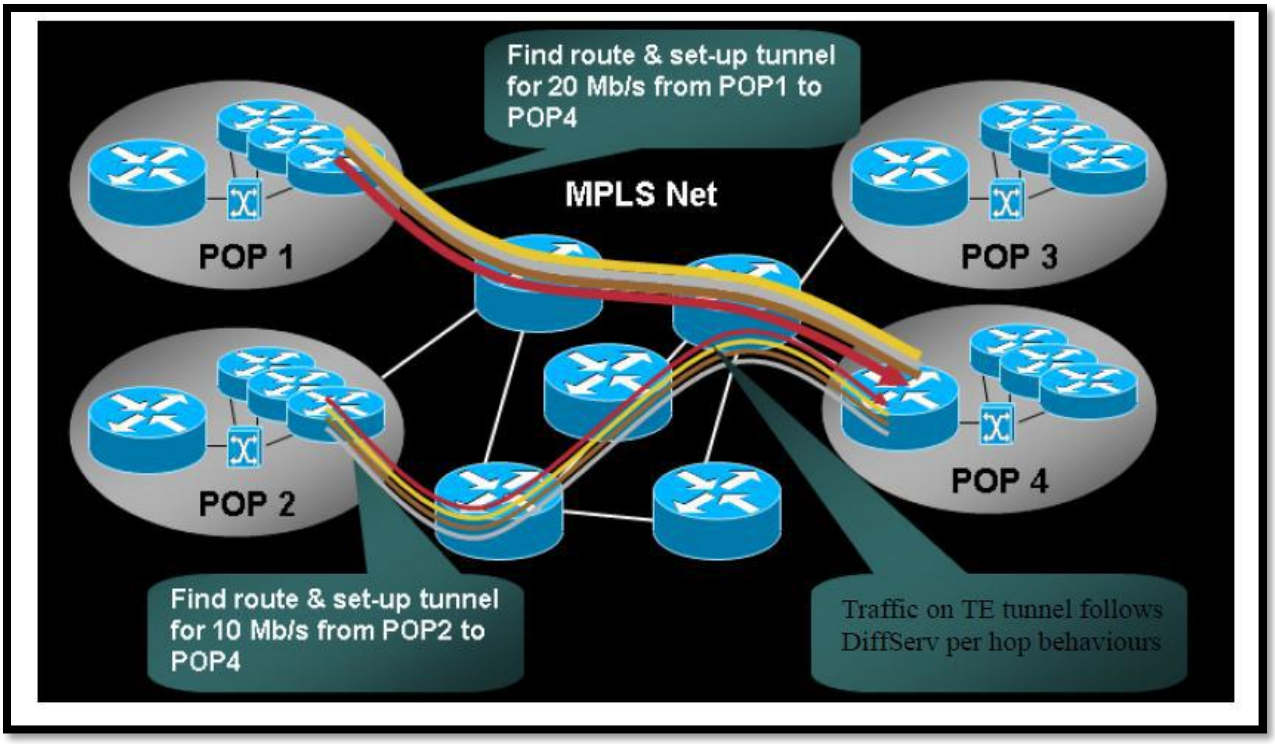

Figure 5: DiffServ-aware Traffic Engineering using MPLS [5]

However, **there is scalability and robustness issue of MPLS-TE**, If the tunnel metrics are changed before SPF is completed we could have undesirable results. Also, as the aggregate traffic is forwarded through dedicated LSPs, **we could have an incredibly large number of LSP's which would add lots of overhead**. In case of link failures if there is no backup path It can take a long time for MPLS detect/notify and then recalculate and create an alternate path, thus necessitating the need of protections [1][5]. **The addition of the various elements that MPLS-TE requires to work such as; OSPF-TE, RSVP-TE, LSP, FRR make it very complex and add lots of overhead.**

The networking industry is growing at a very fast pace, and lots of new innovations are taking place, one such technology that is more commonly deployed is cloud computing, companies and organizations want to be able to host and access their resources instantaneously with ease and low cost, there is a tremendous demand for state of the art datacenters to handle this, datacenters also need more intelligent and efficient network management systems like Software Defined Networks (SDN).**SDN is able to provide a fine-grained, adaptive traffic management as opposed to fixed approaches like Equal-Cost Multi-Path (ECMP).**

2 Traffic Load Balancing in SDN (Datacenter Environment)

In SDN as seen in figure 6, the control plane and data plane are separated, thus the network traffic is divided into control layer traffic and data layer traffic, in addition to this the forwarding decision calculation in SDN are centralized rather than distributed, because of this SDN can more comprehensively consider multiple optional link utilization rates and flow characteristics to perform superior load balancing when compared to traditional IP/MPLS networks [6].

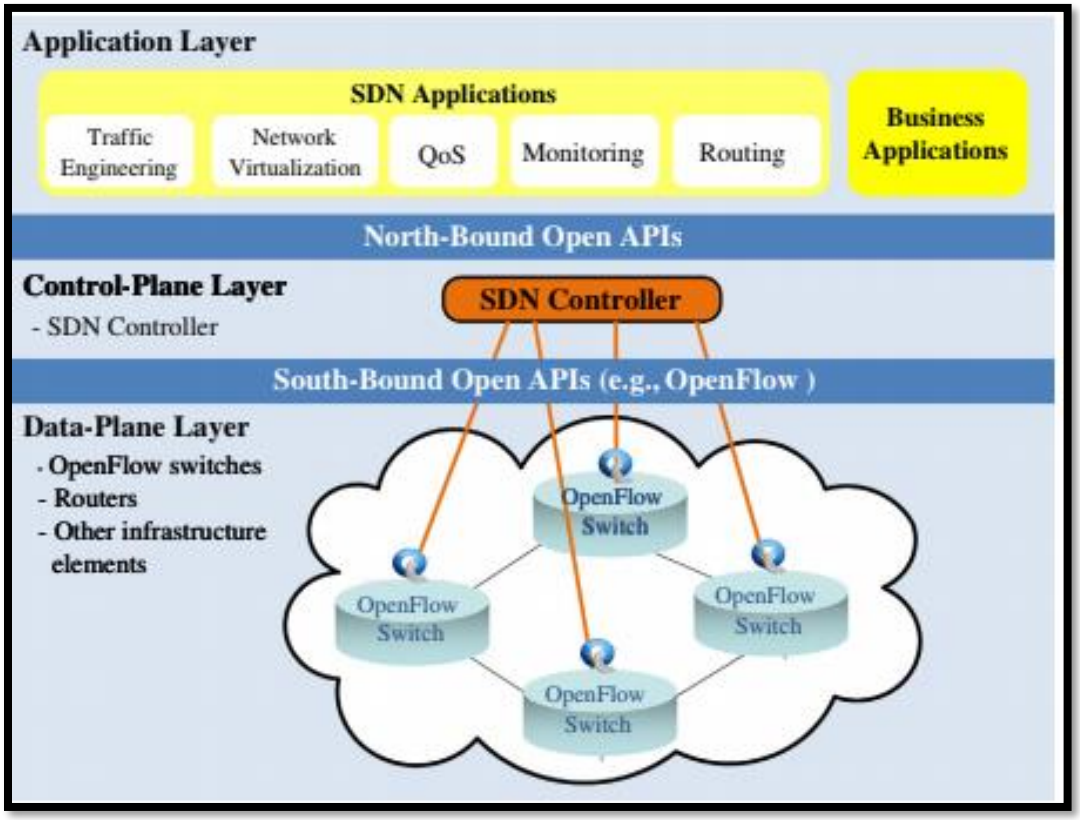

Figure 6: SDN Architecture [1]

Several protocols have been proposed for load balancing, one of those protocols discussed earlier in details is Equal cost multipath (ECMP) which uses hash algorithm to achieve load balancing.

The problem with ECMP is that it attempts to load balance by randomly splitting the flows across equal cost paths using flow hashing. But as we know flows vary in size and duration, due to the lack of consideration of size and duration of the flow, **ECMP may often end up causing hot-spots in the network**. In figure 7, a few long-lived flows (elephant flows) end up being hashed to the same path, this will inevitably affect the network performance even though there is spare capacity elsewhere in the network [11].

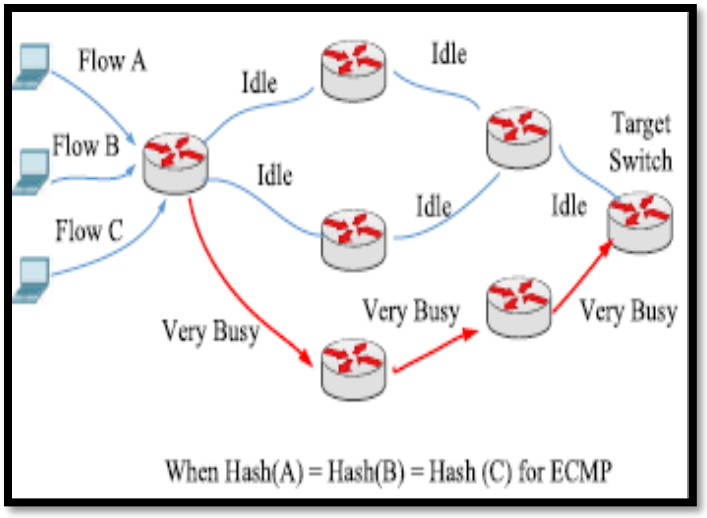

Figure 7: Load Imbalance of ECMP [6]

2.1 Flow based load balancing (Elephant Flow)

In datacenters, traffic flows are separated in two parts, either short-lived (mice flows) or throughput bound (elephant flows) [4]. Most SDN techniques propose solving this problem by identifying the elephant flows, and the controller focuses on them for making forwarding decision to achieve near optimal load balancing of network traffic [6]. Since the controller has a global view of the network and knows the state of all the links in the network it is better positioned to make traffic engineering decisions that can help improve the network throughput, achieve better bandwidth utilization and ease congestion as well [11]. Some of these techniques are Hedera, Mahout, MicroTE and DevoFlow.

a. **Hedera** is a SDN Traffic management system that manages network traffic by effectively utilizing the available bandwidth in a datacenter [6]. It focuses on the elephant flows to better distribute the load in a datacenter network [7]. Elephant flows are detected at the edge switches by polling them every five seconds, if it detects a flow rate that is greater than a specific threshold, i.e. 10% of network interface controller(NIC) it will be marked as an elephant flow. After elephant flows are detected, it will calculate the bandwidth requirement of those flows and together with a global view of the load condition at that time it will calculate a reasonable path for those flows in such a way that they are non-conflicting and distributed properly across the available paths, thus improving bandwidth utilization [7]. **In hedera the constant polling occurs every 5 seconds and due to that there is high overhead and latency**.

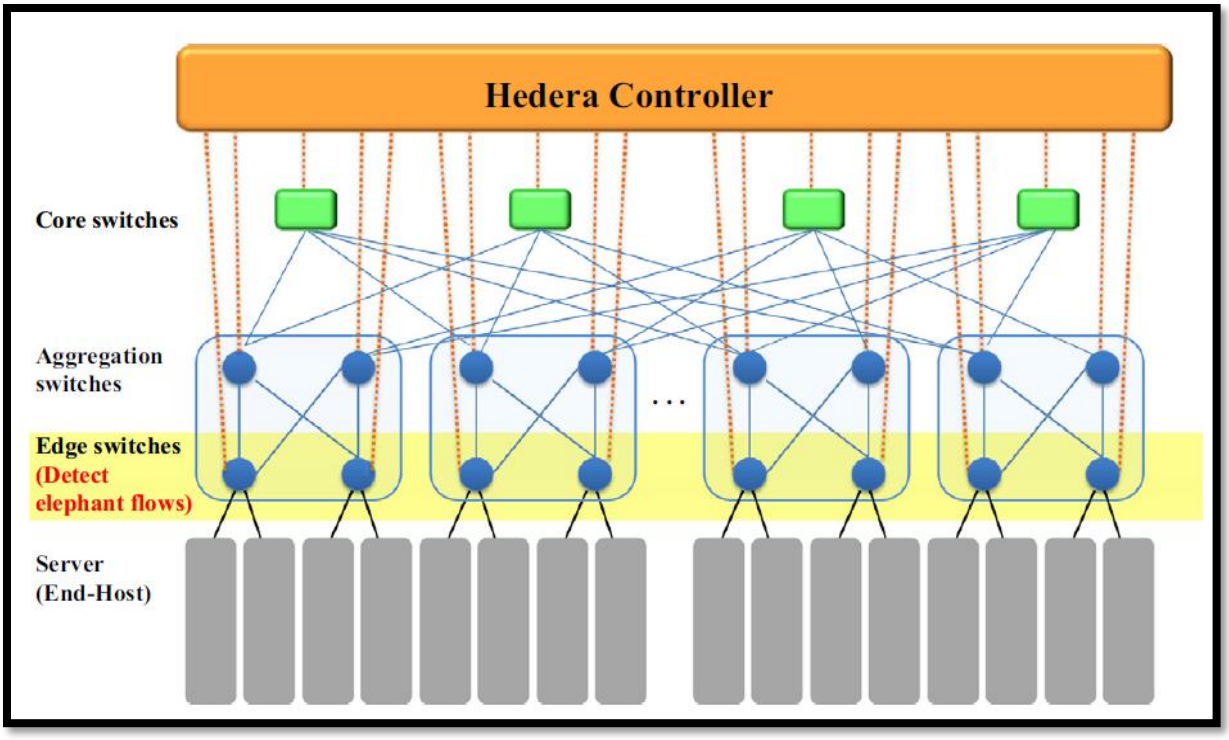

Figure 8: Hedera architecture [1]

b. **Mahout** traffic management system attempts to overcome the overhead and latency problem in Hedera. It uses a separate backend server, instead of the edge switches which helps to detect the elephant flows at the end-hosts. It uses a shim layer in the operating system to mark the significant elephant flows when the buffer overruns a given rate threshold which is normally 100KB for Mahout. Once an elephant flow is detected, the controller calculates the best path using flow statistics and link utilization and updates the corresponding rules in the flow table. Regular network traffic will be handled by using ECMP. The main advantage of the separate server is that mahout can detect elephant flows faster and eliminating the issues of hedera **however it requires end-host modification to detect the flows** [8].

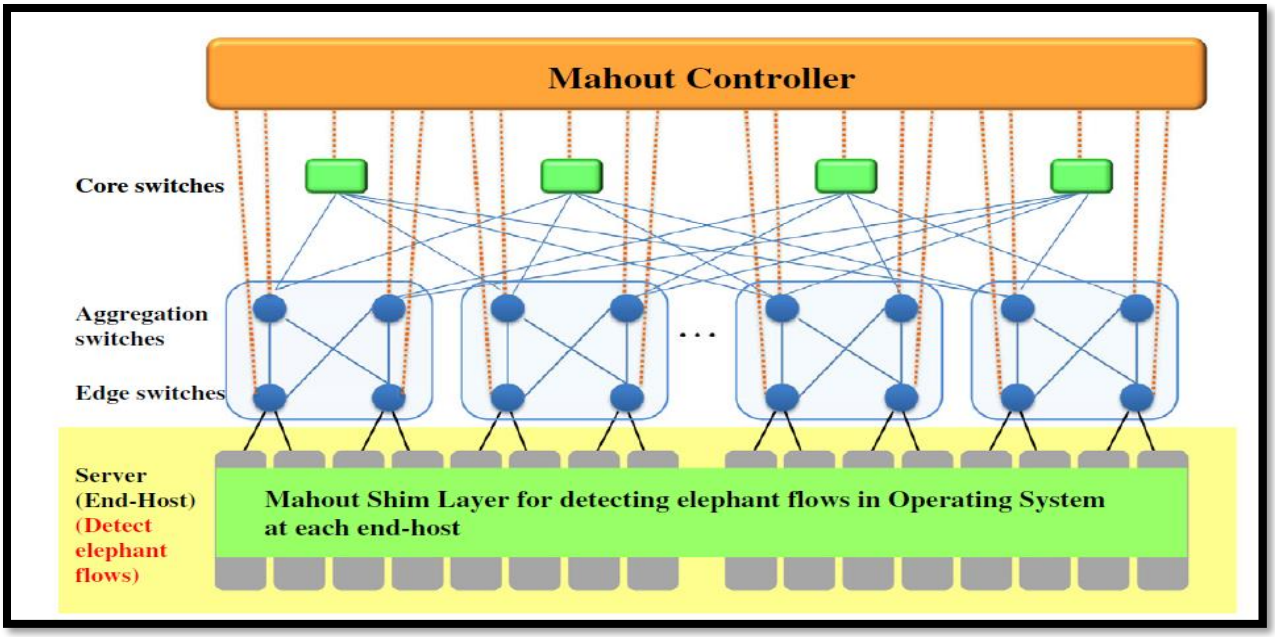

Figure 9: Mahout Architecture [1]

c. **MicroTE,** like Mahout detect the elephant flows at the end-host, but it assumes that datacenter traffic is bursty in nature and unpredictable over long time, and focuses on short-term prediction, to quickly adapt to the changes in traffic pattern to achieve fine grained traffic engineering [9]. When the flow status has clearly changed, it triggers flow aggregation behavior and based on the difference in size of average and instantaneous flow rate, it determines if the flow is an elephant flow. It then calculates best path to route the traffic based on a global view of the network on multiple paths or deals with it using a heuristic ECMP algorithm if it's not an elephant flow [9]. **It requires host modification and lacks the scalability due to short execution cycle.**

d. **DevoFlow** also detects elephant flows at the edge switches, it uses packet sampling to detect elephant flow, by monitoring the switch statistics, if the flow exceeds a given threshold i.e 1-10Mb it will be marked as an elephant flow. DevoFlow uses wildcards rules to handle micro flows which the switches use to route microflows locally without involving the controller, while the controller manages the overall control of the network and routes elephant flows. [6] [10]. In this way, DevoFlow can reduce the load on the centralized controller allowing it to perform other necessary function, thus improving the overall network scalability and performance [6].

2.2 Problem of Flow based Load Balancing (Elephant Flow)

Statistics show that in a datacenter elephant flows only account for 10% of the entire flows, but they carry more than 80% of the entire traffic volume of a datacenter [11].
Thus, it's obvious that aiming to better distribute the elephant flows will significant ease congestion, therefore the techniques discussed above focus on the elephant flows and used other baseline routing techniques like ECMP to manage the rest of the datacenter flows (mice flows) [11].
Once the threshold is reached the flow is marked as an elephant flow, and the controller then computes the best path for this elephant flow and then pushes this computed path to the switch. But **all these techniques use a flow-based load balancing approach**, thus the elephant flow is limited by the bandwidth of a single physical link, it may still be difficult to achieve equal load on the paths. The path that will deliver the elephant flows is still prone to being oversubscribed while other paths remain underutilized [11]. This would certainly affect the performance of operations like big data transfer.

2.3 Load Balancing- Splitting Elephant Flows

[11] proposes an elephant flow load balancing mechanism which uses a weighted multipath routing algorithm to solve the above problem. This proposal in [11] requires three basic steps, first just as in the earlier schemes it requires the elephant flows to be detected, this is done at the end host to reduce the overhead on the controller, after the elephant flows are identified it will involve splitting the elephant flows. At the same time based on a **weighted routing algorithm** it will involve finding a set of dynamically computed ratios (weights). This dynamically computed ratio will be used to spread the traffic at each hop across the available next hops for given traffic demands. Once the path is computed it is finally installed in the switches to forward the flows/packets.

2.3.1 Elephant Flow Detection

In this approach the elephant flows are detected at the end host similar to some of the earlier techniques, once the flow exceeds a given threshold over a determined time window, it is marked and identified as an elephant flow. After the flow is detected as an elephant a weighted multipath routing algorithm will be used to handle the flow all other flows that don't cross the threshold are called mice flows and will be handled by default ECMP.

2.3.2 Elephant Flow Splitting

In OpenFlow communication protocol, group tables and flow tables are used to support multipath routing. Once an elephant flow is identified it is assigned a group, the group table has a set of group action buckets, and each group bucket contains a set of actions to be

applied to the matching flows. The weight field in the action bucket is used to define the share of traffic process by the group in the bucket as in figure 10.

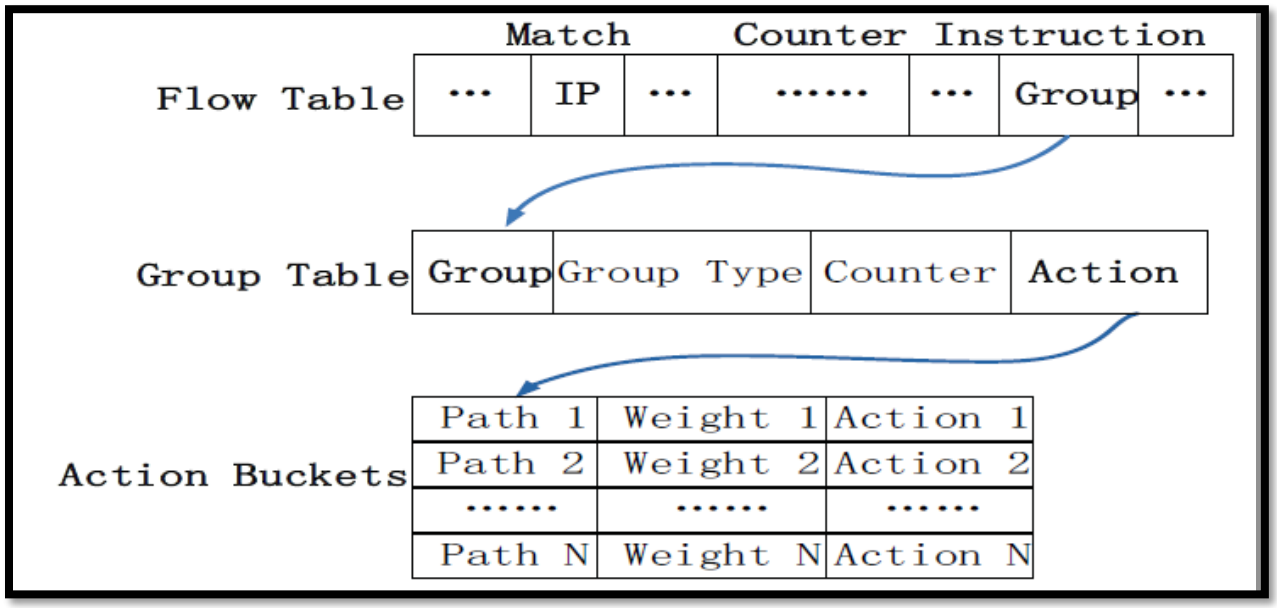

Figure 10: OpenFlow Group Table [11]

2.3.3 Architecture

The proposed architecture has the following modules as seen in figure 11:

- The **Topology module** provides the network topology details
- **Monitoring module** monitors and stores the statistics of the OpenFlow switches in memory as a snapshot object
- The Topology and Monitoring modules send their information to the **Elephant Path computation module**
- The **elephant computation module** uses a weighted multipath routing algorithm to compute the load ratio of candidate paths between any pair of end hosts, and then spreads the elephant flows across multiple links
- The **Path install, and Track module** then installs and tracks the path computed by the elephant path computation module into the OpenFlow switches as flow entries.

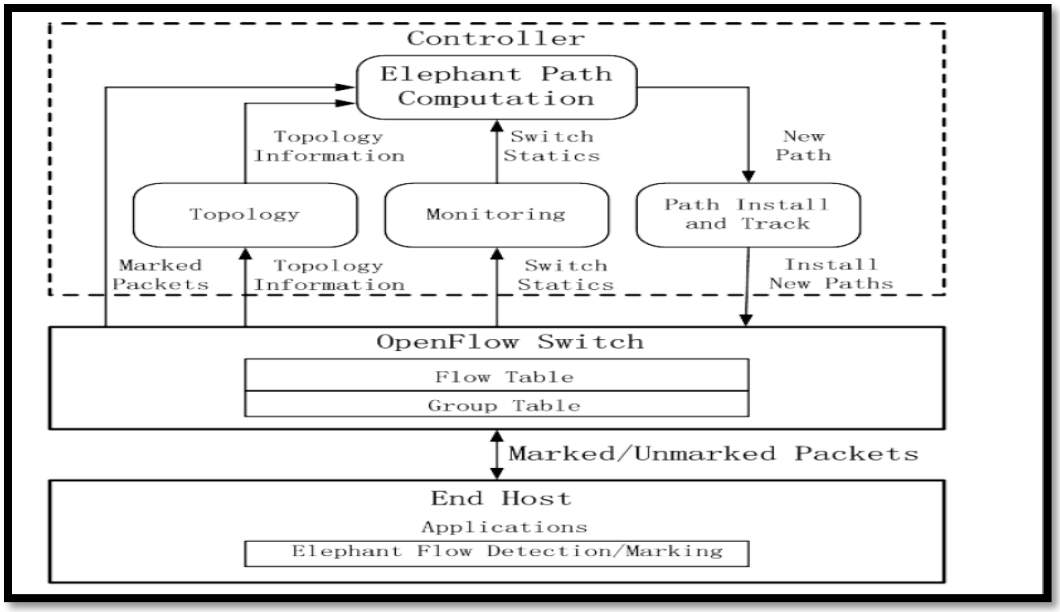

Figure 11: Architecture (Elephant Flow Splitting) [11]

2.3.4 Reordering Avoidance

To avoid the issue of packet reordering, flowlet switching scheme is implemented in the switch. It measures the delay difference Δt between paths, if two consecutive packets appertaining to the same flow whose arrival interval is less than difference Δt, they take the same path, otherwise a new flowlet is beginning thus it might be relocated, if desired. Picking a flowlet timeout larger than the maximum latency of the set of parallel paths, consecutive flowlets can be switched independently with no danger of packet reordering [12].

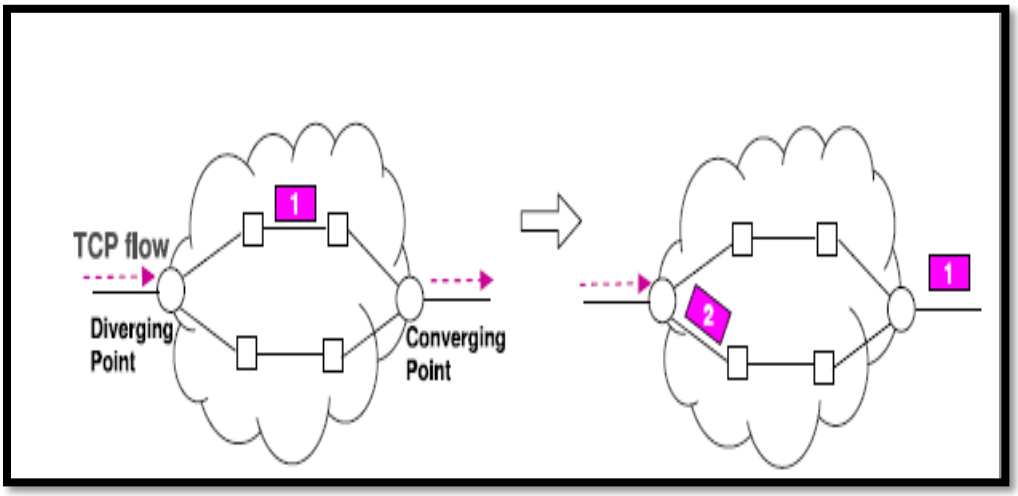

Figure 12: Reordering Avoidance [12]

From evaluations as seen in figure 13,14 and 15, Weighted Multipath Routing scheme considered in [11] has **low latency and significant higher throughput as well as much higher utilization as compared to Single Path Routing and ECMP**.

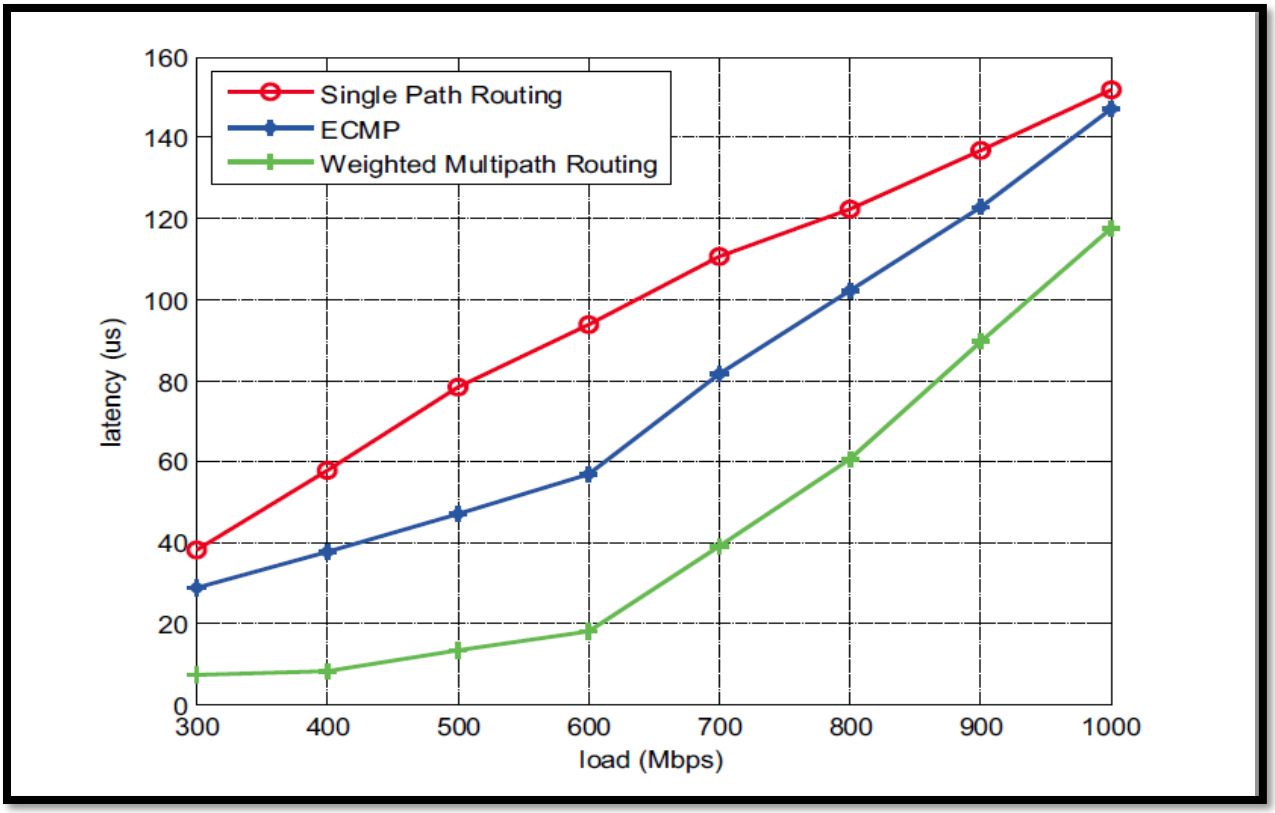

Figure 13: Network latency of the weight multipath routing VS Single path routing, ECMP [11]

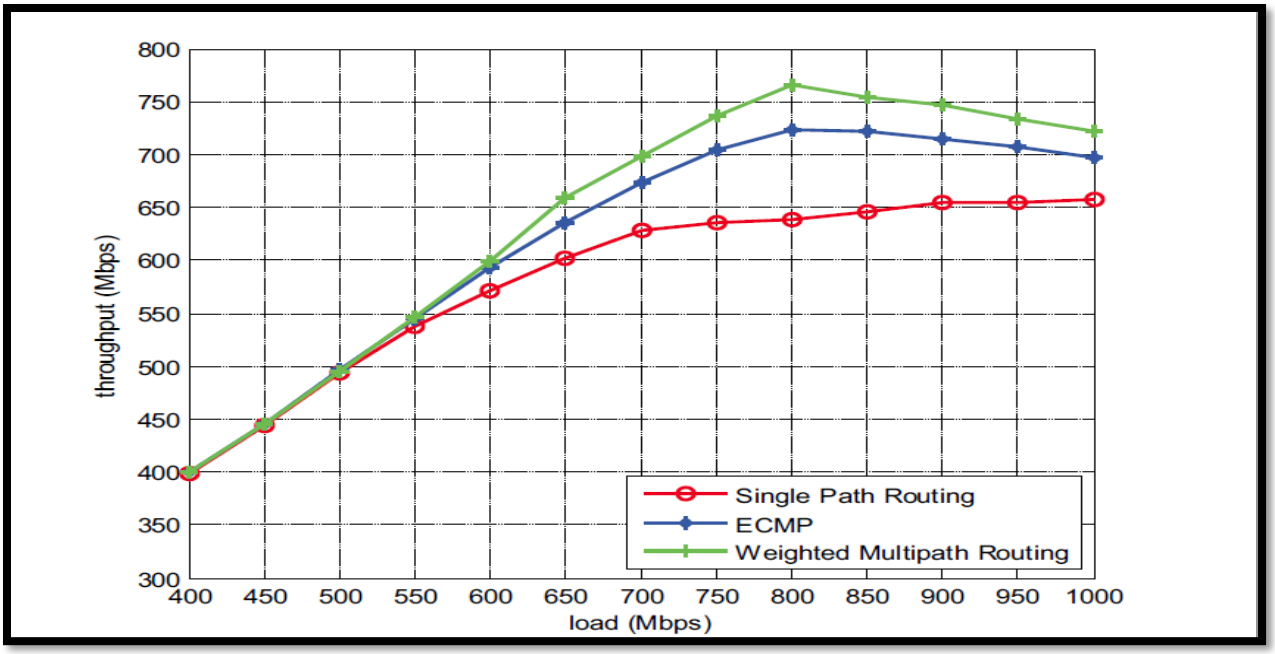

Figure 14: Throughput performance of the weighted Multipath routing versus Single Path Routing, ECMP [11]

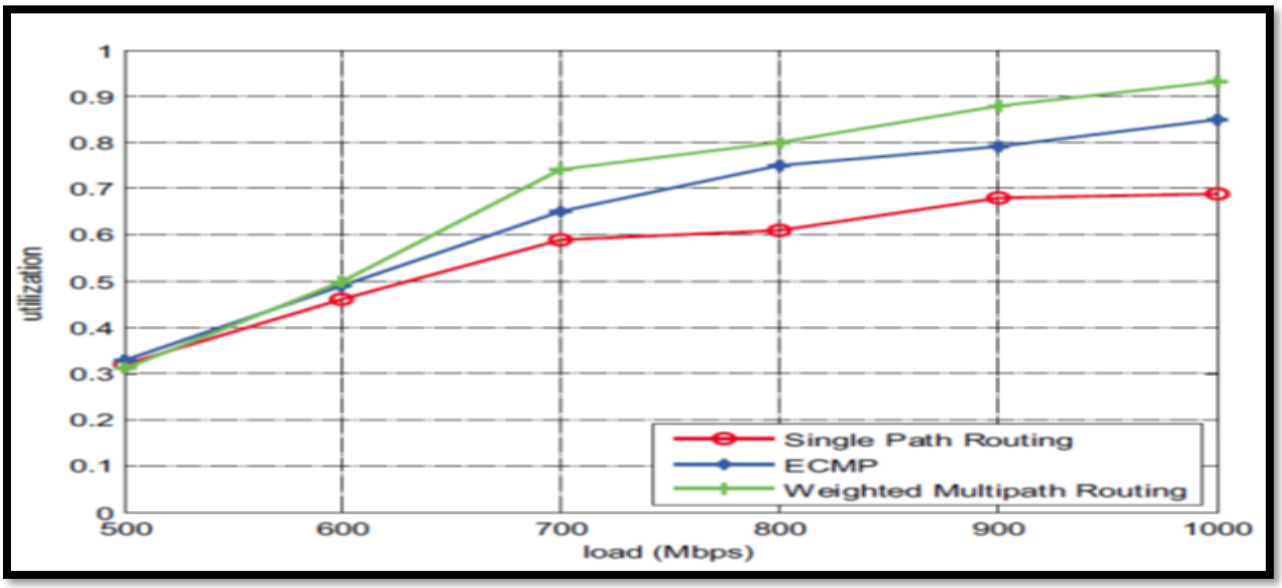

Figure 15: Utilization performance of the weighted Multipath routing versus Single Path Routing, ECMP [11]

Splitting the elephant flows seems to be a good way to better utilize the network and avoid congestion due to elephant flow. However, we need to compare its performance with the flow-based approaches like DevoFlow, Hedera, Mahout and MicroTE. We also need to weigh the implications of using the flow let switching scheme to avoid reordering problem in terms of added complexity and overhead [12].

2.4 Problem of Handling Mice flows with ECMP

Hedera uses ECMP for mice flows, DevoFlow uses static multipath routing and the microflow path is randomly selected according to a precomputed probability distribution. Mahout uses a static load balancing. Weighted routing algorithm for splitting elephant flows proposed in [11] also uses ECMP for mice flows.

All the above techniques we have discussed focus on the elephant flows and allow the baseline routing techniques to handle the mice flows which divides the bandwidth equally among flows. None of these techniques provide for dynamic, congestion-aware management of mice flows [13].

Mice flows are 90% of datacenter flows, handling mice flows with baseline ECMP is not necessarily optimal, it can cause wastage of bandwidth and loss of revenue for operators. Hotspots may change with time while other links maybe underutilized.

Consider the example in the Figure 16 below, when several users on Rack1 send 1000 short lived flows at the same time to Rack2, If port "ln1" of switch1is already 96% utilized due to elephant flows using regular elephant flow centric methods, the 1000 mice flows will be equally split with 500 flows through "ln1" of switch1 which is already 96% utilized and the other 500 through ln1 of switch2.if the aggregated rate for the 500 mice flows is greater than the available 4% of switch1 "ln1" it will cause congestion.

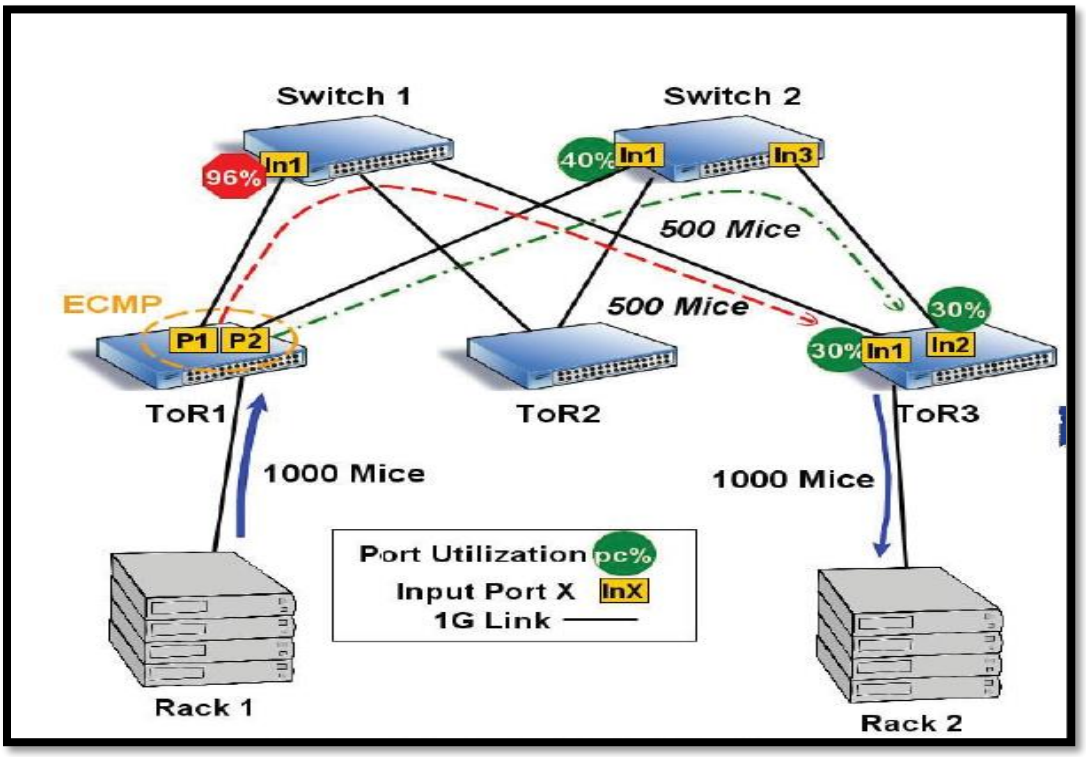

Figure 16: ECMP routing for Mice flows [13]

2.5 MiceTrap to Handle Mice Flows

Intelligent load balancing of mice flows could be used to move the load to underutilized links to reduce congestion. [13] proposes the MiceTrap, which is an OpenFlow-based Traffic Engineering technique that targets the mice flows in a datacenter. It aggregates the mice flows and uses software to find a set of dynamically computed weights, which are then used to spread the traffic across multiple paths for load balancing.

2.5.1 Elephant Flow and Mice Flow Detection

To identify the mice flows, this technique first identifies the elephant flows at the end-host using an existing kernel-level shim layer approach. The shim layer monitors and marks a flow as an elephant flow when its size exceeds a predefined rate threshold over a given time window and all other unmarked flows are thus identified as mice flows. Once elephant flows are detected they are handled by an elephant flow scheduling technique.

2.5.2 Mice Flow Routing

Considering the large number of mice flows, its not practical to handle each mice flow individually due to cost and resource implications and the burden it will put on the controller. Thus, a mice aggregation module is used to collect incoming mice flows and group them based on their destination and uses the group and flow tables together to provide multi-path routing. All the incoming mice flows that match the flow-entry destination IP are pointed to a group. The group table contains the action buckets with each bucket corresponding to a possible path the flow may take to reach its destination. The controller will install a single rule for all the flows to the same destination.

2.5.3 MiceTrap Architecture

The MiceTrap Controller in figure 17 consists of the following modules:

- **Topology**: This module has information on all the links that are up in the network.
- **Stats Collector**: It collects statistics on the link load from the switches by periodically polling the switch.
- **Path Weight:** This module is responsible for computing the path
- **Mice Path Computation:** It computes the shortest path between any source and destination pair for the mice flows.
- **Elephant Path Computation:** This block computes the paths for the elephant flows.

- **Set of Rules and Path Installation:** This block sends the computed rules to the switches.
- **Network Map:** This block has entire network state information such as the traffic matrix and all established routes etc.

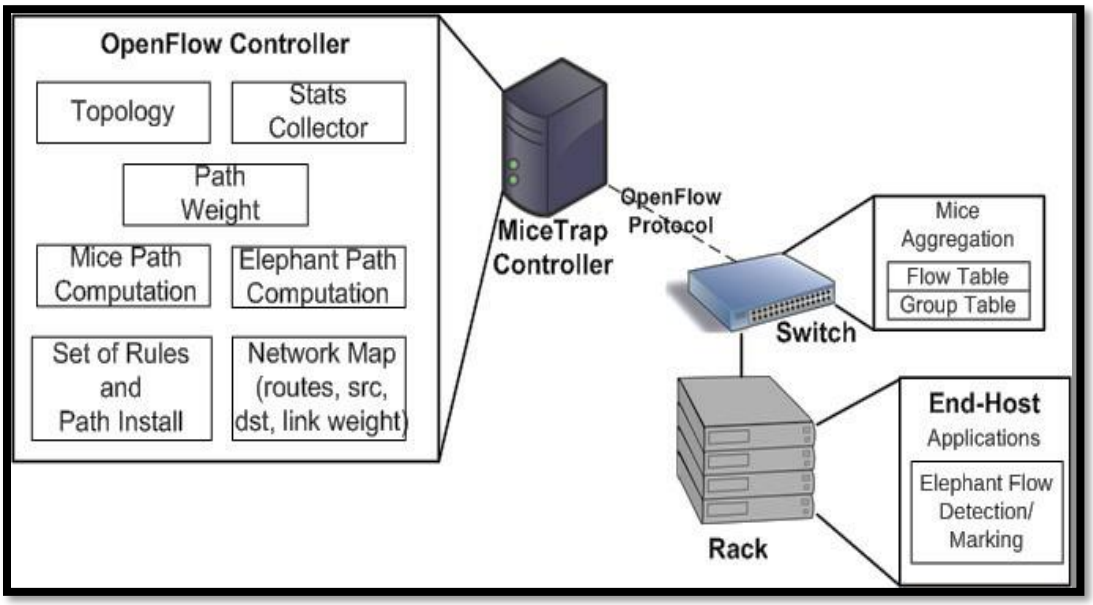

Figure 17: MiceTrap Architecture [13]

2.5.4 MiceTrap Multipath Routing

MiceTrap leverages on the OpenFlow group option as well. It handles multiple mice flows as a single forwarding aggregate and then spreads the flows within an aggregate via multiple paths. This is done using a weighted routing algorithm that takes current network load into consideration.

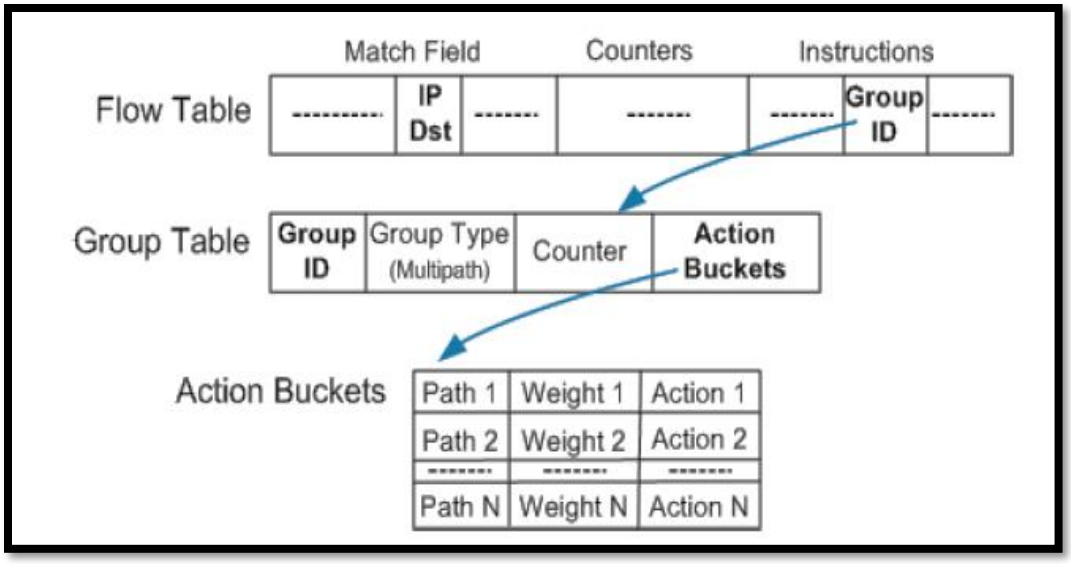

Figure 18: OpenFlow-based Multipath Example

For example, as in figure 18, all the incoming mice flows that have the same destination IP address in the flow table, are given a single group, and the group table contains the action buckets, the action bucket contains the set of paths the flow can take to reach the destination. In this way the available bandwidth is properly utilized, and the mice flows are also protected. To avoid the reordering problem, packets of a single flow are sent on the same path.

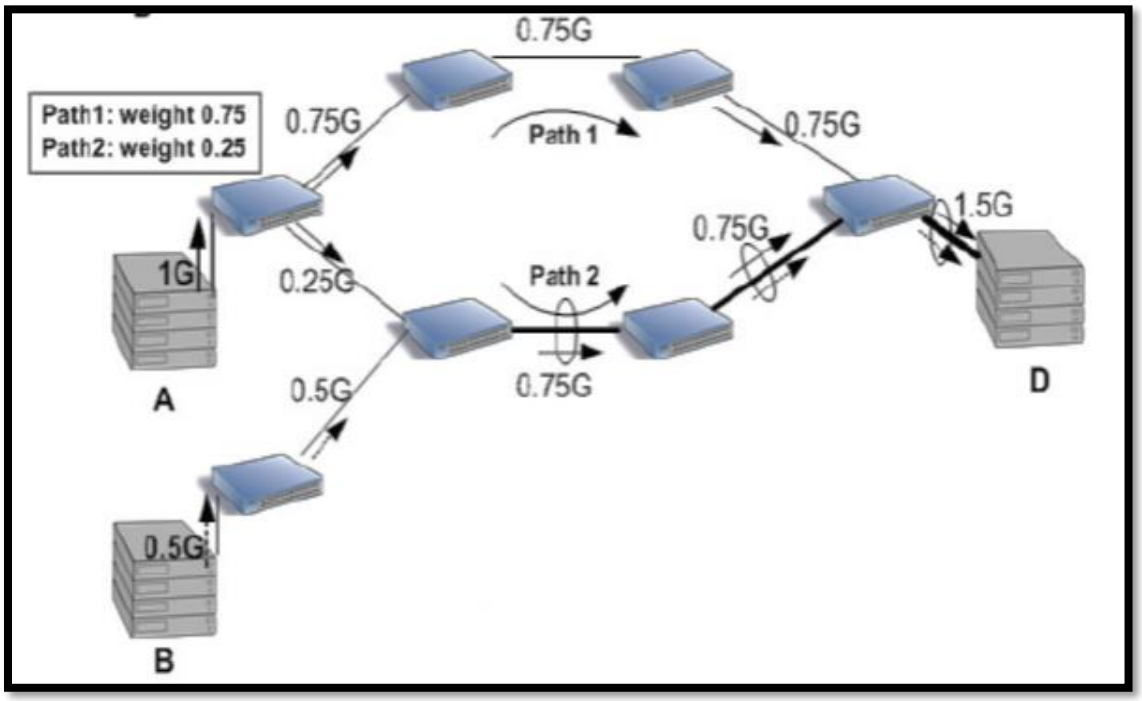

Figure 19: MiceTrap Routing [13]

In figure 19, the earlier problem of overutilizing a link is avoided, and the load is properly distributed over both paths even though the lower link already has a 0.5G load. The traffic is distributed such that only 0.25G is sent on the lower link while more traffic is sent on the underutilized upper link.

MiceTrap saves bandwidth and switch-controller channels by using the group functionality of OpenFlow. A single rule for a destination address can be assigned to a group of flows this will help to reduce wastage of resources, for example, for 1000 mice flows to the same destination, the controller will install a single rule matching the destination, instead of having a rule for each flow. Secondly whenever changes in traffic distribution occur, a single explicit group message can update a set of flow entries avoiding sending an explicit message for each flow.
Weighted routing algorithm achieves better traffic balancing than conventional ECMP, by significantly reducing the load on the links [13].
However, adding the mice flow computation task to the controller may add some overhead and its benefit would need further consideration.

3 Energy Saving

Power consumption has been on a steady rise with the advancement in technology, and one sector that is contributing to a large percentage of this consumption is the Information and Communication Technology [14] [17]. One new technology on the rise is Cloud computing, it has made it easy for organizations to host their resources easily from anywhere with ease, with little or no maintenance cost, on the backend. These cloud computing requirements have introduced the need of super datacenters. To meet the performance requirements of todays time sensitive applications, datacenters must provide sufficient capacity in terms of bandwidth, redundancy and memory.

One fact to consider is that even though all the switches and links in a datacenter are mostly active and consuming energy, they are mostly underutilized, some estimates state that only 30-40% of network devices and links are utilized most of the time. Even though the utilization of the resources varies with time the power consumption is fixed and independent to traffic variations. This is putting lots of pressure cost wise and, on the environment as well, it thus makes it a important to consider methods of reducing the energy consumption, by attempting to make it proportional to the utilization of resources.

The centralized structure of SDN with separation in control and data planes, can be used to deploy energy aware routing algorithms in an efficient and dynamic manner [17]. A lot of research has been done on how to save energy using the SDN paradigm by deploying energy aware routing algorithms efficiently and dynamically. In a datacenter environment the primary culprits of energy consumption are the network units such as links and switches [14].

3.1 SDN Based Power Saving in Datacenter

The two main approaches that are currently deployed to save energy are, link rate adaptive and sleep models [14].
The link rate adaptive model assumes that energy consumption of links is dependent on the data transmission load, thus it aims to save energy by dynamically adjusting the link rates according to traffic demand. While the sleep model attempts to save energy by turning off unnecessary network components such as links and switches and setting other non-working components to sleep mode, most approaches focus on this method [14] [17].

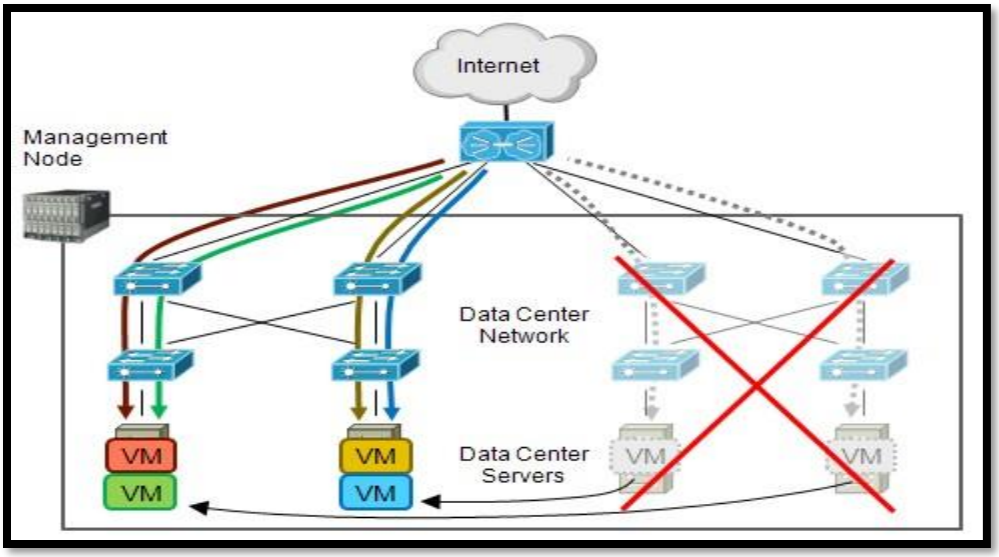

Figure 20: Energy Saving in Datacenter [19]

3.2 Factor to Consider for Energy Saving

In a network most techniques such as QOS and load balancing are deployed to improve its performance, if we reduce the number of active network components this could impact the network performance and quality of service, care needs to be taken to ensure that energy saving doesn't compromise network performance or vice versa.

3.3 QoS differentiation-based energy saving

The method proposed in [15] uses the Iterative Parallel Clustering Algorithm (IPGA) which uses priority scheduling for SDN to save energy, it considers both the QoS requirement and energy consumption of switches. The controller schedules the routing paths and then manipulates the flow tables in the switches, it finds the flows with the highest QOS priority and transmits them preferentially using minimum number of active switches [14] [15].

3.4 ElasticTree Traffic management system

ElasticTree Traffic management system tries to find the minimum subset topology for the Datacenter network that is required to carry the required traffic demands at the time. The controller will turn off the links and switches that are not part of the optimal topology (not required to meet the traffic demand at the time). To find the minimum subset topology ElasticTree proposes three algorithms: Linear Programming (LP)-based formal model, greedy bin packing heuristic, and topology-aware heuristic [18].

3.5 Traffic Aware Energy Saving - Link Utility Based Heuristic Algorithms

[17] tries to solve energy saving in SDN and proposes link utility based heuristic algorithms, NSP and NMU. They can balance a trade-off between energy saving and performance and are topology independent.

3.5.1 Algorithms for Energy Aware Routing

The proposed heuristics main objective is to minimize the energy consumption of links and switches by identifying the underutilized links. Based on the computed information, the controller redirects the flows of selected underutilized links to more utilized replacement path, and then turns off the underutilized links if all flows are redirected. The proposed heuristic algorithms, Next Shortest Path (NSP) and Next Maximum Utility (NMU) are topology independent and based on the utility of links. They aim to replace the under-utilized links with next shortest path and the next path in the direction of the maximum link utility, respectively.

NSP and NMP Example (figure 21)

- If the minimum ration to make the link active, U_{min} is chosen as 0.06 as in figure a.
- The only link with a utilization below 0.06 is link between nodes i and j (0.05), which means it's an underutilized link and can be turned off to save energy, if all the flows are redirected.
- The alternate paths for the traffic between nodes i and j:
  {i-G-j} = 2 hops
  {i-A-B-j} =3 hops
  {i-C-D-E-j} = 4 hops
- Network shortest path (NSP) algorithm choses the next shortest path from the list of available routes as an alternate path for the traffic which is {i, G, j} as in figure b.
- Next Maximum Utility (NMU) choses the link with the maximum utility from the list of available alternate paths {i, C, D, E, J} as in figure c.

After the updates, the flows passing through the edge eij would be redirected to the replacement path. Assuming all links have equal bandwidth in this example, the utility ratio of the link would be added to each link of the replacing path. The minimum threshold value Umin directly affects the set of candidate links.

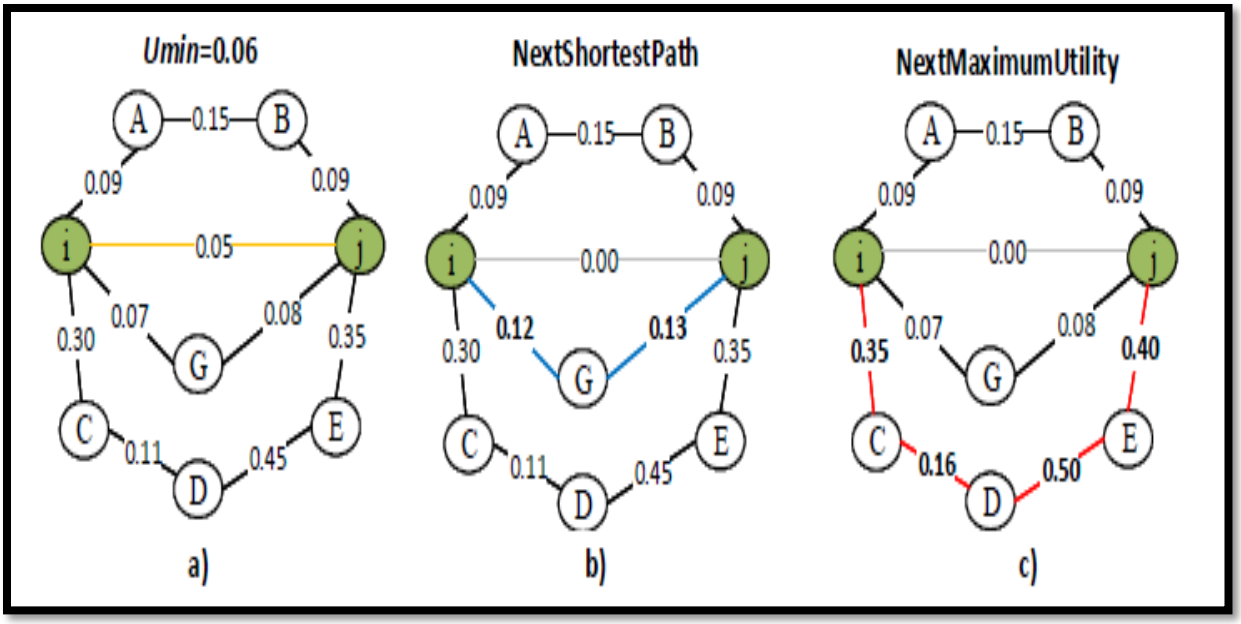

Figure 21: NSP and NMU Example [17]

3.5.2 Energy Saving Performance NSP and NMU

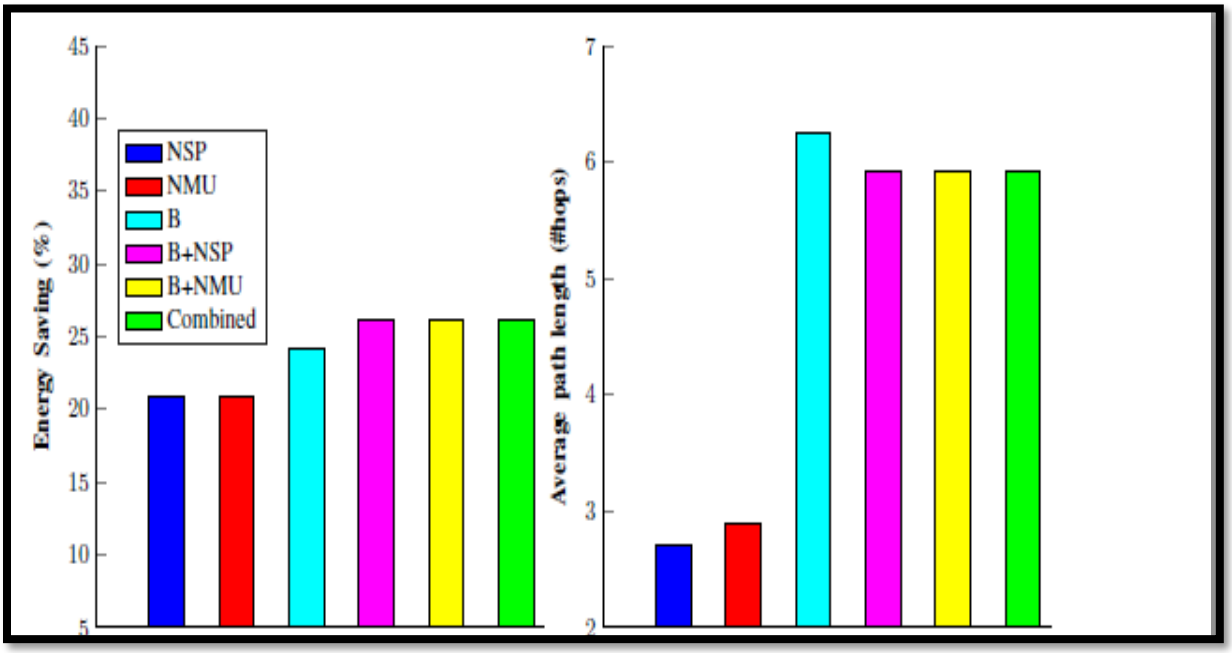

Figure 22: Medium Traffic-Energy Saving and Average Path Length [17]

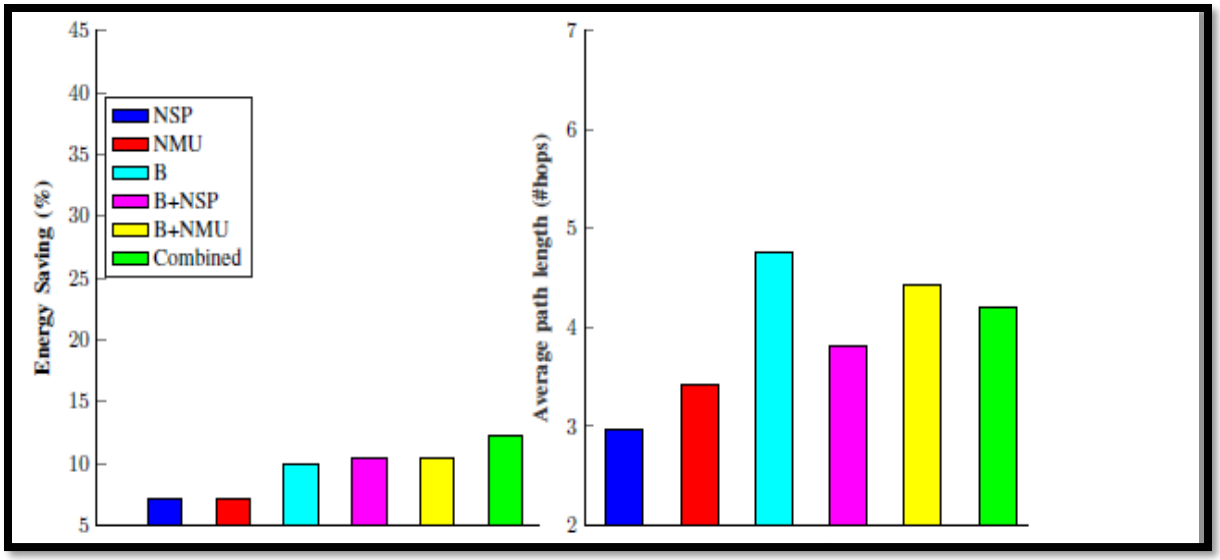

Figure 23: High Traffic-Energy Saving and Average Path Length [17]

From figure 22 and 23 it can be seen that NSP and NMU balance the trade-off between energy saving and performance.

3.5.3 Problem of NSP/NMU:

Consider figure 24, the threshold Umin is set to 0.08. Using the same approach mention in Figure 21 above. The underutilized links traffic is routed to i-A-B-J for NSP and i-C-D-E-j for NMU. **This will maximize the Minimum Link Utilization (MLU) and may eventually cause congestion**. The NMU algorithm may end up finding a suboptimal long path which may affect delay sensitive traffic affecting QoS.

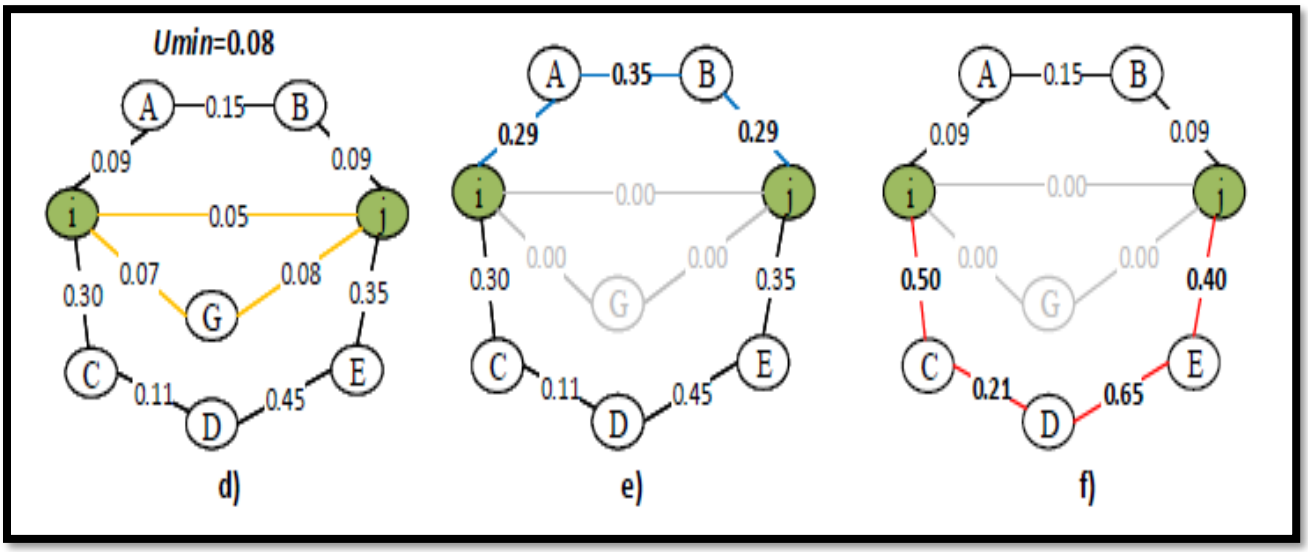

Figure 24: NSP and NMU (maximizing MLU) [17]

3.6 Problem of all the above Energy saving Methods

All the methods discussed above try to assign flows to links with the aim to maximize the link utilization, and this may eventually end up causing severe congestion when there is a sudden change in traffic pattern.

3.7 Energy Saving with Load balancing

To solve the congestion issue that may be caused while trying to save energy, [18] proposes a TE system which aims to save energy and minimize MLU. It consists of two procedures: optimal topology composition and traffic load balancing. The Optimal topology composition aims to find a topology that can accommodate the network traffic and reduce the power consumption just as the other techniques. Once the topology is found the links and switches that are not part of the optimal subset topology are shutdown. However, in the method the traffic is then efficiently distributed over the optimal subset topology with the aim to balance the load, handle congestion and minimize link utilization.

3.7.1 Proposed Architecture

The Traffic Engineering system in [18] has three components as seen in figure 25: **Data Center Network (DCN), SDN Controller, TE Manager**.

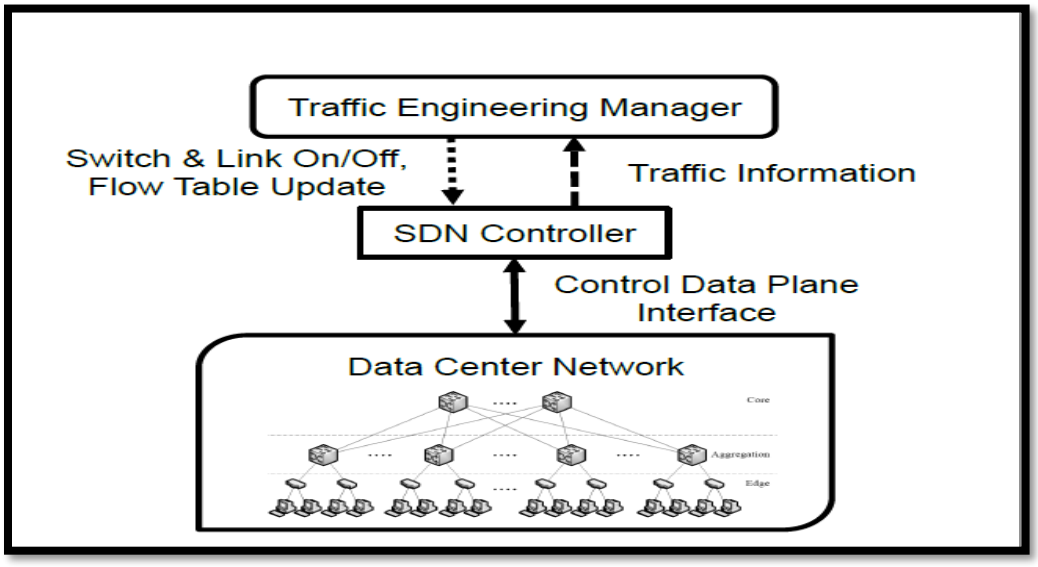

Figure 25: Proposed Architecture [18]

The datacenter SDN switches send their traffic and failure information to the SDN controller, then SDN controller collects the aggregated information and provides it to the Traffic Engineering Manager. The TE manager then uses the Controller provided information to calculate the optimal topology composition and perform load balancing

(see Figure 26 below). Periodically the TE Manager collects traffic and failure information from the SDN controller and finds the minimum subset of links and switches required to meet the entire traffic demand of that period. It then distributes the traffic demands over this optimal topology with aim to minimize Maximum Link Utilization(MLU). The SDN controller finally updates the information in the Switches flow table and turns off/on the switches and links as calculated by the TE manager to minimize power consumption and link congestion.

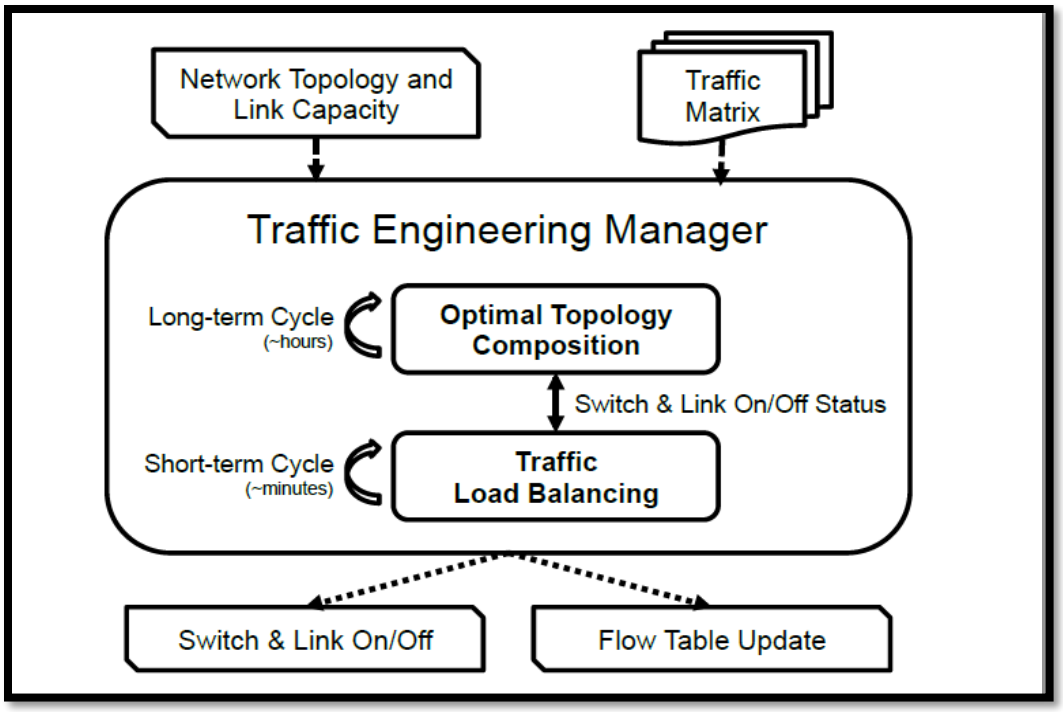

Figure 26: Traffic Engineering Manager Architecture [18]

3.7.2 Algorithms for Energy Saving and Load Balancing

a. **Optimal Topology Composition:**

To find the minimum Topology Composition, [18] proposes the following algorithms:

- Subset Topology Composition using Path-based MCF
- Subset Topology Composition using Heuristic

Their aim is to find the minimum subset topology that minimizes power cost

b. **Traffic Load Balancing Algorithm:**

It uses dynamic traffic load balancing algorithm (equal and unequal cost) to minimize link congestions by distributing traffic demands over the subset topology found by the optimal topology composition algorithms. [18] proposes two different traffic load

balancing algorithms for predicted traffic demands using path based MCF (using Linear Programming) and a heuristic the aim of both is to distribute traffic on the optimal topology such that the MLU is minimized.

3.7.3 Energy Saving Performance

As can be seen in Figure 27 and 28, the proposed mechanism reduces energy consumption to around 41% using the minimum topology composition and reduces the MLU by upto 60% when compared to static routing schemes. It keeps the MLU as low as possible, which is one of the main objectives of Traffic Engineering.

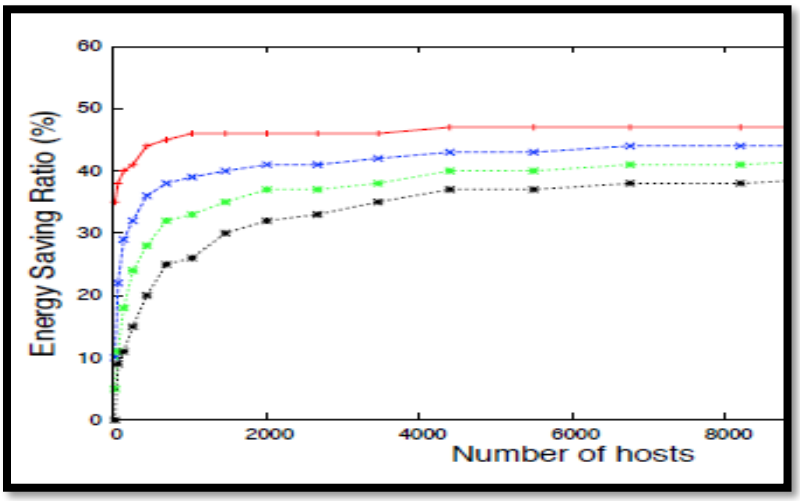

Figure 27: Energy Saving ration (Static/Heuristic/Path based MCF) [18]

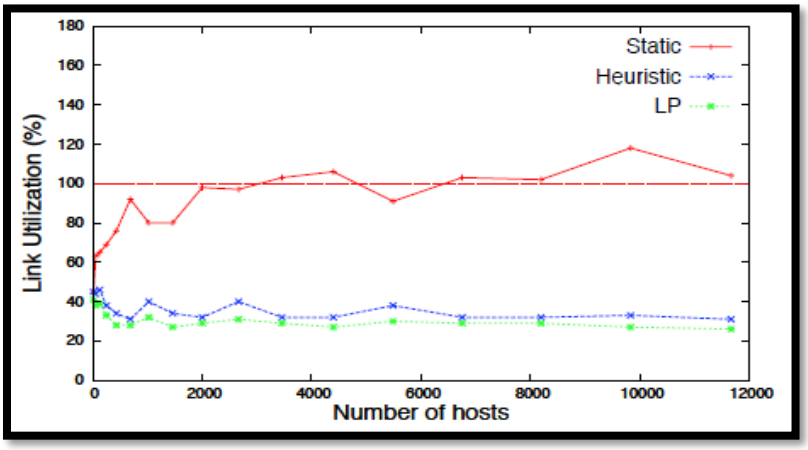

Figure 28: Link Utilization (Static/Heuristic/Path based MCF) [18]

3.7.4 Problem

This technique adds significant overhead due to constant polling for optimal topology and load balancing and there may be additional point of failure/delay at TE Manager.

4 Conclusion

In this paper we have reviewed literature in the field of traffic engineering for both traditional network architecture and SDN. Although manipulating the IGP link weight can achieve some degree of Traffic Engineering, but it lacks the degree of flexibility required to meet modern day Traffic Engineering. MPLS was introduced to compliment this shortcoming and its primary purpose was to be able to Engineer the traffic from congested path to less congested or explicit path. However, MPLS adds significant overhead and complexity to the network as it requires lots of configuration and signaling to be able to achieve its goal.

To reduce the complexity and overhead added by MPLS, SDN based Traffic Engineering techniques such as Hedera, Mahout, MicroTE, DevoFlow have been introduced. Even though these techniques provide superior Traffic Engineering as compared to MPLS. They focus on elephant flows and use a flow-based load balancing approach, thus the elephant flow is limited by the bandwidth of a single physical link. This may still end up causing congestion on certain links while there are underutilized links available.

This problem can be solved by splitting the elephant flows to better utilize the network, however additional complexity will be added to address the reordering issue that may occur. All these techniques (Hedera, Mahout, MicroTE, DevoFlow, Elephant flow splitting), handled Mice flows using some sort of static routing/ECMP. Mice flows account for about 90% of datacenter traffic. Handling mice flows with baseline ECMP is not necessarily optimal, it can cause wastage of bandwidth and loss of revenue for operators. Hotspots may change with time while other links maybe underutilized.

The MiceTrap tries to solve the issue of congestion that may be caused by mice flows. It identifies, and aggregates mice flows and routes them to their destination using the controller in an efficient manner. This method however would end up adding significant overhead on the controller as it need to handle both the elephant flows and mice flows.

To save energy, one proposal was the QoS differentiation-based energy saving technique using IPGA. It aims to find the flows with the highest QOS priority and transmits them preferentially using minimum number of active switches. While ElasticTree Traffic management system tries to find the minimum subset topology for the Datacenter network that is required to carry the required traffic demands at the time. Another proposal uses link utility based heuristic algorithms, NSP and NMU to balance a trade-off between energy saving and performance. However, all these energies saving approaches try to assign flows to links with the aim to maximize the link utilization, and this may eventually end up causing severe congestion when there is a sudden change in traffic pattern.

For energy saving method to be efficient it must consider traffic engineering, the solution to this was found in [17]. It aims to find a topology that can accommodate the network traffic and reduce the power consumption and switches that are not part of the optimal subset topology and shuts them down. The traffic is then efficiently distributed over the optimal subset topology with

the aim to balance the load, handle congestion and minimize link utilization. However, this technique adds overhead due to constant polling for optimal topology and load balancing and adds an additional point of failure/delay at TE Manager.

SDN is a modern technique which simplifies the network management and enables innovation. SDN architecture introduces network programmability and provides a global view of a network and its state. We thus propose an intelligent system that can combine all the techniques discussed in this project and use them dynamically to provide Traffic Engineering and Energy Saving.

Reference

[1] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in SDN-OpenFlow networks," Computer Networks, vol. 71, pp. 1–30, Oct. 2014.

[2] Li and J. Chen, "MPLS Traffic Engineering Load Balance Algorithm Using Deviation Path," in 2012 International Conference on Computer Science and Service System, 2012, pp. 601–604.

[3] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," IEEE Communications Magazine, vol. 40, no. 10, pp. 118–124, Oct. 2002.

[4] R. Trestian, G. M. Muntean, and K. Katrinis, "MiceTrap: Scalable traffic engineering of datacenter mice flows using OpenFlow," in 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), 2013, pp. 904–907.

[5] Class note

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

[7] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, andA. Vahdat, "Hedera: Dynamic Flow Scheduling for Data Center Networks", in Proc. 7th USENIX Symp. on Netw. Syst. Design &Implemen. NSDI'10, San Jose, CA, USA, 2010, vol. 10, pp. 19–19.

[8] A. R. Curtis, W. Kim, and P. Yalagandula, "Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection", in Proc. 30th IEEE Int. Conf. Comp. Commun. IEEE INFO-COM 2011, Shanghai, China, 2011, pp. 1629–163.

[9] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine grained traffic engineering for data centers", in Proc. 7th Conf. on Emerg. Networking Experim. & Technol. Co-NEXT'11, Tokyo, Japan, 2011, p. 8.

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

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

[12] S. Kandula, D. Katabi, S. Sinha, and A. Berger, "Dynamic Load Balancing Without Packet Reordering," SIGCOMM Comput. Commun. Rev., vol. 37, no. 2, pp. 51–62, Mar. 2007.

[13] R. Trestian, G. M. Muntean, and K. Katrinis, “MiceTrap: Scalable traffic engineering of datacenter mice flows using OpenFlow,” in 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), 2013, pp. 904–907.

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

[15] B. Y. Ke, P. L. Tien, and Y. L. Hsiao, “Parallel prioritized flow scheduling for software defined data center network,” in 2013 IEEE 14th International Conference on High Performance Switching and Routing (HPSR), 2013, pp. 217–218.

[16] A. Amokrane, R. Langar, R. Boutabayz, and G. Pujolle, “Online flow-based energy efficient management in Wireless Mesh Networks,” in 2013 IEEE Global Communications Conference (GLOBECOM), 2013, pp. 329–335.

[17] B. G. Assefa and O. Ozkasap, “Link utility and traffic aware energy saving in software defined networks,” in 2017 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), 2017, pp. 1–5.

[18] Han, S. s Seo, J. Li, J. Hyun, J. H. Yoo, and J. W. K. Hong, “Software defined networking-based traffic engineering for data center networks,” in The 16th Asia-Pacific Network Operations and Management Symposium, 2014, pp. 1–6.

[19] https://www.google.ca/search?q=energy+saving+in+datacenter&source=lnms&tbm=isch&sa=X&ved=0ahUKEwj8jM6jocXaAhVE9IMKHTodAM4Q_AUICigB&biw=1536&bih=734#imgrc=VMDdSbRHkueuKM