SDN as an alternative to MPLS branch office connectivity

by

Alan Donagan & Kevin Moore

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

Alan Donagan — Date: 4-18-2016

Kevin Moore — Date: 4-18-2016

Jim Scott, Faculty Advisor — Date: 4-17-2016

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2016

Contents

Figures

Tables

Abstract

In this project, we explored Software Defined Networking (SDN) and looked into some of the real world uses for this emerging technology. By utilizing the OpenFlow protocol and a single Mininet server, we were able to create our own virtual networks in the UC Sandbox. Subsequently, we used OpenDayLight SDN Controllers to manage our switches and network traffic. Once we started experimenting with the practical uses of SDN it became apparent that we could use this as an alternative to a traditional MPLS (Multiprotocol Label Switching) connection. We built this Proof of Concept to demonstrate a very basic scenario showing a Software Defined Wide Area Network (SD-WAN) solution being used to provide connectivity between a Main Office and a Branch Office location. This solution could be implemented at a fraction of the cost of a MPLS connection while providing greater throughput over a connection that could still be secured.

Introduction

This project is focused on exploring a cost effective solution to provide Wide Area Network (WAN) access to a distributed network without the cost of an MPLS network. The solution will provide software layer overtop the network layer to provide a connection to the WAN by way of a consumer level internet connection.

Problem

The cost of an MPLS network connection to a business's Wide Area Network (WAN) can cost between $300-600 per Megabyte per second (Mbps) per month (Gottlieb, A.). These costs can and do prevent companies from expanding or opening a temporary location.

Solution

Creating a Software Defined Network (SDN) solution to connect to the WAN from an Internet connection will reduce this sunk cost (money that is already spent or permanently lost - businessdictionary.com) from the previously mentioned $300-$600 per month to $25-$200 per month based on bandwidth available at 3 Mbps to 150 Mbps. This solution will create a direct connection to the primary network via the internet connection for a fraction of the cost. Therefore opening the door to companies large and small to expand to a more cost effective distributed network.

Project Goals

Create a Software Defined Network solution to connect to the WAN from an Internet connection. This solution will create a direct connection to the primary network via the internet connection.

Overview

The Discussion section of this document covers in more detail the items introduced in this section. It contains the following subsections: Budget; Gantt chart; Problems Encountered; User Profile and Recommendations. The document closes with a list of Works Cited and Appendix.

Discussion

The idea of this project came from an online news article from *sdxcentral.com* about software Defined Networking. The thought of making a Wide Area Network costs more manageable than installing an entire MPLS network is very intriguing.

We used the OpenDayLight SDN controller to build our SD-WAN solution (Figure 1). We then used Mininet to setup a virtual network. With this setup, we're able to configure a network environment that simulates hosts and switches. We used Python to build scripts that simulate two network environments. One environment was setup to represent the Main Office, and a smaller environment was setup as the Branch Office.

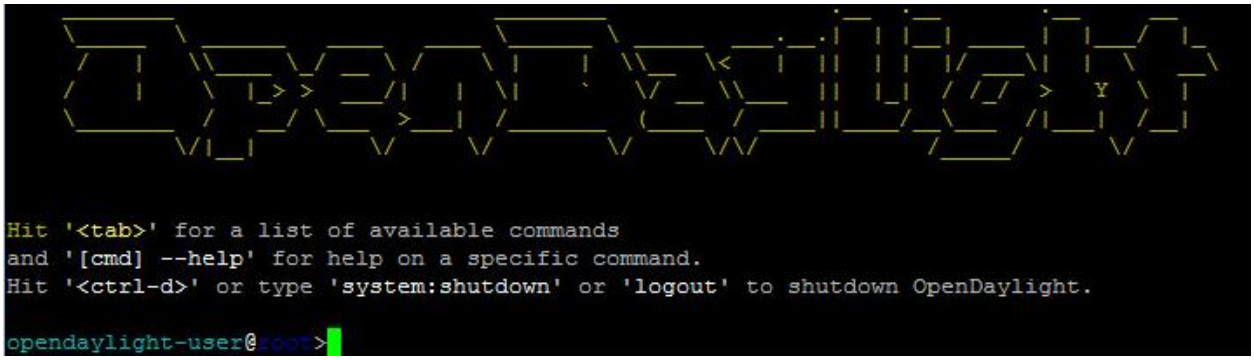


Figure 1: Running the OpenDayLight Controller

In Figure 2, you can see the Mininet server running our Python script and setting up the network environment. The script sets up 8 hosts and 5 switches and builds the necessary links between them. A pingall command is run to ensure all hosts can communicate.

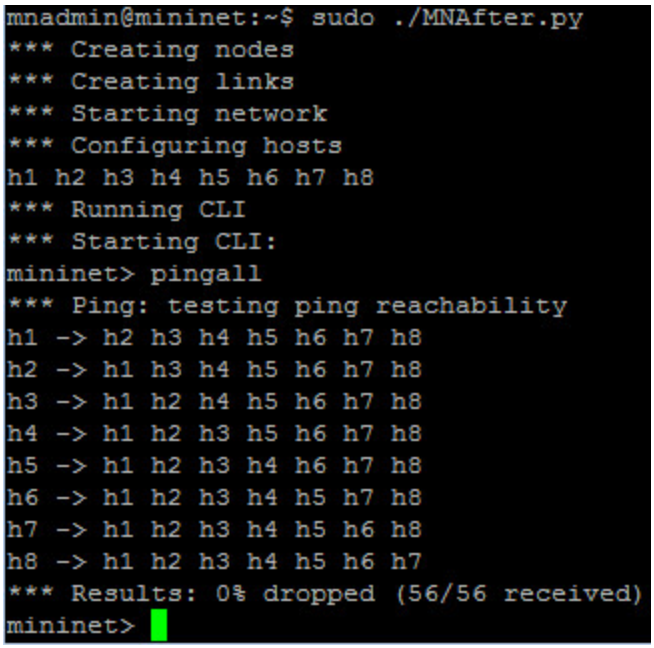


Figure 2: Running a Python script on the Mininet server to build the network environment.

Once set up, you can see the environment in ODL and begin to work with it as seen in Figure 3. This is where admins can start adding flows and managing connections using the software based management tool, allowing him to control traffic based on IP and port. There is a link between the switches openflow5 and openflow1. This is the link that joins the 2 networks (Main Office to the Branch Office) and can be managed by ODL. We used a very basic setup to demonstrate our Proof of Concept so in a real world application, the environments would have routers in place and static IPs provided by their ISP. With the static IPs, ODL can reach the routers over an internet connection and network administrators can manage traffic to allow branch office connectivity. Figure 4 shows a network diagram of this configuration and the ODL server communication to both networks.

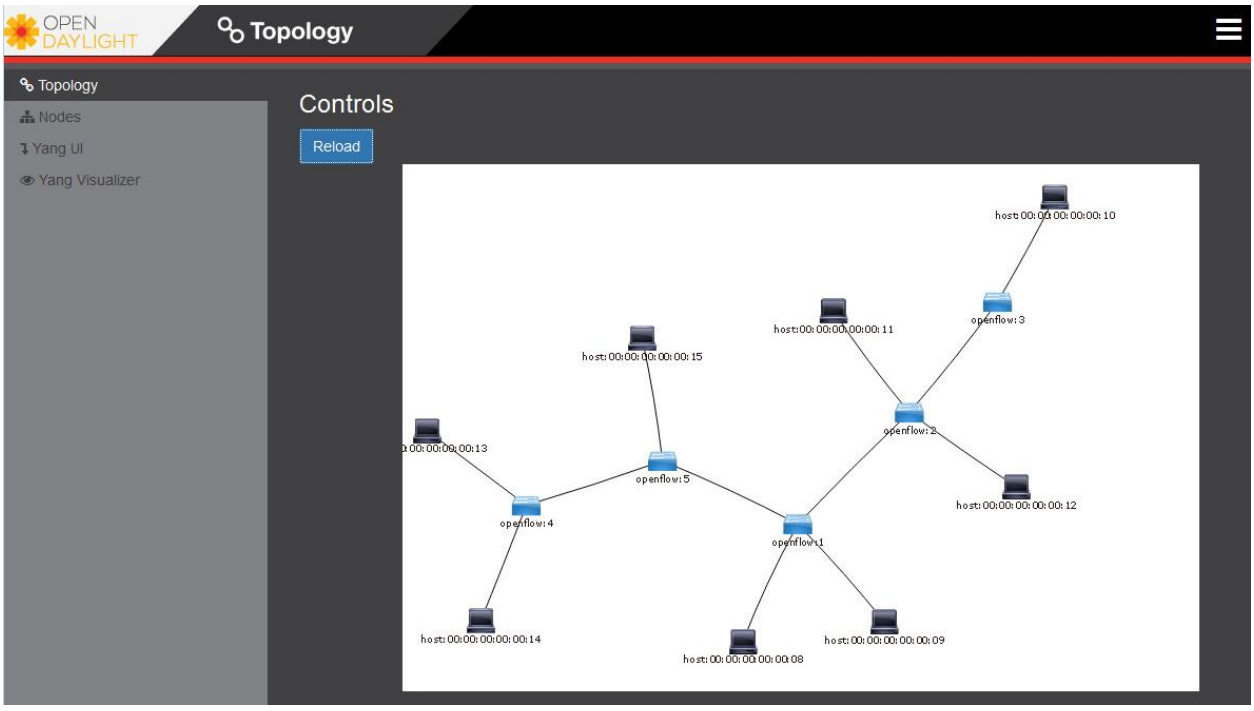


Figure 3: The Mininet network as seen in ODL

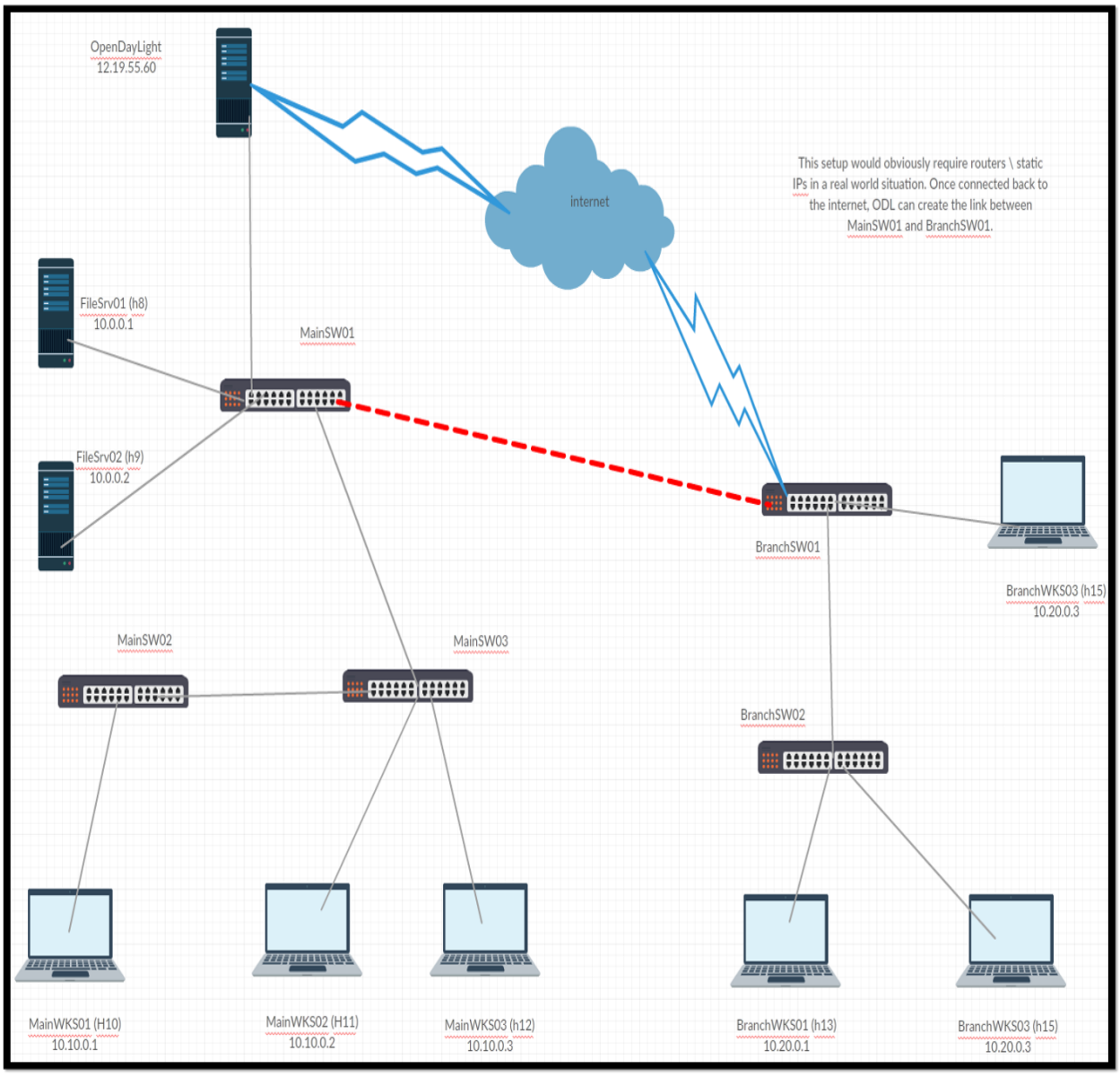


Figure 4: A network diagram of the Main Office and Branch Office locations.

Using the Yang UI, Figure 5, administrators can work with an MD-SAL Datastore and the API calls. This is where the more advanced features can be configured allowing total control of the network environment.

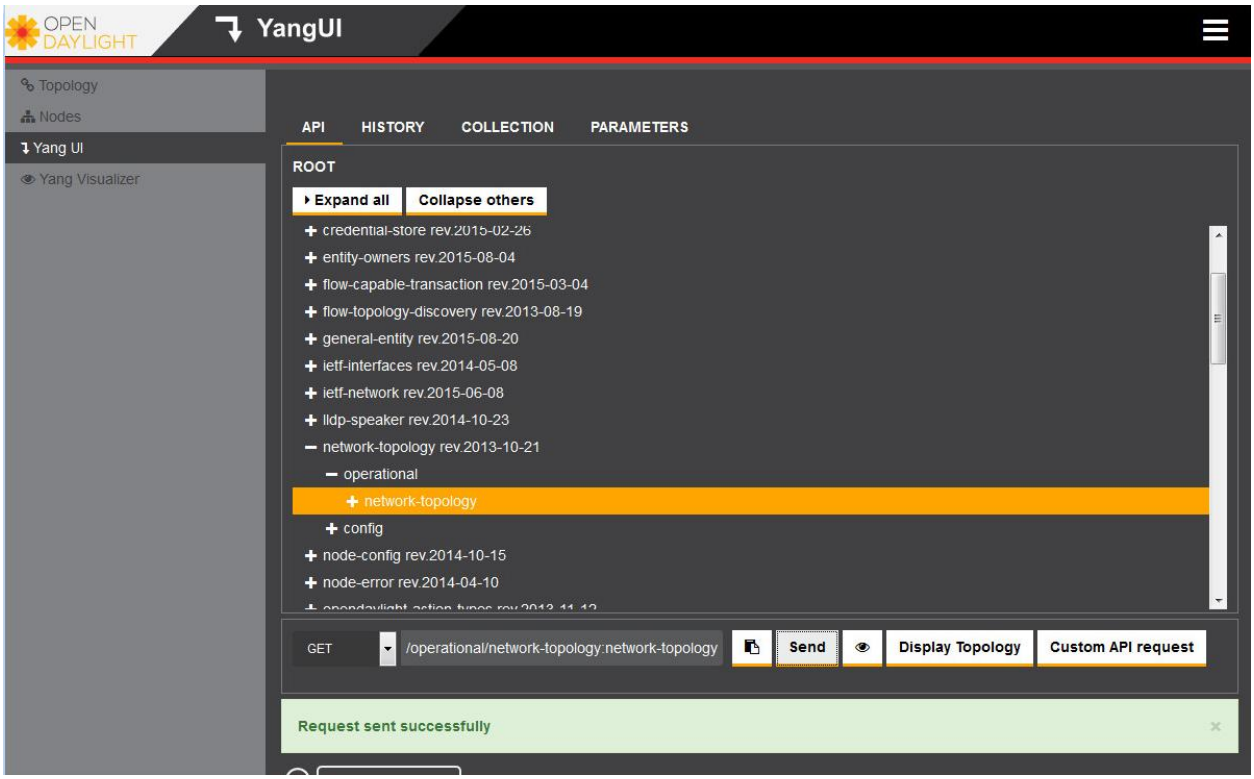


Figure 5: Using Yang UI to work with the API Calls

One of the additional features we added was the Web Console, Figure 6, which provides more of a GUI to the Karaf container where the OpenDayLight instance is installed. This GUI gives easier access to manage things like features and OSGi bundles. In this figure, you can see we only have 67 features installed out of the 611 that are available. You can also see the OpenDayLight server is using the latest version, Beryllium, which is a Production ready version that ODL released on February 22, 2016 (Lovato, J.).

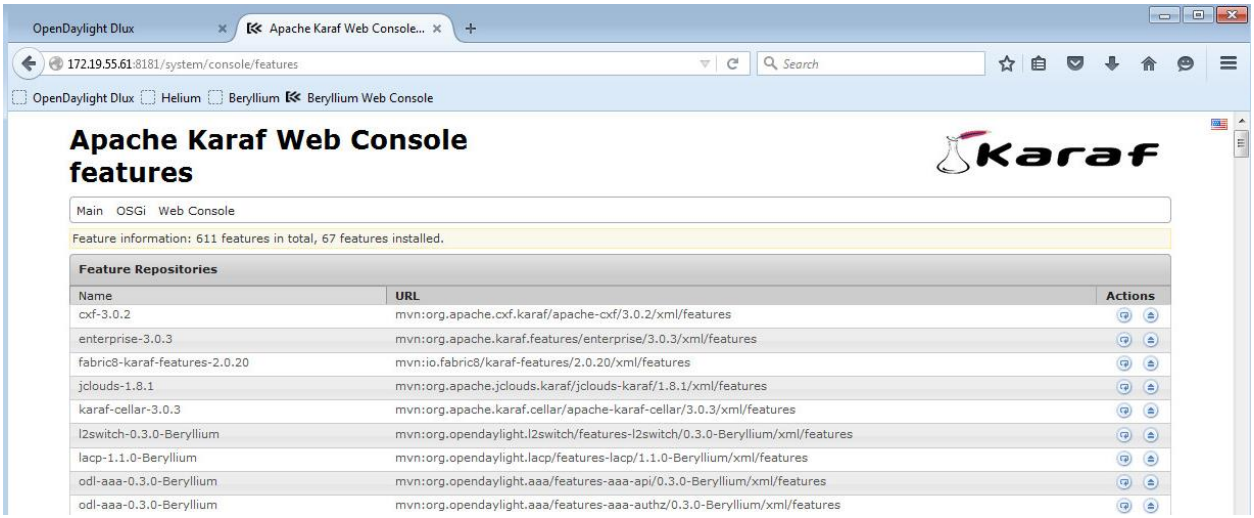


Figure 6: Using the Web Console to work with features

Budget

The budget for this POC is $0 as a result of using Open Source Software and building our test environment in the UC "Sandbox" (a virtual lab built on VMWare vCloud Director). A true budget is somewhat difficult to estimate due to several variables. For the initial project, a lot of time is invested in researching and building a demo SDN (Software Defined Network) environment. This time would be reduced when implementing the system in an actual environment. There will still be costs with analyzing the existing network, building and testing the correct solution, then deploying it.

The numbers in Table 1 show these costs and include a value for the cost of our research if this were done outside of an academic environment. The Real World example is based on a two location setup connecting a branch office to a main office. The solution uses the same OpenDaylight SDN Controller which is currently a leader in this technology. The hardware

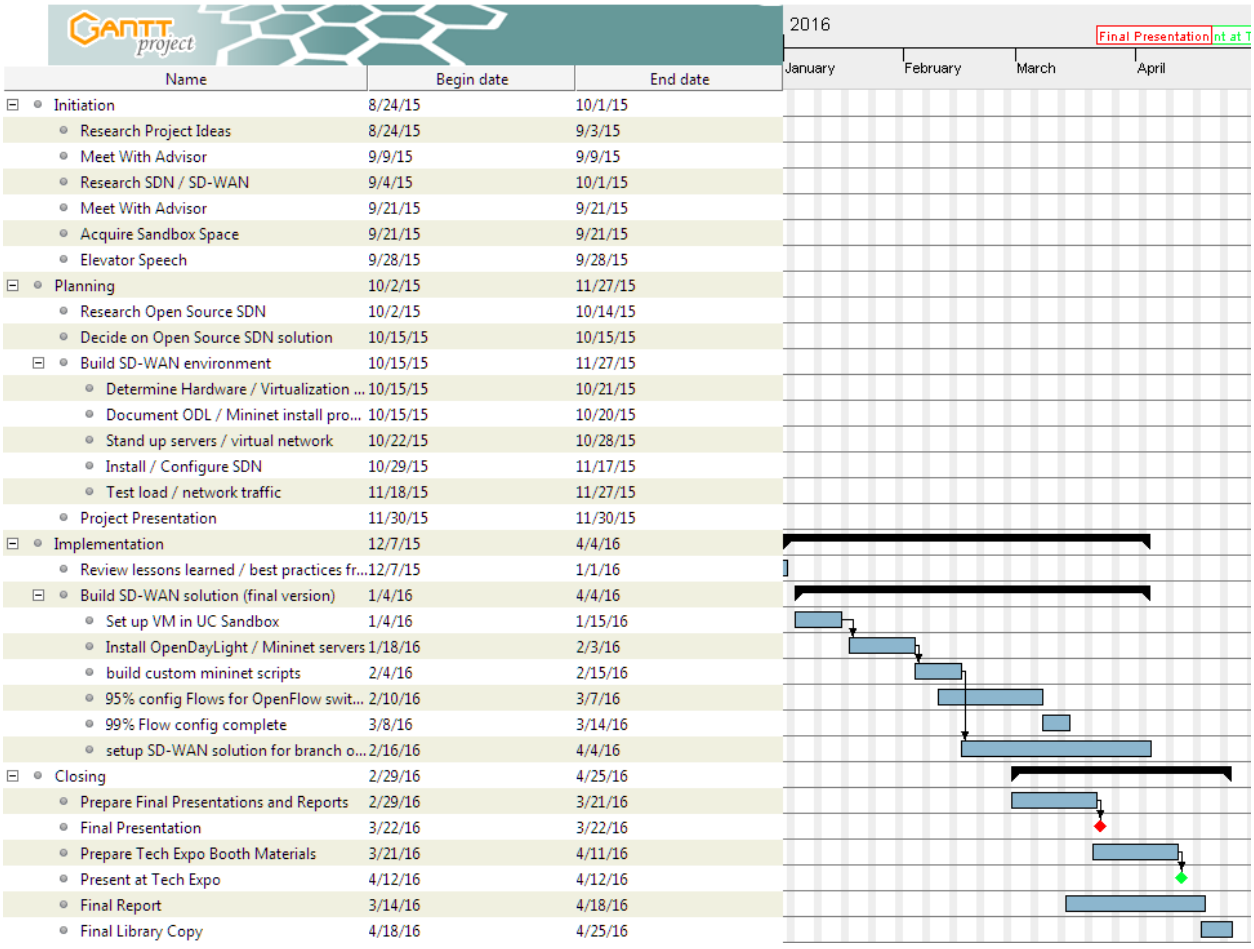
estimate is based on 2 servers with minimum specs. These costs would obviously vary based on the size and scope of the implementation.

	POC	Real World Cost
Research Costs ($75 / hr)	$18,750	$3,000
Labor ($75 / hr)	$15,000	$7,500
Software	$0	$0
Hardware	$0	$4,000
Total	$33,750	$14,500

Table 1: Estimated Costs for POC and Real World Application

Schedule

The Gantt chart, Figure 7, below shows the timeline for this project. This includes researching the project, acquiring the needed resources, choosing the best SDN solution, and building it out for the final demonstration.



Name	Begin date	End date
Initiation	8/24/15	10/1/15
Research Project Ideas	8/24/15	9/3/15
Meet With Advisor	9/9/15	9/9/15
Research SDN / SD-WAN	9/4/15	10/1/15
Meet With Advisor	9/21/15	9/21/15
Acquire Sandbox Space	9/21/15	9/21/15
Elevator Speech	9/28/15	9/28/15
Planning	10/2/15	11/27/15
Research Open Source SDN	10/2/15	10/14/15
Decide on Open Source SDN solution	10/15/15	10/15/15
Build SD-WAN environment	10/15/15	11/27/15
Determine Hardware / Virtualization ...	10/15/15	10/21/15
Document ODL / Mininet install pro...	10/15/15	10/20/15
Stand up servers / virtual network	10/22/15	10/28/15
Install / Configure SDN	10/29/15	11/17/15
Test load / network traffic	11/18/15	11/27/15
Project Presentation	11/30/15	11/30/15
Implementation	12/7/15	4/4/16
Review lessons learned / best practices fr...	12/7/15	1/1/16
Build SD-WAN solution (final version)	1/4/16	4/4/16
Set up VM in UC Sandbox	1/4/16	1/15/16
Install OpenDayLight / Mininet servers	1/18/16	2/3/16
build custom mininet scripts	2/4/16	2/15/16
95% config Flows for OpenFlow swit...	2/10/16	3/7/16
99% Flow config complete	3/8/16	3/14/16
setup SD-WAN solution for branch o...	2/16/16	4/4/16
Closing	2/29/16	4/25/16
Prepare Final Presentations and Reports	2/29/16	3/21/16
Final Presentation	3/22/16	3/22/16
Prepare Tech Expo Booth Materials	3/21/16	4/11/16
Present at Tech Expo	4/12/16	4/12/16
Final Report	3/14/16	4/18/16
Final Library Copy	4/18/16	4/25/16

Figure 7: Gantt chart

Problems encountered

We have been testing several solutions, but this is a new technology and multiple solutions are not always available for such cases. The testing of these solutions is being done on the UC "sandbox" where we are limited to five hosts making testing not all that encompassing. To get around this we will be using a Mininet server to create a virtual network. We can then manage the virtual network using the OpenDayLight (ODL) controller.

There were also issues getting OpenDayLight installed. After installing it on the server and running the script, authentication using the default admin account did not work. After trying to install various versions and repeatedly getting the same issue, we turned to the ODL community. This is a common issue and appears to be mostly related to the order in which the ODL features are installed. We were finally able to get it installed with authentication on Ubuntu Desktop. Once this was confirmed to be working, all steps were documented and a final version was installed on Ubuntu Server (See Appendix A: Installing OpenDayLight).

We decided to use an Open Source SDN solution for this POC to keep costs minimal. After the initial research, we narrowed our choices down to FloodLight and OpenDayLight. While the FloodLight installation was much simpler, the solution didn't seem to be as robust and had limited support. OpenDayLight is one of the leaders in this new technology and is even used by Cisco who has been heavily involved with the development (Duffy, J.). There is more support for this controller and a wide array of tools and applications that are currently available. For these reasons, we decided to use ODL.

After working with ODL for several months, a new version (Beryllium) was released on Feb 22, 2016. Beryllium is a production ready version, is more stable, and has better documentation. It was decided to stand up another Ubuntu server and install Beryllium to take advantage of the newest release and support documents.

Testing

1. Demonstrator will need SSH access to Ubuntu and Mininet servers from Windows 7 demo machine
    a. Ensure successful SSH connection using PuTTy
    b. Demonstrator should not receive warning prompt for the certificate
    c. Demonstrator should then receive a login prompt from the server
    d. Successful login should put the demonstrator at a Linux command prompt
2. Using PuTTy to connect to each OpenDayLight server, the demonstrator will navigate to OpenDayLight directorie and launch ./bin/karaf
    a. Using PuTTy to connect to each OpenDayLight server, demonstrator should be able to log in using odlviewer / Letme1n!
    b. Demonstrator can navigate to /distributions/karaf/target/assembly
    c. Running ./bin/karaf should launch ODL and display the 'OpenDayLight' logo, leaving the demonstrator as the OpenDayLight prompt
3. Using the Firefox browser on the Windows 7 computer, the demonstrator will connect to each OpenDayLight DLUX interface
    a. Both servers should be bookmarked on the Bookmark toolbar and labeled as 'OpenDayLight' and 'OpenDayLight2'
    b. Demonstrator should be able to connect to each server by clicking on the Bookmark

c. Demonstrator should be able to login with the default admin / admin credentials

d. The DLUX web interface should display a blank topology

4. Connect to Mininet server and import first network script

    a. Using PuTTy to connect to the Mininet server, user should be able to log in with mnadmin / Letme1n!

    b. Launch script by running <insert command name here> which should start the first network configuration

    c. Run a 'pingall' command. Each host should ping every other host it can reach

    d. Using the DLUX web interface, reload the topology on OpenDayLight1. Confirm the correct network

5. Use WireShark and show successful data traffic

    a. Launch WireShark tool from the Windows client

    b. <insert steps to generate traffic>

    c. <insert steps to look for specific traffic and show expected data flow>

6. Connect to Mininet server and end the network config, clean up, then import the next config

    a. Using PuTTy to connect to the Mininet server, user should be able to log in with mnadmin / Letme1n!

    b. End the currently running Mininet configuration by typing 'exit'and clean config by running mn -c

    c. Launch script by running <insert command name here> which should start the first network configuration

    d. Run a 'pingall' command. Each host should ping every other host it can reach

    e. Using the DLUX web interface, reload the topology on OpenDayLight1. Confirm the correct network

f. Using the DLUX web interface, reload the topology on OpenDayLight2. Confirm the correct network

7. Use WireShark and show successful data traffic
    a. Launch WireShark tool from the Windows client
    b. <insert steps to generate traffic>
    c. <insert steps to look for specific traffic and show expected data flow>

User Profile

Network Engineers	Network Engineers and Sys Admins should have used various interfaces including Linux based operating systems, switch and router CLIs, and web based management consoles
System Administrators	System Administrators will have experience with Linux based servers and application installations. Network engineers should be familiar with hardware based network configurations and working with network component CLIs
End Users	As a means to connect to the network

Figure 8: User Profile

Task Experience:

Network Engineers should have a strong understanding of networking concepts, and the current network requirements and configuration. They should be able to configure the necessary components including VLANs, port security, redundancy, etc. Sys Admins should have Linux installation and administration experience. Basic knowledge of REST programming and API calls may eventually be necessary to take advantage of some of the advanced features. It would be preferable if both had some knowledge of tools such as Mininet which will assist in

the initial configuration and testing. A Sys Admin, IT Support, or even an End User could assist with plugging the server in once deployed to a site.

Frequency of Use:

The majority of use will be during the implementation. A lot of work is required to setup and configure the SD-WAN. Once this setup is complete, it should be used in limited situations consisting of network maintenance tasks. The solution will provide network connectivity so all users will use it in that context.

Key Interface Design Requirements that the Profile Suggests:

The SDN solution is software based but will rely primarily on CLI. There will be no customizing the web interface since this uses prebuilt Open Source software.

Conclusion

Software Defined Networking (SDN) is still in its infancy when it comes to changing the way networks function. SDN is a new and exciting way to make the network run while using preexisting consumer internet. A major player in the game will decide how this technology will be used in the years to come such as Cisco. Having the ability to control your network across the cloud will greatly benefit small companies looking to expand and to the enterprise corporations looking to trip their IT expenses. Look out for SDN to make its way into networking and software circles in the upcoming years.

References

Duffy, Jim. “Cisco reveals OpenFlow SDN killer.” *NetworkWorld.com*, last modified April 2, 2014 http://www.networkworld.com/article/2175716/lan-wan/cisco-reveals-openflow-sdn-killer.html

Gottlieb, Andy. “Why does MPLS cost so much more than Internet connectivity?” *NetworkWorld.com*, last modified April 19, 2012 http://www.networkworld.com/article/2222196/cisco-subnet/why-does-mpls-cost-so-much-more-than-internet-connectivity-.html

Lovato, Jill. “OpenDaylight Helps Organizations Solve Key Network Challenges with Fourth Platform Release.” *Opendaylight.org, last modified March 28, 2*016 https://www.opendaylight.org/news/foundation-news/2016/02/opendaylight-helps-organizations-solve-key-network-challenges-fourth

“SDN-Software-Defined-Networking” *sdxcentral.com*, 2014 https://www.sdxcentral.com/flow/sdn-software-defined-networking

“Sunk Cost” *Business Dictionary, 2015* *http://www.businessdictionary.com/definition/sunk-cost.html*

Appendix A: Installing OpenDayLight

//Install JDK and Maven

sudo apt-get install openjdk-7-jdk
sudo mkdir -p /usr/local/apache-maven
wget http://ftp.wayne.edu/apache/maven/maven-3/3.3.3/binaries/apache-maven-3.3.3-bin.tar.gz
sudo mv apache-maven-3.3.3-bin.tar.gz /usr/local/apache-maven
sudo tar -xzvf /usr/local/apache-maven/apache-maven-3.3.3-bin.tar.gz -C /usr/local/apache-maven/
sudo update-alternatives --install /usr/bin/mvn mvn /usr/local/apache-maven/apache-maven-3.3.3/bin/mvn 1
sudo update-alternatives --config mvn
sudo apt-get install vim
vim ~/.bashrc

//add these lines to the end of /.bashrc
export M2_HOME=/usr/local/apache-maven/apache-maven-3.3.3
export MAVEN_OPTS="-Xms256m -Xmx512m"
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64

//build ODL
sudo apt-get install git
git clone https://github.com/opendaylight/integration.git
curl https://raw.githubusercontent.com/opendaylight/odlparent/master/settings.xml --create-dirs -o ~/.m2/settings.xml
cd integration
mvn clean install -DskipTests

//install OpenVSwitch
sudo apt-get install openvswitch-switch

//Run ODL
cd distributions/karaf/target/assembly/bin
./karaf -of13

//use ./karaf clean -of13 if it hangs

//install features IN THIS ORDER
feature:install odl-restconf odl-l2switch-switch odl-mdsal-apidocs odl-dlux-core

//connect to DLUX using a browser
http://<ODL Server IP>:8181/index.html
admin / admin

//after successful install
#feature:install odl-dlux-all (installs full DLUX features)
#feature:install odl-openflowplugin-flow-services-ui