A Software Defined Network Implementation Using Floodlight and Open vSwitch

California State University DOMINGUEZ HILLS

Carlos Ontiveros, Bin Tang, Mohsen Beheshti

contiveros2@toromail.csudh.edu, btang, mbeheshti@csudh.edu

Computer Science Department, California State University Dominguez Hills

Abstract

Software defined networks (SDN) are not entirely new to the computer industry but technical limitations and past policies delayed and prolonged its wide acceptance in the industry. This past decade has seen a rebirth to the possibilities of virtualizing network functions in computer networks. SDN can create simpler network architectures, improve security, and efficiency. Network traffic bottlenecks arise when traffic must pass through middleboxes such as routers and firewalls. Middleboxes have limited capacities that may result in bottlenecks. SDN offer the ability to programmatically manage large networks. In this poster we show how Open vSwitch has been implemented on a physical testbed and connected to the Floodlight SDN controller which is used to programmatically reroute network traffic to alieve bottlenecks and increase performance and efficiency.

Introduction

Traditionally, computer networks were created using physical devices such as switches, firewalls, routers, and plenty of wires in order to be able to connect them all together. Middleboxes are network devices such as firewalls and switches. They perform network functions and have become virtualized in order to increase flexibility and ease management. Switches range in power and purpose from ordinary home usage to powerful industrial standards. Replacing or upgrading physical middleboxes is very challenging and costly. Software defined switches or virtual switches provide greater flexibility. Virtual switches are cross platform compatible and share common semantics which means that network administrators do not have to worry about learning new semantics from contracting a new vendor or upgrading their hardware. Software defined networks (SDN) offer the ability to programmatically manage large networks and create new features for old hardware devices.

Fat-Tree Topology

In Figure 1, a k-ary fat-tree is shown with k = 4, where k is the number of ports of each switch. There are three layers of switches: edge switch, aggregation switch and core switch. This fat-tree supports k3/4 physical machines but in this experiment we only built the end points. All are runnig Ubuntu 16.04 LTS and they are labeled with yellow, blue, and red boxes. We used 2 End Point servers to virtualize the aggregation and edge switches. We used one Core server to virtualize the core switches, SDN controller, and middle boxes.

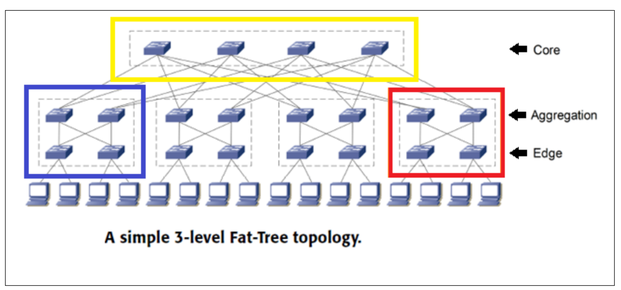


Figure 1

Hardware Resources

- 3 Servers *(See Figure 2)*
- Server A: Connects to Server B and Server C
  - Memory: 256GB (16x16GB)
  - CPU: 64x AMD Opteron(tm) Processor 6380 (4x16 Core CPU)
  - Disk: 1TB
- Server A & B Connect Aggregation and Edge Switches
  - Memory: 32GB
  - CPU: Intel® Xeon(R) CPU E5-2623 v3 @ 3.00GHz × 16
  - Disk: 1TB
  - Graphics: Gallium 0.4 on NVD9

Figure 2

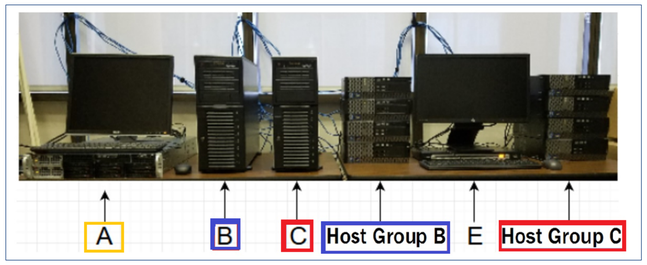
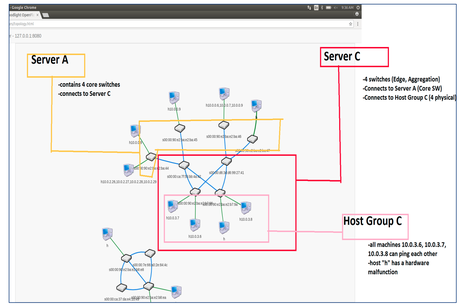


Figure 3



Software Defined Network

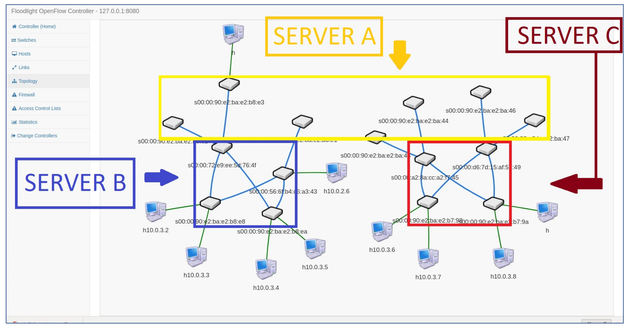
SDN Environment:

- OS: Ubuntu 16.04 LTS
- Network: OpenFlow 1.3
- Development: Java 8 64-Bit

Controller and Switches:

- Controller: Floodlight 1.2 *(See Figure 3)*
- Virtual Switch: Open vSwitch 2.5 *(See Figure 4)*

Figure 4



Conclusion

This project implemented Open vSwitch to connect eight host machines in CSUDH CS Department, to demonstrate the feasibility of software-defined networks and the SDN controllers. The Floodlight SDN controller is used to programmatically reroute network traffic to alieve bottlenecks and increase performance and efficiency. Open vSwitch has been implemented on physical testbed but the algorithms have not been tested on it.

Future Work

- Use the simulation tool Mininet to implement traffic rerouting
- For future work, we will finish implementing all algorithms onto the physical testbed and then compare the performances versus the simulated environment on Mininet.

Acknowledgements

This research is funded in part through the National Science Foundation (NSF) under Grant No.1551221.