**Department Of Computer Science & Information Systems**


**A Report on**

# Investigations into problems and issues related to hybrid-SDN setup involving multi-domain scenarios

## Submitted By

**M Sharat Chandra (2014A7PS108P)**

## Under the supervision of

**Prof. Dr. Rahul Banerjee**

**In the partial fulfilment of the requirement of the course**

**CS F376: Design Oriented Project**

**Birla Institute of Technology & Science**

**Pilani**

# Acknowledgement

I would like to express my special gratitude to Professor Dr. Rahul Banerjee who gave me the opportunity to do a project focussed on solving challenging problems in the field of Software-Defined Networking. I would also like to thank Pranjal Ranjan sir who gave valuable inputs as and when needed. I would like to thank my colleague Ameesha Mittal for supporting and contributing to the setup of the SDN research test-bed. This project has greatly helped me in understanding the real world implementation of SDN which will help solve the challenges that exist in developing and deploying a fully functional hybrid networking systems of the future involving SDN.

**Table of Contents**

# 1. Introduction

Software-defined networking (SDN) is growing up to be a standout amongst the most encouraging solutions for future Networking. It decouples the control plane from the data plane and provides programmability for network application development by breaking the vertical integration which existed for years. In a general SDN scenario, the network is divided into many subdomains each under a different controller. The goal of this project is to investigate the issues which exist in a hybrid multi-vendor scenario where both open-source and proprietary SDN components exist.

# 2. Background

## 2.1 Software-Defined Networking

"Software-Defined Networking (SDN) is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services. The OpenFlow® protocol is a foundational element for building SDN solutions." [1]

### 2.1.1 The SDN Architecture

Every traditional network device has two planes. The first plane is the forwarding plane which forwards the data; therefore, it is also called as the data plane. The second one is the control plane, which performs the tasks which need intelligence in the network such as routing and topology discovery. Figure 1 given below depicts the idea of SDN which decouples the two aforementioned planes and makes the network intelligent, programmable and responsive one that can be controlled by a centralized controller.

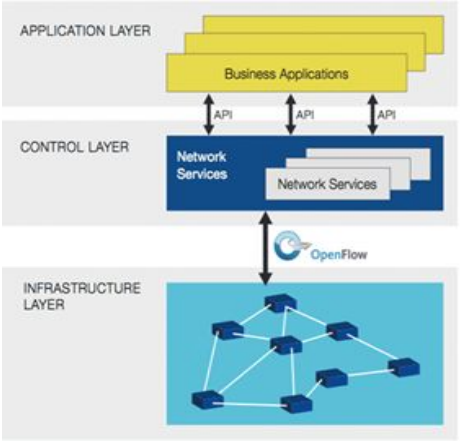


**Figure 1: Basic Architecture of SDN network**

Figure 1 illustrates the three layers present in the SDN architecture.

- **Infrastructure layer** comprises the forwarding network elements. Hence, it is also called the data plane or the forwarding plane. The data plane mainly performs data forwarding, as well as information monitoring and collection of statistics.
- **Control layer** is responsible for programming and managing the forwarding/data plane. It is also called the control plane. The control plane consists of software controllers which communicate with the forwarding plane through the standardized interfaces known as **southbound interfaces**.
- **Application layer** mainly consists of network applications which can facilitate in the introduction of new network features such as security, novel forwarding schemes, manageability and also assist in the configuration of the network. The controllers provide an abstracted and global view of the network to the application layer. The application layer uses that information to program the network in any way desired. The interface between the application layer and the control layer is known as the **northbound interface**.

### 2.1.2 Advantages of SDN

Software Defined Networking will change the way that system specialists and designers construct and work with their networks to accomplish their requirements. With the introduction of SDN, the manageability and programmability of networks has increased. Also, the networks have become open-source and non-proprietary. SDN provides more control of the networks, permits to tailor and to enhance their networks to lessen the general cost of maintaining the network. Some of the main SDN benefits can be summarized below [2].

- **Fast Service Deployment**
- **Automated Configuration**
- **Network Virtualization**
- **Reducing the Operational Expense**
- **Network Management Simplicity**

# 3. Problem Statement and Objectives

## 3.1 Problem Statement

**“Investigations into problems and issues related to hybrid-SDN setup involving multi-domain scenarios”**

## 3.2 Objectives

- Develop an understanding of hardware SDN components available
- Read and understand the manuals and related literature to help configure the test-bed
- Creation of research test-bed involving hybrid multi-vendor SDN Devices

- Creating a framework that would allow creation and execution of various test scenarios
- Design and implementation of mechanism that would help solve multi-domain interaction for generating a global topology

# 4. Creation of Research Testbed

This project is to be carried out in two phases to achieve the aim of creation of a multi-vendor hybrid test-bed. The first phase includes creating a fully functional test bed using just the devices from a single vendor: Hewlett-Packard®. The next phase intends to include Cisco's Non-ACI (Pure SDN) and ACI devices. Once the hybrid test bed is set up, various experimental scenarios can be designed and performed.

## 4.1 Phase I: Fully-functional Hardware test-bed involving uni-vendor SDN devices

The first phase of setting up the SDN research setup includes creation of a setup consisting of devices (SDN switches) of HP: HP 3500yl-24G J9310A series. The current setup includes 3 such devices and 3 lenovo servers. The hardware requirements of the test bed servers are mentioned in table 1 and the test-bed is pictorially depicted in figure 2.

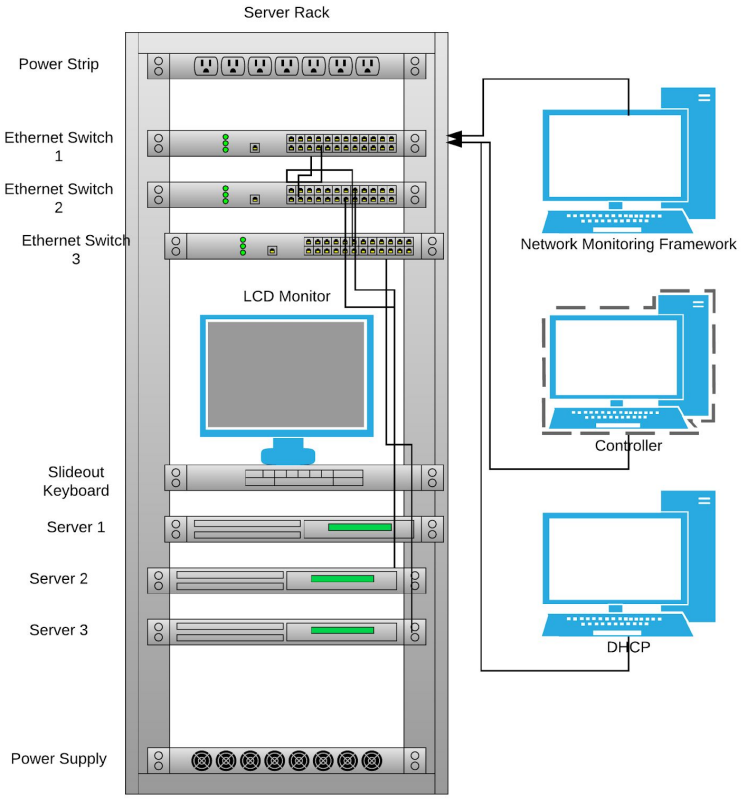


**Figure 2: Test-bed Phase I**

| Specification Type | Value |
|---|---|
| CPU Type | Xeon™ |
| Processor | 2 |
| Cores | 12 |
| CPU Frequency (GHz) | 2.5 GHz |
| Memory (GB) | 64 |
| DISK (GB) | 8000 |
| NIC (Mbit/s) | 1000 |
| Kernel | 4.4.0 |
| OS | Ubuntu 16.04 LTS |

**Table 1: Hardware Specifications of server class machines in test-bed**

### 4.1.1 Switch Setup and Configuration

The switches in HP 3500yl-24G J9310A series have a provision to configure them in OpenFlow enabled mode by creation of minimum two VLANs. One is the VLAN to which the SDN controller is connected to (this VLAN is non-SDN) and another one (can be more than one here) which is an openflow instance that the controller is controlling. So, the controller's view of the OpenFlow enabled switch is just this OpenFlow enabled VLAN.

Thus, this gives us a workaround to increase the no. of switches the controller sees virtually be creating more OpenFlow instance on a single switch by having different VLANs as their members. All the VLANs being talked about here are port-based VLANs. This particular way of configuring is due to the restrictions that HP devices have and not a general way of configuring SDN enabled hardware devices.

The controller used in the setup is the trial version of HP's own HP VAN SDN Controller. It provides functionalities such as Hybrid fallback mode, REST APIs to control the traffic and install flows and Topology viewer to intuitively have a look at the topology and quickly debug if anything goes wrong. To enable OpenFlow version 1.3.0, the firmware of the each switch has been upgraded to version 16.02 from 15.12.

### 4.1.2 Issues Faced

While setting up the phase 1 of the test-bed the issues faced are listed below:

- Lack of official documentation explaining the VLAN separation requirement.
- Incompatibilities between OpenFlow versions due to older firmware.
- Incompatibilities between Open-Source controllers and proprietary hardware switch.
- Broadcast storms created due to hybrid environment  provided by default in the switch.

Thankfully, with the help of various resources such as blogs[8] and video tutorials [9] we were able to fix all the issues and create a fully functional testbed consisting of HP 3500yl - J9310A switches and server class machines.

## 4.2 Phase II: Fully-functional Hardware test-bed involving Multi-vendor SDN devices

The goal of this phase is to include Cisco's Non-ACI (pure SDN) devices (CISCO Nexus 3172TQ) to the test-bed first and then include ACI devices(CISCO Nexus 9372TX NX-OS-Mode switches). To understand the possibility of hybrid test-bed coexisting together, we need to first get an insight into what ACI is and how is it different from open-source OpenFlow enabled pure SDN. For this a literature survey of the guides and datasheets has been done to investigate into the issues that might occur in such a multi-vendor scenario.

### 4.2.1 Differences between CISCO’s ACI and OpenFlow

Cisco's ACI is a tightly coupled, policy-driven infrastructure solution. Cisco depicts ACI as an all encompassing design, enabling programmability of network components through dynamic, application-driven network policy models and centralized automation. OpenFlow is a protocol, and foundational component of the software-defined networking (SDN) architecture, which decouples control of network and functions of forwarding. software-based controllers that maintain a global view of the network with centralizes network intelligence and control.

Currently, OpenFlow is the only open standards-based "southbound" protocol for communicating between an SDN controller and dataplane. It allows for the direct configuration of network hardware such as switches and routers making networks more programmable and satisfying the SDNs dynamic configurable network goal. Currently, APIC (Cisco’s policy-driven ACI Controller) can only push policies down to devices running Cisco's AVS, although Cisco is working  to distribute an ACI OpFlex agent for Open vSwitch.

A key feature of OpenFlow is to use flows to identify network traffic, based on pre-defined match rules that can be dynamically programmed using the SDN controller software. OpenFlow conveys forwarding information from a controller to the dataplane (a collection of switches), instructing the switches what to do when a flow matches the parameters in the match field. APIC, the Cisco policy controller, acts as a central repository for all policies, and manages and configures the policy on each of the switches in the ACI fabric. ACI also serves as a platform for third-party services such as advanced security, load balancing and monitoring.[7]

### 4.2.2 Considerations

Before moving on to creating such a hybrid multi-vendor scenario the considerations that need to be carefully studied are:

- Possibilities of co-existence of ACI and OpenFlow devices.
- Creation of framework for API Controller and HP Controller to communicate with each other.
- Global master controller to configure and install flows into devices of both the vendors.

# 5. Generation of global topology through multi-domain controller interaction

Any multi-vendor scenario creates a network which is split into multiple domains. Multiple domains created are under the control of different controllers. The  coexistence of such controllers and methodologies have been studied in my previous project. This work focuses

on generating a consistent global topological state  of the network that is divided into multiple sub-domains under separate controllers.

## 5.1 Topology Discovery Mechanism in SDN

The topology discovery mechanism[10] in SDN uses a special protocol called Link Layer Discovery Protocol (LLDP). The SDN controller periodically constructs and sends these messages to the data plane and requests the switches to flood them. The received messages by neighbouring switches are then sent back to the controller using packet-in messages. Using such messages the controller maintains information about the links and hence the topology of the network.

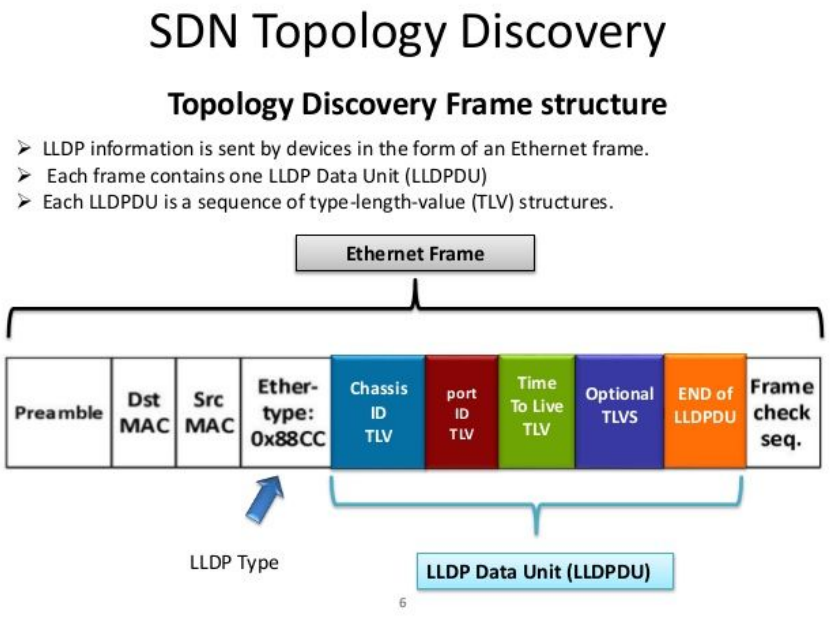


**Figure 2: Topology Discovery Frame Structure**

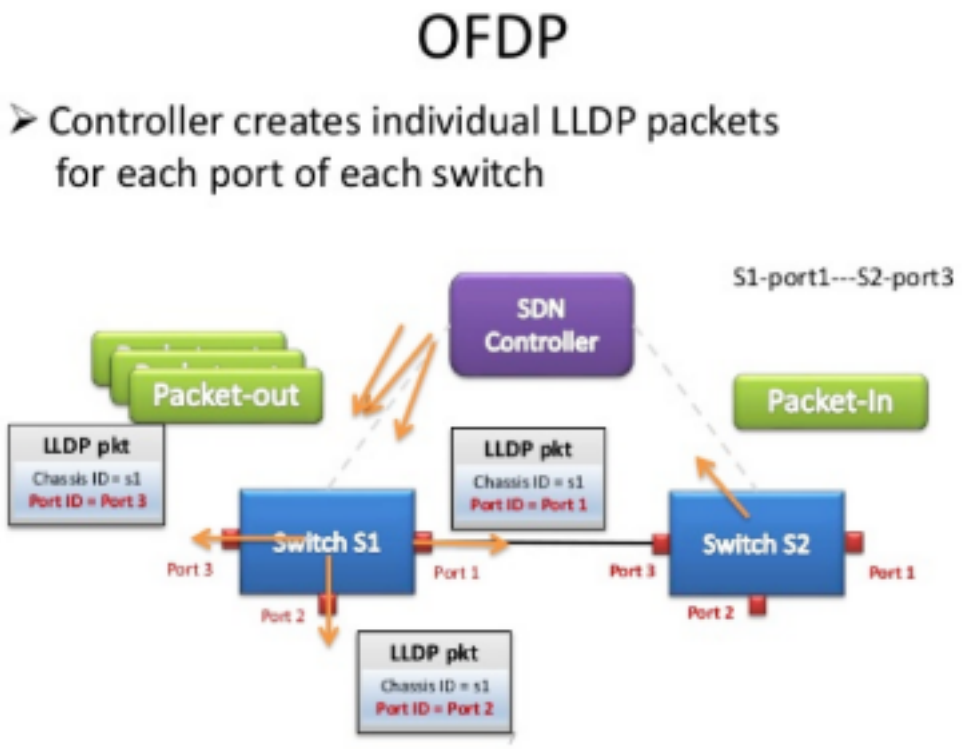


**Figure 3: Depiction of LLDP control messages**

## 5.2 Patching the controllers

Each controller in the multi-domain network uses LLDP messages to learn information about the switches under its domain. All of this information needs to be aggregated to generate the global topology. A key point to note here is that, the switches which lie at the boundary will be receiving the LLDP packets from the neighbouring switches which may be part of a different domain. The default implementation in a controller is to inspect the packet and drop it if it doesn't come from the switches that it is under control of. Clearly, this needs to be patched as otherwise the information about links connecting domains cannot be captured. Once, we modify the controllers to add the link to their topology, there is a need to fill the gap of information. Given that the datapath ids of the datapath hardware are unique in nature, the foreign switch(to the domain under consideration) can be uniquely identified and a link is formed from the foreign switch(source) to the boundary switch (destination). In a similar fashion, it can be noticed that the reverse link will be similarly identified by the controller of the other domain. Thus, through the minor patch, the controllers can now identify half of the neighbouring links.

## 5.3 Generating the Global topology

The global topology can now be generated using a custom script that collects all the information from different controllers and aggregates them to create a stable network global topology state. This script uses the REST API interfaces provided by the controllers which given information about the dataplane switches that they are a controlling and also the links between them. Due to the patch that we've made the links now also include inter-domain links.
Thus, intra and inter domain links put together produce the global topology from different controllers thereby now enabling multiple applications to utilize this information to say run a global routing algorithms, load balancing algorithms or to monitor the network as a whole.

# 6. Conclusion

This work intends to bring out a working testbed setup of univendor (Hewlett-Packard) and seeks to investigate into issues related to deploying a multi-vendor SDN testbed. Once such a testbed is established with multiple domains, this work also develops a design and implementation of mechanism that would help solve multi-domain interaction for generating a global topology.

# 7. References


[1] Definition of Software-Defined Networking ONF.
URL:https://www.opennetworking.org/sdn-resources/sdn-definition
Last Accessed: 18/03/2017
[2] N. Feamster, J. Rexford and E. Zegura "The Road to SDN", ACM, December 30, 2013.
[3] HP 3500yl J9310A Switch Specification Document
URL: https://www.hpe.com/h20195/v2/GetPDF.aspx/4AA2-6758ENW.pdf
Last Accessed: 19/03/2017
[4] Management and Configuration Guide for HP 3500yl J9310A
URL:ftp://ftp.hp.com/pub/networking/software/3500_5400_6200_MgmtConfigGde-July2006-59913826.pdf Last Accessed: 19/03/2017
[5] Is ACI Really SDN? One Point of View to Clarify the Conversation
URL:http://blogs.cisco.com/datacenter/is-aci-really-sdn-one-point-of-view-to-clarify-the-conversation Last Accessed: 19/03/2017
[6] CISCO Nexus 3172TQ Specifications
URL: www.cisco.com/c/en/us/support/switches/nexus-3172tq-switch/model.html
Last Accessed: 19/03/2017
[7] Understanding OpenFlow, VXLAN and Cisco's ACI
URL:http://www.networkcomputing.com/networking/understanding-openflow-vxlan-and-cisco-aci/551182896 Last Accessed: 19/03/2017
[8] David Bombal's Blog
URL: http://pakiti.com/

Last Accessed: 19/03/2017
[9] Youtube Video on HP switch Configuration and HP VAN SDN Controller
URL: https://www.youtube.com/watch?v=2HYI4qwh0XE
Last Accessed: 19/03/2017
[10] Pakzad, Farzaneh, et al. "Efficient topology discovery in software defined networks." Signal Processing and Communication Systems (ICSPCS), 2014 8th International Conference on. IEEE, 2014.