Department Of Computer Science & Information Systems

Semester Report

CS F266 : Study Oriented Project

Project FlexNet

Software Defined Networking in Data Centers

INVESTIGATIONS INTO COEXISTENCE OF OPEN SOURCE AND PROPRIETARY SWITCHES IN SDN.

Under the supervision of

Prof. Dr. Rahul Banerjee

(Professor, Head CSIS)

Submitted By

Vidhi Jain (2014A7PS113P)

Birla Institute of Technology & Science

Pilani

Acknowledgement

We would like to thank Prof. Dr. Rahul Banerjee for giving us the opportunity to work under his guidance. We have been granted experimental research area in Centre for Software Defined Networking Research Lab, BITS Pilani, Rajasthan. We thank the Software Development and Educational Technology Unit (SDETU) for providing us with computing resources for our simulations and experiments. We would also like to thank Mr. Pranjal Ranjan for his valuable insight, for breaking big challenges into achievable goals, and for giving us periodic feedback at each step. The work has been done in collaboration with fellow students, M Sharat Chandra (2014A7PS108P), Kapil Singhal (2013A7PS038P) and Ameesha Mittal (2014A7PS107P).

Contents

1. Introduction

In today's age of data driven technology, the Data Centers need to manage the network flows efficiently. Software Defined Networking (SDN) brings the unprecedented flexibility and power of programmability on networks. It is an emerging paradigm to unbundle the vertical integration of the controller and the dataplane in the proprietary devices. Separation of a network's control logic from the underlying routers and switches will result in logical centralization of network control, and introduce the ability to program the network. [1]

Traditional networks have used hard-coded algorithms as forwarding protocol, dictated by the hardware manufacturers of the underlying communication devices. The use of a programmable dataplane allows control over end-to-end transfer of critical information and operation optimisation based upon real-time traffic patterns, and management requirements (also known as policy).

The size, heterogeneity, and quality requirements, particularly in the future PC systems will soon make the conventional methodologies for configuration, optimization, and troubleshooting networks inefficient and eventually obsolete [2]. To understand the various components of SDN, we have developed a realistic, cost-effective test-bed for research in academia. This report also focuses on the switching techniques in SDN, both in software and hardware. We further explore the interoperability of the open source and proprietary switches in such networks.

2. Motivation

Software Defined Networking (SDN) offers opportunity to program smarter networks. To unleash the power of programmability on networks, the theoretical knowledge needs to be implemented on an experimental research test-bed for understanding SDN components. There is lack of credible, scalable and robust experimental research platform for hands-on investigation of native SDN, SDN-enabled and hybrid network scenarios.

There are many research test-beds that have enabled large-scale experimentation like 'Global Environment for Network Innovations' (GENI) in USA [3], 'OpenFlow in Europe: Linking Infrastructure and Applications (OFELIA)' in Europe [4] and 'OF@TEIN: Toward an OpenFlow-enabled SDN (Software-Defined Networking) Infrastructure over TEIN' over Korea [5]. 'Corporation for Education Network Initiatives in California' (CENIC) [6] in California, and 'Research Infrastructure for large-Scale network Experiments' (RISE 2.0) [7] in Japan. The broad objective of these projects is to provide a large scale network research testbed, but their benefits have not yet reached the academia in developing countries. Without a systematic understanding of essential concepts, advantages and challenges through experiment and implementation, it is unlikely to achieve its promising objectives under real-world constraints.

A simplistic design for SDN research testbed is required, which includes infrastructure, services and software on which various experiments can be conducted for testing of scientific hypotheses. In networking scenario, a typical testbed consists of Compute devices (servers), Network devices (switches, routers) and Storage (disks).

2.1 Understanding SDN

The Open Networking Foundation (ONF) [8] is a non-profit consortium dedicated to development, standardization, and commercialization of SDN. ONF has provided the most explicit and well received definition of SDN as follows:

> *Software-Defined Networking (SDN) is an emerging network architecture where network control is decoupled from forwarding and is directly programmable* [9][2]

The architectural principles of SDN can be summarised as in [10]:

1. ***Separation of network control and forwarding functions***

The dataplane comprises of the switching elements in the network. Our dataplane consists of both software switches (OVS instances) and OpenFlow-enabled hardware switch (HP 3500 yl).

2. ***Logically centralized network control elements***

An SDN controller is an application that manages flow control to enable intelligent networking and provides programmability of network via NBIs. In our setup, a RYU controller, present in one of the VMs installed in the server, acts as a logically centralized controller for the entire network.

3. ***Programmable interfaces for the network (at multiple levels)***
    a. ***Southbound Interface***

Southbound APIs are used to communicate between the Controller and the dataplane of the network. As OpenFlow is the most popular implementation of SDN and is currently used in production settings, much research has been performed to utilize and improve the protocol. Our testbed uses OpenFlow 1.3 as the southbound interface to defines in a standardised way how the SDN Controller should interact with the forwarding plane and make adjustments in the network by installing flow rules.

    b. ***Northbound Interface***

The northbound interface is used to communicate between the Controller and the services and applications running over the network. In our architecture it is the REST API provided by Ryu which enables programmability in the entire network.

SDN implementations have some typical characteristics which differentiate them from the traditional network architectures. Firstly, network controllers are independent software elements. Secondly, system elements use open, standardized interfaces.

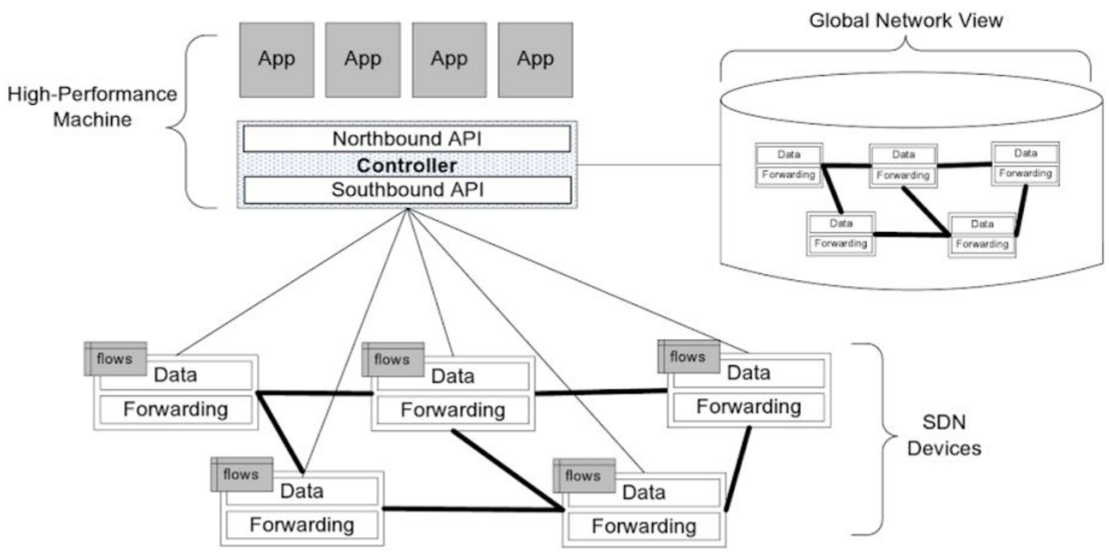


Figure 1. SDN Operation Overview

Source:[36]

2.2 SDN in Data Center Networks

Data Centers need to be Scalable, Secure, Shared, Standardized, and Simplified (5 S's) . They need to have workload mobility, which means that the network must comprise of large L2 domains for VM mobility. In a data center, there is significant server-to-server traffic (East-West Traffic) as compared to server to user (North-South Traffic). One Facebook request required 88 cache looks, 35 database lookups, 392 backend RPC calls. Internet traffic 935X the http request/response [12] There is also a requirement for congestion management on Ethernet. [11]

Typical architectures today consist of either two- or three-level trees of switches or routers. A three-tiered design has a core tier in the root of the tree, an aggregation tier in the middle and an edge tier at the leaves of the tree. A two-tiered design has only the core and the edge tiers. For experimentation purpose, topology design has been restricted to the three-tier design.

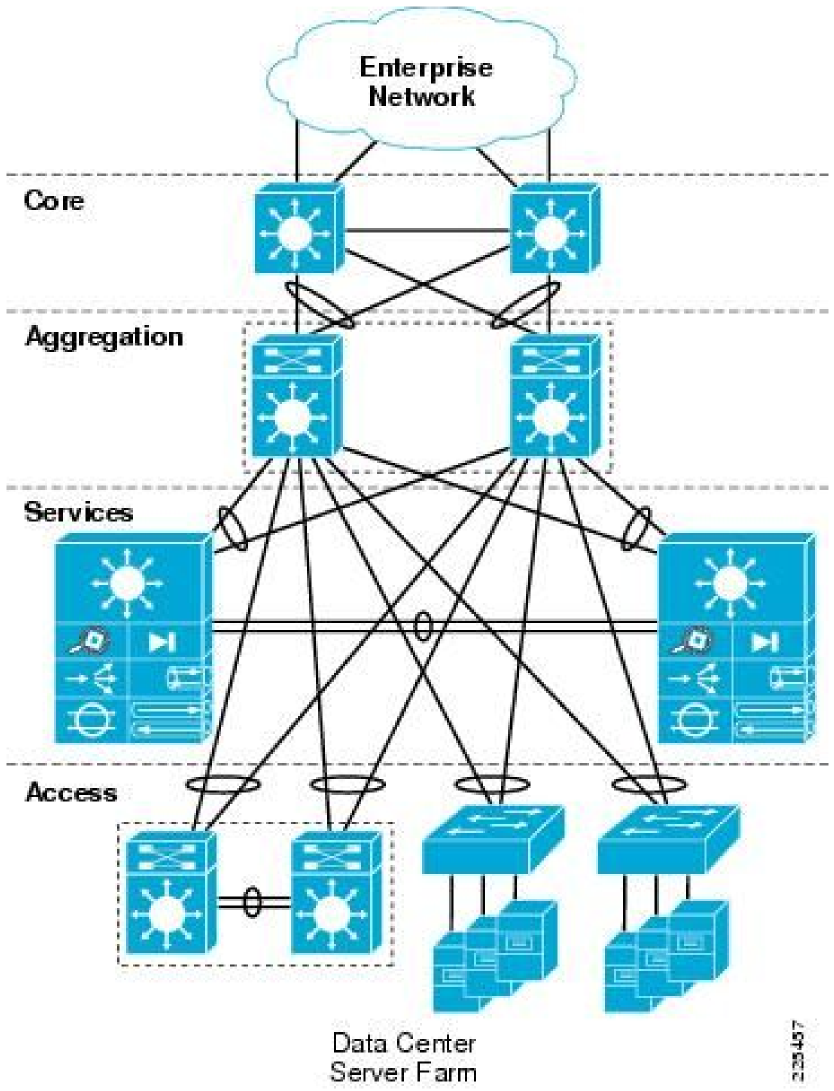


Figure 2. A Typical Data Center Topology (Cisco Data Center Infrastructure [13])

SDN technology is promising technology that offers one touchpoint solution (SDN Controller) versus thousands of touchpoints (Network Elements in legacy networks). This greatly reduces network management complexity as well as network downtime [14]. With programmability on network behaviour, SDN can meet the service level agreement of the customers with transparency and better control.

3. Design of Network Research Test-Bed

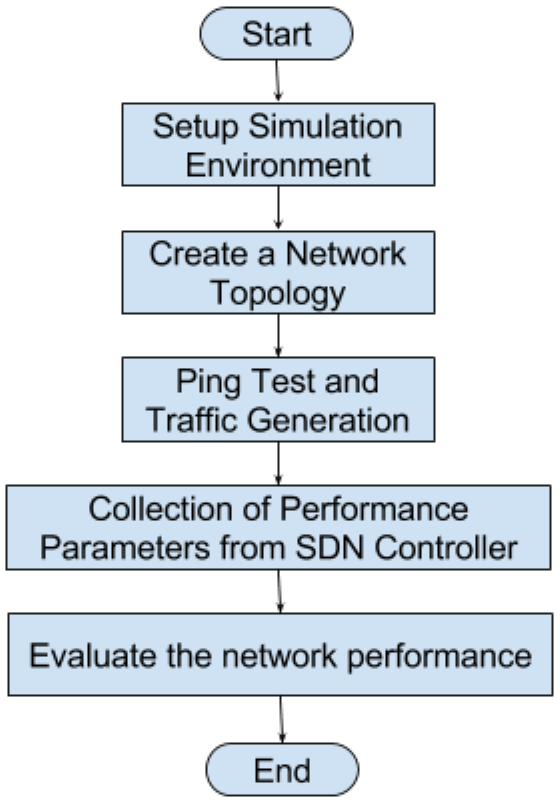

Start
Setup Simulation Environment
Create a Network Topology
Ping Test and Traffic Generation
Collection of Performance Parameters from SDN Controller
Evaluate the network performance
End

Figure 3. Process Flow Design of Experimental Network Research TestBed

We have followed an incremental approach to build a testbed for large-scale experimentation, which can enable a novice to setup own testbed to understand the concepts of SDN. We demonstrate results of some experimental scenarios and discuss about the variety of ways to utilize the research testbed.

Table 1. Hardware Requirements

Capacity	**Phase I**	**Phase II**
CPU Type	Intel Core i7	Xeon^{TM}
Processor	1	2
Cores	8	12
CPU Frequency (GHz)	3.40 GHz	2.5 GHz
Memory (GB)	32	64
DISK (GB)	1000	8000
NIC (Mbit/s)	1000	1000 (x 4)
Kernel	3.10.0+2	3.10.0+2
XEN	6.5	6.5
Model	Dell 7040 OptiPlex	Lenovo System x3690 M4

3.1 Preliminary Setup

The preliminary setup is focused on getting quick understanding of SDN components. As described in Table 1, we have focused on utilizing the readily available desktop CPUs and traditional network switch to begin with. This has several advantages: cost-effectiveness, for beginner's experience, to identify different challenges in domain of SDN. It contains 3 XenServers. In addition, we have used 2 Ubuntu storage pools for ISO files and Virtual Machine (VM) files.

To setup the storage pools, we configured Network File Sharing (NFS). The Ubuntu systems are configured as the NFS server while the remaining host servers in the network have been configured as NFS common.
For VM disk image files, the directory `/var/lib/libvirt/images` is mounted in each of the server machines to the respective NFS directory in the storage pool. For ISO files, the new VM creation on any of the servers can specify `boot from NFS`.

We also have a Windows machine containing XenCenter for monitoring. We have setup the Controller in a VM which controls the Open vSwitches in each of the XenServers. Each XenServer can be assumed to be a virtual datacenter rack, with Open vSwitch (OVS) acting as a top of the rack (TOR) switch. We set up ssh server in each machine to remotely access them for configuration. We additionally have two display monitors to control a particular machine when network intensive tasks are being carried out.

The XenServers were initially connected via a legacy/non-SDN switch. This was to observe the challenges involved in operating SDN with the real world constraint of legacy network. An SDN enabled switch would have given the controller a full view of network topology. With this setup, we have been able to minutely observe where the controller fails to discover the network when legacy network device is used. The failure is primarily due to different link discovery protocols and message packets.

Figure 4. Physical topology – Phase 1

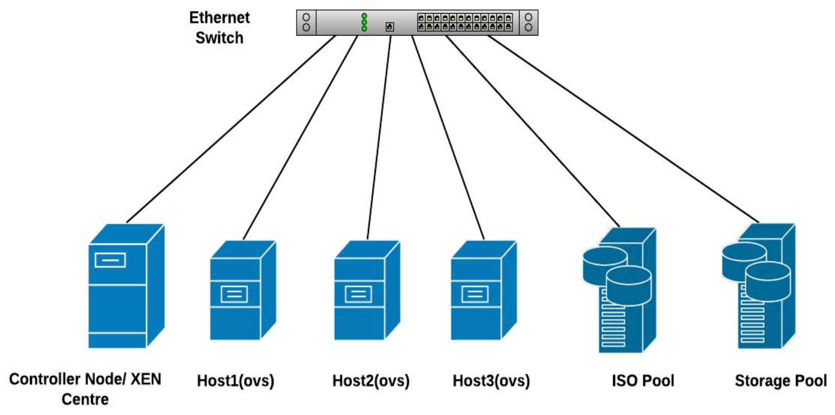
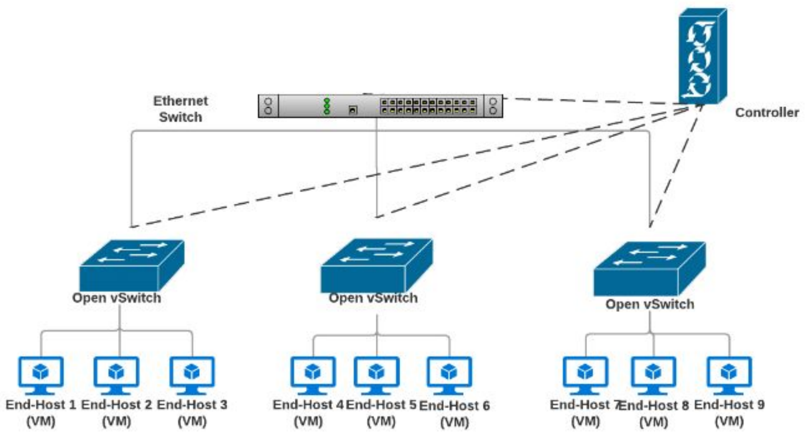


Figure 5 . Logical topology – Phase 1



3.2 Need for SDN hardware switch

The above preliminary setup has helped us to precisely observe and identify the challenges in hybrid environment of legacy networks and SDN. We discuss the problems observed with this setup.

- The traditional hardware network switch has its own operating system and cannot be configured by our Controller.
- The Link Layer Discovery Protocol (LLDP) messages are incompatible within SDN and traditional networks. This leads to inconsistent topology mapped by the Controller.
- In order to configure the network for optimized packet flow for a higher priority task, the Controller must install the corresponding flow rules to the switches. In not a complete SDN environment, this task cannot achieve the maximum possible efficiency.

These reasons necessitates an SDN enabled hardware switch for observing the complete impact of software defined network environment.

3.3 Need for Server-class Machine

The research testbed is well-suited for preliminary experiments to understand the concepts of SDN. But in order to conduct large-scale experimentation, we required to move to server class machines. We have used three Lenovo System x3650 M4 server [17] which can have upgrade path to 16 hard-disk drives (HDDs) or solid-state drives (SSDs). It has four standard embedded Gigabit Ethernet ports and two optional embedded 10 Gb Ethernet ports without occupying PCIe slots. The system Intel Xeon processors and supports up to two processors, 24 cores, and 48 threads maximize the concurrent execution of multithreaded applications. It supports up to 1866 MHz memory speeds. These powerful machines are thus suitable for use in managing databases, virtualization, HPC and cloud applications.

3.4 Final Experimental Setup

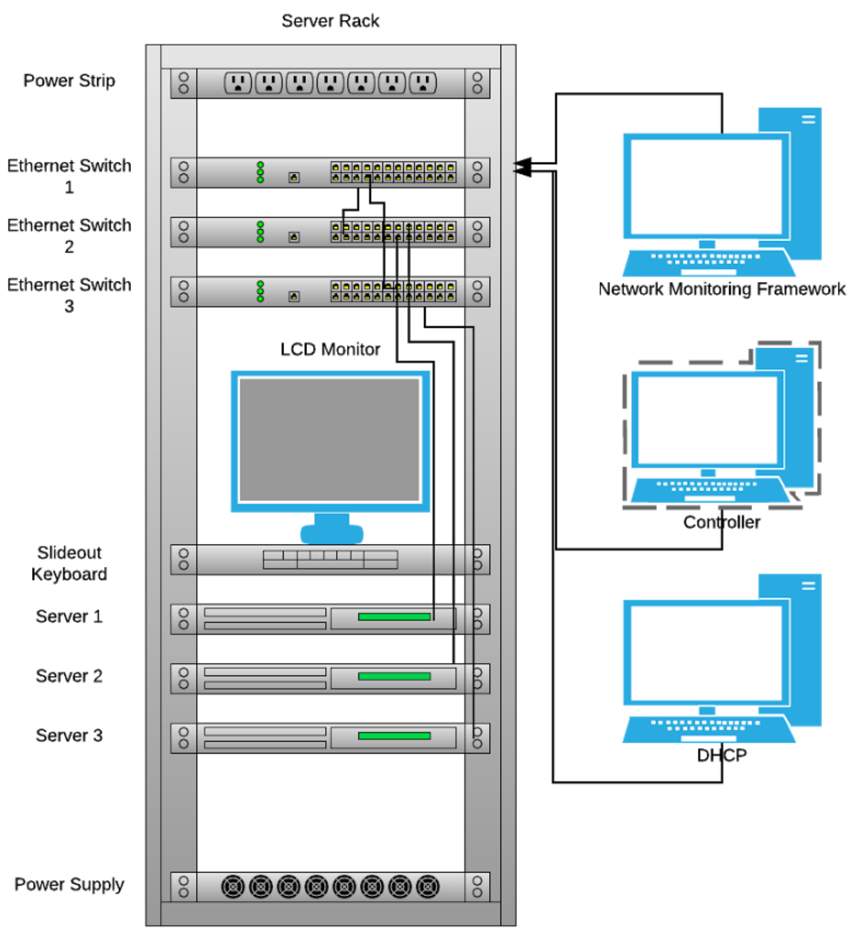
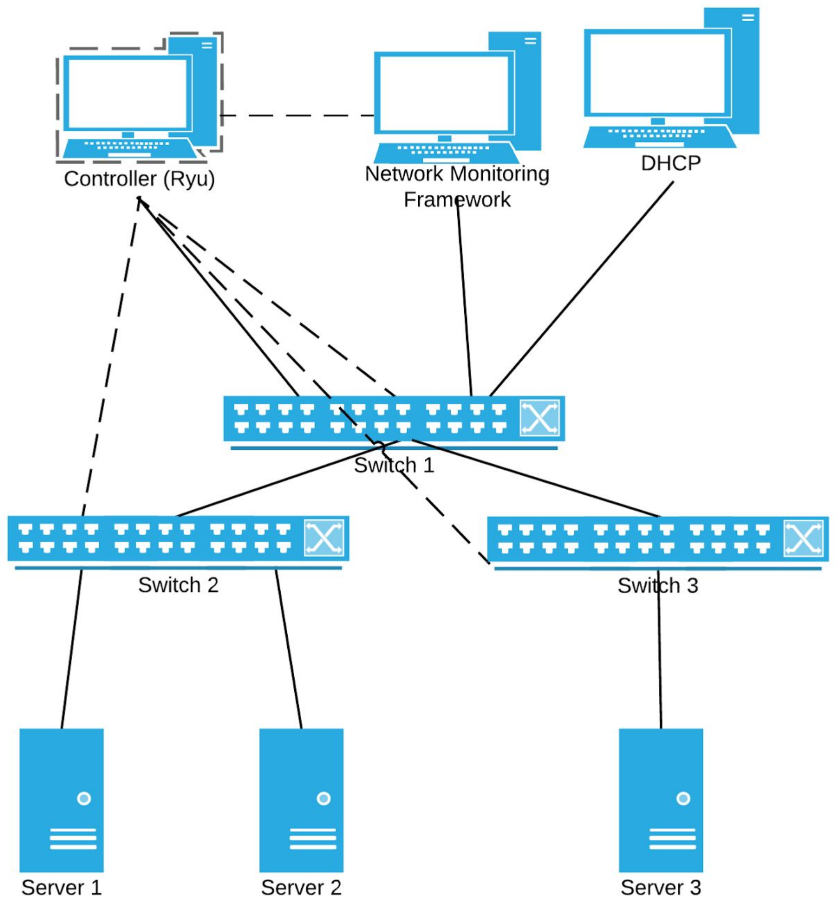


Figure 6. Physical topology - Phase 2

We have setup the following testbed for our final experimental evaluations. In Figure 3, we show the physical setup of the testbed. We have three server class machines which support four NIC. This enabled us to create the logical topology as shown in Figure 4. Each of the eight NIC interface has been associated to Open vSwitch. Thus we achieve the two-level hierarchy model for understanding data center networks with //HP SDN switch name as core switch, the eight OVS as the top of the rack switch and their corresponding VMs as the rack servers.

Figure 7. Logical Topology - Phase 2



4. Analysis of Switches in SDN

Software Defined Networking evolved from the concept of decoupling the lower-layer packet/frame forwarding from the control function that intelligently determines how application traffic should be transported. The separation of the control plane from the forwarding plane allows networks to facilitate packet processing in new and innovative ways and created a new paradigm for network virtualization. SDN has opened up a whole new world of network design and enabled creative approaches to networking. SDN has also caused us to reconsider how security policies are enforced within the network. [16]

4.1 Software Switches

The study is important to analyse the right software switch which can make a network more flexible, intelligent and cost effective. One of the most widely used is Open vSwitch (OVS), a production quality, multilayer virtual switch, available as open source project. [18]

OpenFlow[19] is the initial standard interchange interface [20] characterized between the controls and forwarding layers of SDN. OpenFlow permits immediate access and control of the forwarding plane for network components like switches and routers, both physical and virtual (hypervisor-based).

An OpenFlow enabled switch [21] is the basic forwarding device that forwards the packets according to its flow table, which is similar to traditional forwarding tables but not managed and maintained by the switch. It is connected to the controller via a secure channel on which OpenFlow messages are exchanged between the switch and the controller. [22,23,24]

A switch in OpenFlow network has one (or more) flow table(s) that contains a set of entries, each of which consists of fields named match, action, and counters. All packets that have been processed by the switch, are compared against the flow table. If a packet header matches a flow entry, an action for that entry is performed on the packet (e.g., the action might be to forward a packet out a specified port). In the event that no match is found, the packet is sent to the controller over a secure channel. The counters are reserved for collecting statistics about flows. They store the number of received packets and bytes, as well as the duration of the flow. A flow table consists of flow entries for OpenFlow version 1.0.

Table 2: Main Components of a Flow Entry in a Flow Table

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

Match Fields are to match against packets. These consist of the ingress port and packet headers, and optionally other pipeline fields such as metadata specified by a previous table.
Priority is required to match precedence of the flow entry.
Counters are updated when packets are matched.
Instructions define how to modify the action set or pipeline processing.
Timeouts refer to the maximum amount of time or idle time before flow is expired by the switch.

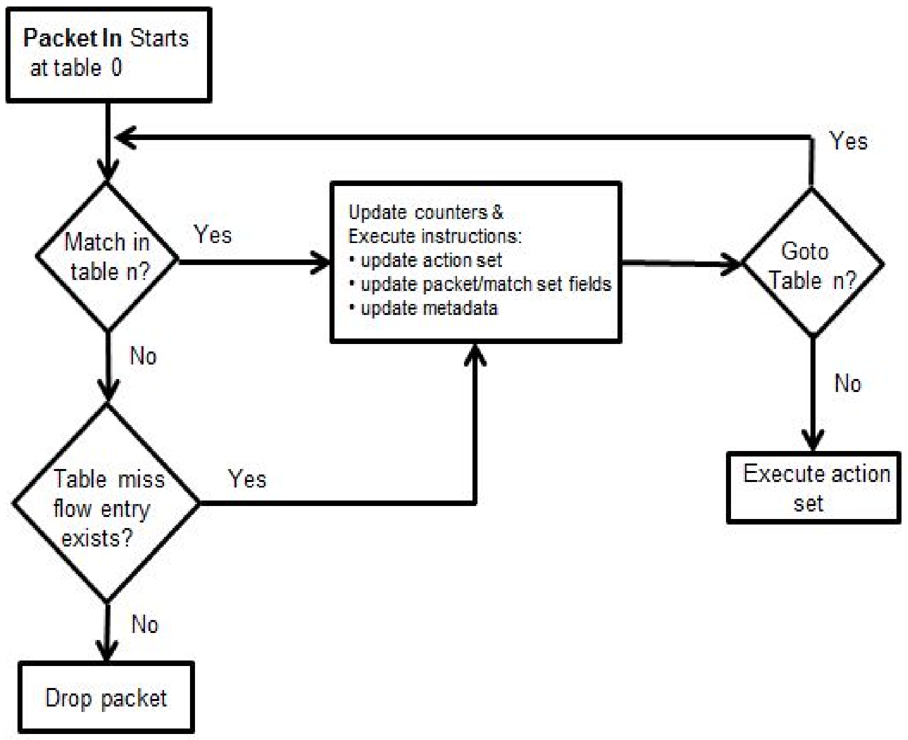
Cookie is the opaque data value chosen by the controller. It may be used by the controller to filter flow entries affected by flow statistics, flow modification and flow deletion requests. It is not used when processing packets.
Flags alter the way flow entries are managed, for example the flag OFPFF_SEND_FLOW_REM triggers flow removed messages for that flow entry.

The header field in the flow table consists of different fields on which the incoming packets are compared to:

Incoming switch port
IEEE 802.3 Ethernet source and destination address
IEEE 802.3 Ethernet type
IEEE 802.1Q VLAN ID and priority
IP source and destination address
IP proto field
IP Type Of Service (TOS) bits
TCP/UDP source and destination ports

If an ingress packet matches one of the match fields in the flow table, an action should implied to that packet. Forwarding the packets action to each physical port must be supported by the OpenFlow switch. Additionally there are virtual ports defined by the OpenFlow standard as special targets that the packets can be forwarded to. The incoming packets can be matched against various fields on every layer of the OSI model for a packet, ranging from the data link to the transport layer as well as on the incoming switch port. In order to match all the header fields, the special value ANY can be used in the Flow table.

Figure 8: Flow Chart shows the packet Flow through the OpenFlow Switch


Packet In Starts at table 0
Match in table n?
Yes
No
Update counters & Execute instructions:
• update action set
• update packet/match set fields
• update metadata
Goto Table n?
Yes
No
Execute action set
Table miss flow entry exists?
Yes
No
Drop packet

Source: **[25]**

4.1.1 Setting up OVS in Citrix Xenserver [26]

In our initial testbed setup, we have used Citrix Xenserver 6.5. Since Open vSwitch requires Python 2.7 or later, we installed Python 2.7 on the servers.

4.1.2 Setting up OVS in Kernel Virtual Machine (KVM)

First, follow the setup instructions in [INSTALL.md] to get a working Open vSwitch installation.

KVM uses tunctl to handle various bridging modes, which you can install with the Debian/Ubuntu package uml-utilities.

Then we need to modify or create custom versions of the qemu-ifup and qemu-ifdown scripts. The following files are created - /etc/ovs-ifup and /etc/ovs-ifdown. [27]

There are other software switches available like Lagopus[43] and Indigo Virtual Switch [44].

Many SDN implementations will be enhanced by linking SDN software to the underlying network hardware. For most data center network applications, the underlying physical network, working in conjunction with SDN protocols, is critical to deliver a highly scalable, high-performance, low-latency network. We discuss about the hardware switches.

4.2 Hardware Switches

Hardware switches promise faster execution than the software solutions [36]. Suppliers such as Cisco, HP, Dell, Avaya, Juniper and Brocade offer high-performance network fabrics in conjunction with SDN to speed data center operations. Vendors such as Pica8, Plexxi and others have designed specific (hardware) switches that are designed to deliver enhanced SDN performance. [28] These implementations have performance critical requirement in data centers. The price differential between commodity and non-commodity switches is a strong incentive to investigate networks from many small commodity switches rather than fewer larger and more expensive ones.

In our testbed, we have used three HP 3500yl-24G PoE+ switches (J9310A). We had setup the vlan configuration to enable openflow [38].
VLAN 10 is arranged as the OpenFlow VLAN. VLAN 1 is set for the out of band administration which utilized for correspondence with the HP VAN SDN Controller. Arrangement (ProCurve) switches require that a different VLAN be utilized for correspondence with the Controller that does not have OpenFlow empowered. A separate out of band port could likewise be utilized if favored.

VLAN 1 = Management VLAN (Used for communication with the Controller. This VLAN does not have OpenFlow enabled.

VLAN 10 = OpenFlow VLAN

4.3 Co-existence of Open Source & Proprietary Switches

The traditional networking switches which are integrated solutions provided by manufacturers are commonly referred as black box switches. These devices do not provide any knowledge of its internal workings. Their implementation is opaque to the users. The OS and hardware are vertically integrated and there are only a few controls that the user can manage. Developing on the black box solutions require extensive knowledge of the OS and APIs used by the switch manufacturers. There are inherent limitations to the extent that one can work on customising these switches and building tools for automation.

Thus, disintegrating the network switch has become essential for building software defined data centers (SDDC) [40]. Several solutions are available to integrate the open source and proprietary solutions for network switches. The variety of devices that have introduced different levels of co-existence of open source and proprietary switches have been discussed as follows.

Open Switches

Open switches are switches in which the equipment and programming are separate elements that can be changed autonomously of each other. For instance, Juniper EX or MX is often bought along with JUNOS, a Cisco Catalyst switch is bought along with IOS.

Bare Metal Switches

Bare Metal switches involve purchase of the equipment which has no brains incorporated. This is also the manner by which assemble application servers. Many organisations like Accton, Quanta QCT, Alpha Networks, and Delta Computer provide bare metal network switches. These are the same organizations are original design manufacturers (ODMs). This is then integrated with various operating systems available like Broadcom's FastPath, Big Switch Networks' Switch Light, Cumulus Networks' Cumulus Linux, and Pica8's PicOS. Juniper OCX1100.

White Box

A white box switch differs from a bare metal switch as it has an operating system installed. The hardware and the OS is not integrated as they are in a “black box” switch. Some of the white box switch solutions are Edge Core Networks - DCSS SwitchOS installed, or white box with Cumulus Linux installed, Juniper's OCX1100 with JUNOS, Pica8 with PicOS.

Brite Box

Brite box refers to BRanded whITE box. "Brite-box" is a Gartner-authored portmanteau of "branded" and "white-box" [39]. Gartner has called brite-box switching a “disruptive trend” which is expected to account for more than 10 percent of data center switching and routing sales by 2018, up from less than 4 percent in 2013. There is a slight distinction from white box, that the brite box switches are made by an ODM with a brand name, similarly as bare metal.

Hyperscale data centers of Google, Amazon and Facebook have utilised open switches and white-box switches. Such transition in network switches has offered several benefits as

- Massive capital cost reduction (5X-7X less p/port cost)
- Reduced vendor lock-in
- Increased software flexibility/programmability

Facebook has developed networking solutions like Wedge, FBOSS [42] and Backpack [41] to enhance the performance of their data center networks. The modular design of Wedge switch allows users to modify or replace any of the components to better meet their needs. The software layer FBOSS helps to to implement a hybrid of distributed and centralized control.

5. Experimental Scenarios

5.1 Networking Monitoring and Benchmarking

A real time network monitoring system is required to keep a watch over the progressive system state. The design requirements for such a system in SDN environment are considered as follows. [29]

Fault Detection Whenever a link or node failure happens, the network monitoring system should detect this and warn as soon as possible. This option is already incorporated in OpenFlow, where the controller is alerted every time, there is a change in the topology. The monitoring system would only require data from the controller.

Per-link Statistics Internet Service Providers (ISPs) require statistics for every link. This would allow them to meet the SLAs and assure QoS within the boundaries of their network, without bandwidth over-provisioning. There are several ways to measure link utilization, packet-loss and delay, but those ways require additional infrastructure and have more drawbacks, which add to the overhead.

Minimum Overhead The proposed solutions should not add too much network overhead. The overhead should be scalable and tolerable by the controller along with the current number of active flows at any moment. The component should be able to obtain statistics based on the routing information, thus, sending a query requests only to those devices that are currently active.

Accuracy Utilizing monitoring solutions that directly poll statistical data from the network present the best results in terms of accuracy, but are not granular enough. The methods used should be capable to achieve accuracy comparable or better than the current implementations.

Granularity The system should be able to account for different type of services. It should be able to make distinction between flows that have specific needs, i.e. require special care (bandwidth, delay, etc.). Furthermore, it should make distinction between applications, as well as, clients. This is another feature that is already integrated in OpenFlow.

We have developed a network monitoring framework using Ryu APIs and monitoring tools like Ganglia as well as Collectd. We have used Ryu [30], a python based controller. For benchmarking, tools like iperf, nuttcp and ostinato have been used.

In the second phase of our testbed, benchmarking with iperf3 between two server connected in SDN-enabled gave the following result [32].

```
Connecting to host 10.0.0.31, port 5201
```

```
[  4] local 10.0.0.32 port 33562 connected to 10.0.0.31 port 5201
[ ID] Interval        Transfer   Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   346 KBytes  2.84 Mbits/sec   24   5.66
KBytes
[  4]   1.00-2.00   sec   127 KBytes  1.04 Mbits/sec   21   5.66
KBytes
[  4]   2.00-3.00   sec   127 KBytes  1.04 Mbits/sec   17   5.66
KBytes
[  4]   3.00-4.00   sec  0.00 Bytes  0.00 bits/sec   15   7.07 KBytes

[  4]   4.00-5.00   sec   127 KBytes  1.04 Mbits/sec   11   5.66
KBytes
[  4]   5.00-6.00   sec   127 KBytes  1.04 Mbits/sec   19   5.66
KBytes
[  4]   6.00-7.00   sec  0.00 Bytes  0.00 bits/sec   20   5.66 KBytes

[  4]   7.00-8.00   sec   127 KBytes  1.04 Mbits/sec   15   5.66
KBytes
[  4]   8.00-9.00   sec  0.00 Bytes  0.00 bits/sec   20   4.24 KBytes

[  4]   9.00-10.00  sec   127 KBytes  1.04 Mbits/sec   13   2.83
KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval        Transfer   Bandwidth       Retr
[  4]   0.00-10.00  sec  1.08 MBytes   909 Kbits/sec  175
sender
[  4]   0.00-10.00  sec   935 KBytes   766 Kbits/sec
receiver

iperf Done.
```

In non SDN environment, we observed the following result:

```
Connecting to host 10.0.0.12, port 5201
[  4] local 10.0.0.15 port 35969 connected to 10.0.0.12 port 5201
[ ID] Interval        Transfer   Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   112 MBytes   942 Mbits/sec   0    236 KBytes

[  4]   1.00-2.00   sec   111 MBytes   935 Mbits/sec   0    236 KBytes

[  4]   2.00-3.00   sec   111 MBytes   935 Mbits/sec   0    247 KBytes
[  4]   3.00-4.00   sec   111 MBytes   934 Mbits/sec   0     247 KBytes

[  4]   4.00-4.97   sec   108 MBytes   937 Mbits/sec   0     247 KBytes

- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval        Transfer   Bandwidth       Retr
[  4]   0.00-4.97   sec   555 MBytes   936 Mbits/sec   0
sender
[  4]   0.00-4.97   sec  0.00 Bytes  0.00 bits/sec
receiver
```

The observed values of bandwidth indicate striking difference in the two environments. The possible reason for unbelievable low performance is that the openflow agent of the hardware switch between the dataplane and the controller is performing on low memory capacity or transfer rate. This could lead to unacceptable delays in the performance of the network switches.

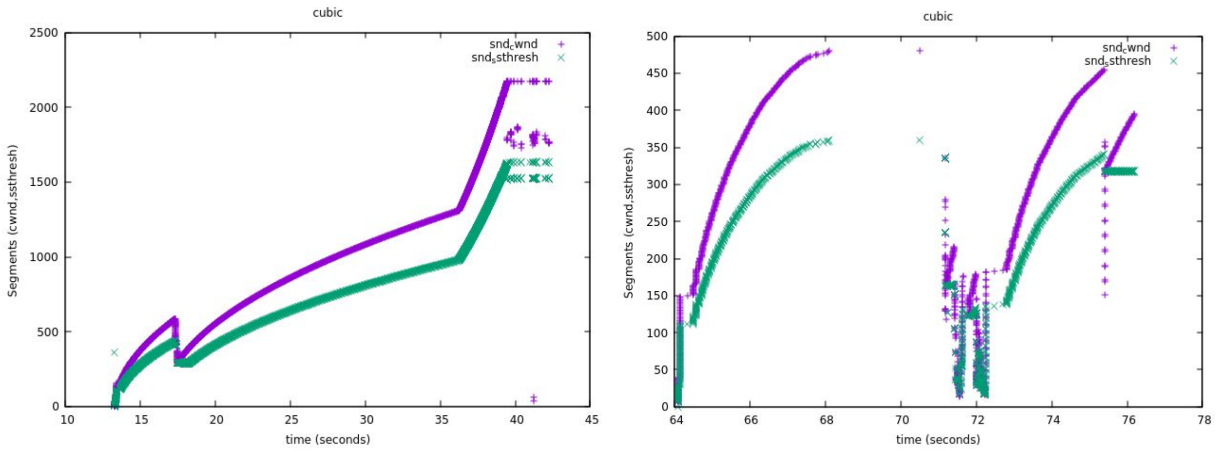
5.2 Virtual Machine Migration

The testbed offers the capability of Live Migration of Virtual Machines from one host machine to another. Several experiments can be conducted exploiting this offering. A comparative analysis of live migration in SDN and Legacy network scenario can be carried out. In Data Centers, where the live migration of Virtual Machine happens very frequently, there might be a need to prioritize such traffic in between servers. Utilizing the programmability offered by SDN, flow rules can be installed which would prioritize the migration which could not have been done in the legacy network. To study the impact of SDN in this scenario, tests were conducted on our test bed replicating the above-mentioned scenario on a small scale.

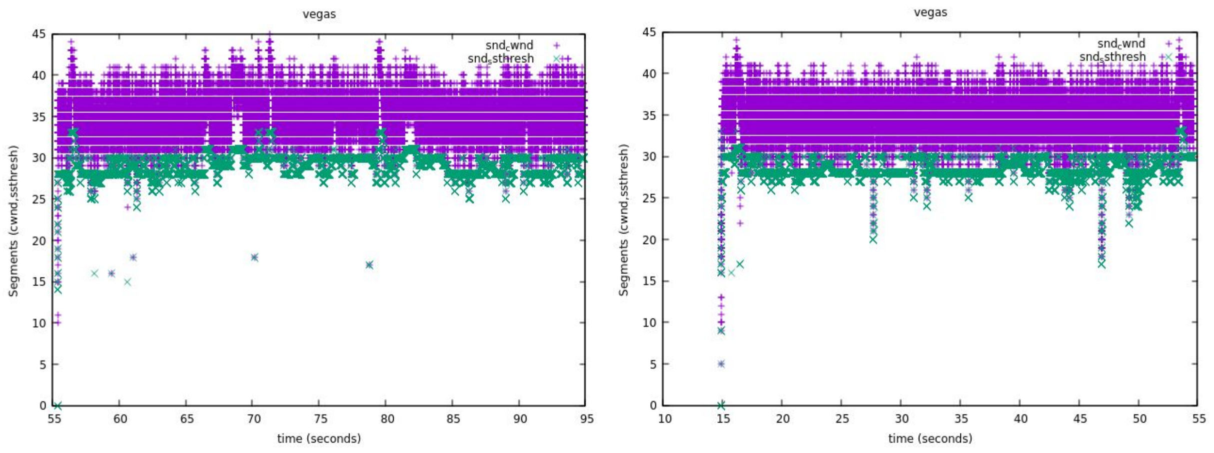
5.3 TCP Congestion Estimate

TCP congestion in SDN environments is seen to increase due to possible transfer of new packets to the controller for routing decision. In Non SDN and SDN environment setup, we observed the following graphs for the different TCP algorithms.

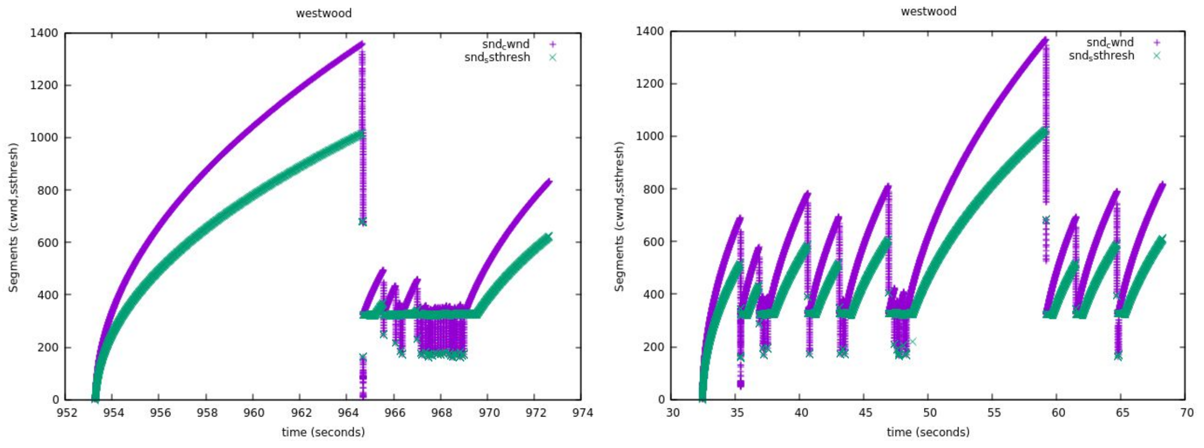
(a) Cubic



(b) Vegas



(c) Westwood

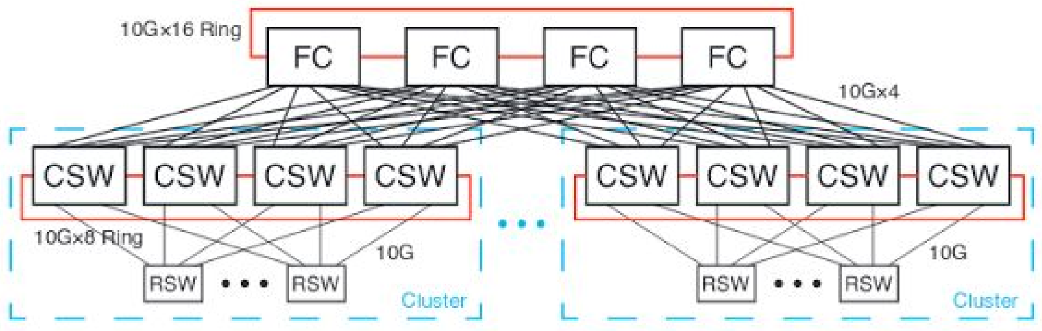


[33] The results from the TCP congestion is necessary to identify the gap in legacy and SDN. The programmatic control can further help in automated load balancing [34] in server cluster to ensure QoS.

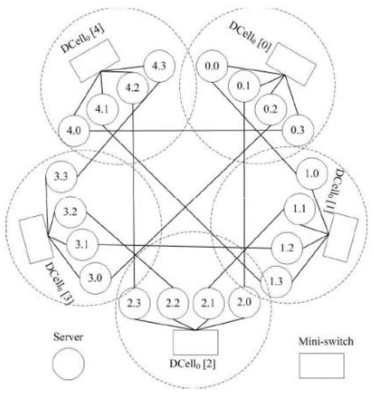
5.4 Study of Data Center Topologies

Data centers are growing in importance as they support and sustain the Internet-based applications including search (e.g., Google, Bing), video content hosting and distribution (e.g., YouTube, NetFlix), social networking (e.g., Facebook, Twitter), and large-scale computations (e.g., data mining, bioinformatics, indexing). For cost-effective deployment and maintenance of the infrastructure, it is essential to come up with efficient network management techniques.
SDN has shown major impacts in data center networks where large volume of data flows among several thousands of machines. We analyse the previous work to replicate Data Center Topologies in simulated environment [11], and provide a proof-of-concept to build these topologies on our hardware testbed. Some of the popular data centre topologies that can be studied by extending this testbed are:

a) Facebook "4 Post" Data Center Architecture [12]



b) D-Cell Topology [35]



Simulation on Mininet were generated and tested for these topologies. The proof-of-concept to setup these topologies on the physical testbed can be demonstrated by adding more network components.

6. Discussion and Conclusions

6.1 Challenges

Throughout the study on SDN components, we came across many challenges. The major ones are discussed below.

- ***Incompatibility of libvirt and Open vSwitch (OVS)***

Initial phase of the network research testbed, we have been trying to integrate OVS with Libvirt libraries and QEMU/KVM. It is difficult as one cannot define any configuration options through libvirt and need to set up fake bridges. Though Libvirt release 0.9.11 has added support for Open vSwitch, it is yet to be evaluated for successful operation. Strong OVS support built into libvirt is required for setting up virtual network topology.

- ***Incomplete topology monitoring due to legacy hardware switch***

Legacy switch used in the initial phase of the network testbed caused incomplete mapping of the network topology on the SDN controller interface. This difficulty helped us to closely understand the issues in incompatibility of operation of SDN and non-SDN environments. The complete topology was obtained when SDN switch was used, which could be controlled by the controller.

- ***Link Layer Discovery Protocol (LLDP) message inconsistency in SDN and legacy networks***

One of the major causes of incompatibility of SDN and non-SDN environments has been identified as inconsistency of Link Layer Discovery Protocol (LLDP) message. It can be overcome by developing a parser in the SDN enabled switches to understand the traditional LLP messages and sent it to the controller to accurately map the topology.

- ***Setting up of separate VLANs in hardware switch for OpenFlow or traditional functionality***

HP 3500yl 24G PoE requires setting up of management VLAN for Controller to switch operation and OpenFlow VLAN to connect SDN components. This is leads to unnecessary overhead of ethernet connections between the servers and switch for each VLAN.

- ***Slow performance of OpenFlow Parsing Agent in hardware switch***

The OpenFlow Parsing Agent in HP 3500 yl 24G PoE provided slow performance of around 2Mbps, whereas the traditional working of the switch has been observed around 1 Gbps. Such slack in performance in SDN mode is possibly due to less efficient memory to store the flow rules.

- ***Lack of DHCP server built into the switch***

The hardware switch doesn't support in-built DHCP server application like several network switches do. To overcome this, we configured our own DHCP server. This was connected to the management VLAN in the hardware switch to assign the IP to each server.

6.2 Contributions

The study allowed us to get in-depth hands-on understanding of the latest network technologies. We have been successful in configuring the three HP switches as well as servers into a small topology which allows us to provide proof-of-concept for a network research testbed in academia. We have been able to configure the switches in OpenFlow mode and connect them to a controller to analyse and evaluate the performance in SDN environment.
Some of the sub-tasks we have achieved in this process are described as follows.

- Analysis of scenarios where SDN is useful, discussed various challenges in this domain.
- Performance Analysis of Native vs Virtual Machine on KVM : System and Network
- Survey of various test beds in SDN and virtualized workload management
- Performance evaluation of our testbed using benchmark tools
- Setup of relevant experimental scenarios to be performed on the proposed testbed

6.3 Discussion

We need to resolve the incompatibility of LLDP messages in SDN components and legacy networks, in order to overcome the real-constraint of hybrid environment. One of the approach can be to setup a gateway between the SDN and non SDN domains so that they can communicate with each other. Another approach can be to incorporate capability in the SDN switches to parse the traditional LLDP messages.
The distributed operating system for SDN can be tested on this physical research testbed and evaluations for the proposed solutions like ONOS [37] is possible.
With the view of data center networking environment, optimising and controlling the virtual resource migration is critical. Improved control over live migration can significantly impact the efficiency of a data center in terms of management and service.
The accuracy can be improved based upon a combination of past statistics of per link characteristics and weighted measurements analysis, without imposing additional overhead. Furthermore, more experiments in a real environment (i.e. with real traffic) are needed to fully evaluate the proposed measurement approaches.
SDN technology enables us to monitor the network topology through a big picture view. The APIs (Northbound) from the controller can service the information about all of the network the network administrators. This information can lead to deployment of automated load distribution, multipath routing as well as security monitoring.
Network Function Virtualisation (NFV) comes hand in hand with SDN. Various aspects of network can be in virtual machines like Firewall, DHCP, DNS, Router(Quagga), etc. To enable their live migration and maintenance, the implementation of SDN technologies decides the efficacy of such network models. Further, one can investigate into the performance of container and hypervisor based network architectures in SDN. We can also look into performance enhancement of large clusters like MPI or Hadoop, by configuring SDN for their optimal performance.

7. References

1. Diego Kreutz et al : Software-Defined Networking: A Comprehensive Survey. Proceedings of the IEEE 103(1): 14-76 (2015)
2. Xia, W., Wen, Y., Foh, C. H., Niyato, D., and Xie, H. 2015. A Survey on Software-Defined Networking. IEEE Commun. Surv. Tutorials 17, 1, 27–51.
3. Berman, M., Chase, J. S., Landweber, L., Nakao, A., Ott, M., Raychaudhuri, D., Ricci, R., and Seskar, I. 2014. GENI. A federated testbed for innovative network experiments. Computer Networks 61, 5–23.
4. Salsano, S., Blefari-Melazzi, N., Detti, A., Morabito, G., and Veltri, L. 2013. Information centric networking over SDN and OpenFlow. Architectural aspects and experiments on the OFELIA testbed. Computer Networks 57, 16, 3207–3221.
5. Kim, J., Cha, B., Kim, J., Kim, N. L., Noh, G., Jang, Y., An, H. G., Park, H., Hong, J., Jang, D., Ko, T., Song, W.-C., Min, S., Lee, J., Kim, B., Cho, I., Kim, H.-S., and Kang, S.-M. 2013. OF@TEIN. An OpenFlow-enabled SDN Testbed over International SmartX Rack Sites. APAN Proceedings 36, 0, 17.
6. 'Corporation for Education Network Initiatives in California' (CENIC) Online:[https://www.bc.net/sites/www.bc.net/files/uploads/BCNET2015/Presentations/Regional_Advanced_Network_Organizations_Today_SEvans.pdf], last accessed: 20.10.2016
7. 'Research Infrastructure for large-Scale network Experiments' (RISE 2.0) Online:[https://www.apan.net/meetings/apan38/Sessions/FIT/apan38th_itoh.pdf], last accessed: 24.11.2016
8. Open Networking Foundation (ONF). [Online]. Available: https://www.opennetworking.org/, last accessed: 29.11.2016
9. "Software-defined networking: The new norm for networks," Palo Alto, CA, USA, White Paper, Apr. 2012. Online:https://www.opennetworking.org/images/stories/downloads/white-papers/wp-sdn-newnorm.pdf, last accessed: 14.11.2016.
10. Glenn Dasmalchi, Understanding SDN 'A Survey of SDN Technologies and Associated Use Cases', 2015
11. Raj Jain, Washington University in Saint Louis, CSE570S: Recent Advances in Networking - Data Center Virtualization, SDN, Big Data, Cloud Computing, Internet of Things (Fall 2013) Internet Document: http://www.cse.wustl.edu/~jain/cse570-13/ftp/m_03dct.pdf, last accessed: 29.11.2016
12. Nathan Farrington and Alexey Andreyev, Facebook, Inc., CA 94025, USA 'Facebook's Data Center Network Architecture' in Optical Interconnects Conference, 2013 IEEE on 5-8 May 2013
13. Integrating the Virtual Switching System in Cisco Data Center Infrastructure, Internet Document at http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/vssdc_integrate.html, Updated:Mar 20, 2009, last accessed: 29.11.2016
14. Rakesh Saha at IBM and Amit Agarwal at Google, 'SDN Approach to Large Scale Global Data Centers', Presentation at OpenNet Summit, Available online:
15. http://opennetsummit.org/archives/apr12/agarwal-saha-mon-datacenters.pdf, last accessed: 15.11.2016

16. http://www.networkworld.com/article/2905257/sdn/is-an-sdn-switch-a-new-form-of-a-firewall.html
17. Product manual of Lenovo System x3650 M4 server, Online:https://www.manualslib.com/manual/1187508/Lenovo-System-X3650-M4.html, last accessed: 29.11.2016
18. Open vSwitch Online:http://openvswitch.org/, last accessed: 21.11.2016
19. Tomas Hegr, Leos Bohac, Vojtech Uhlir, Petr Chulmsky, "OpenFlow Deployment and Concept Analysis", Information And Communication Technologies And Services Volume: 11, Number: 5, 2013
20. W.Braun and M Menth "Software Defined Networking Using OpenFlow: Protocols, Applications and Architecture Design Choices", Future Internet, 12 May 2014
21. OpenFlow Switch Specification: Version 1.1.0. https://www.opennetworking.org /images/stories/downloads/sdn-resources/specifications/OpenFlow/openflow-spec-v1.1.0.pdf. Accessed: 19/10/2016.
22. OpenFlow Switch Specification: Version 1.3.0. https://www. opennetworking.org /images/stories/downloads/sdn-resources/ onf-specifications/openflow/openflow-spec-v1.3.0.pdf. Accessed: 19/06/2015.
23. OpenFlow Switch Specification: Version 1.4.0. https://www. opennetworking.org /images/stories/downloads/sdn-resources/ onf -specifications /openflow /openflow-spec-v1.4.0.pdf. Accessed: 19/06/2015.
24. OpenFlow Switch Specification: Version 1.5.0. https://www. Open networking.org /images/stories/downloads/sdn-resources/onfspecifications/openflow/openflow-switch-v1.5.0.noipr.pdf. Accessed: 19/06/2015.
25. OpenFlow Concept, http://openflownetworking.blogspot.in/2016/04/openflow-concept.html, Accessed on 27.10.2016
26. Open vSwitch for Xenserver http://openvswitch.org/support/dist-docs-2.5/INSTALL.XenServer.md.html, Accessed on 11.10.2016
27. Open vSwitch for KVM http://openvswitch.org/support/dist-docs-2.5/INSTALL.KVM.md.html, Accessed on 14.10.2016
28. Lee Doyle, http://searchsdn.techtarget.com/answer/Does-software-defined-networking-require-special-SDN-switches
29. Niels L. M. van Adrichem," OpenNetMon: Network Monitoring in OpenFlow Software-Defined Networks", IEEE 2014
30. Ryu: Component-based Software Defined Networking Framework. https://osrg.github.io/ryu/. Accessed: 19/10/2016
31. Yao-Yu Yang, Chao-Tung Yang ,"Implementation of Network Traffic Monitor System with SDN" IEEE 39th Annual International Computers, Software & Applications Conference, 2015
32. Yuanhao Zhou1, Li Ruan, "A Method for Load Balancing based on Software Defined Network" Advanced Science and Technology Letters Vol.45 (CCA 2014), Pages.43-48
33. Domenico Giustiniano, "Optimizing TCP Performance in Multi-AP Residential Broadband Connections via Mini slot Access", Journal of Computer Networks and Communications 2013, pp.2-12

34. Qingwei Du and Huaidong Zhuang," OpenFlow-Based Dynamic Server Cluster Load Balancing with Measurement Support", Journal of Communications Vol. 10, No. 8, August 2015.
35. Brian Lebiednik, Aman Mangal, Niharika Tiwari, 'A Survey and Evaluation of Data Center Network Topologies', Journal article CoRR, vol. abs/1605.01701, 2016
36. Paul Goransson and Chuck Black. 2014. Software Defined Networks: A Comprehensive Approach(1st ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. pp. 66-68
37. Pooja. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, et al. ONOS: towards an open, distributed SDN OS. In Proceedings of the third workshop on hot topics in software defined networking, pages 1–6. ACM, 2015. Pages 572-578.
38. Configuration of HP 3500yl 24G PoE switch in OpenFlow http://pakiti.com/openflow-configuration-hp-provision-switches-hp-van-sdn-controller/
39. Gartner$_{®}$ report: "The Future of Data Center Network Switches Looks 'Brite' ", Available: http://www.gartner.com/document/2928517
40. Curt Brune, Doug Youd, JR Rivers, Nolan Leake, Roopa Prabhu, Cumulus Linux OS for bare metal switches. Available Online: https://cumulusnetworks.com/blog/linux-network-os/
41. Zhiping Yao, Jasmeet Bagga, Hany Morsey, 'Introducing Backpack: Our second-generation modular open switch', 8.11.2016,
42. Yuval Bachar, Adam Simpkins 'Introducing Wedge and FBOSS- the next steps toward a disaggregated network', 18.06.2014
43. Jason Long, Lagopus Switch, a high performance OpenFlow 1.3 switch. Avaliable online documentation: http://www.lagopus.org/ , 16.12.2016
44. Indigo Virtual Switch (IVS), open source virtual switch built on Indigo Framework, Online http://www.projectfloodlight.org/indigo-virtual-switch/, 16.12.2016