Improving QoS in a Software-Defined Network

Capstone Research Paper

April 30, 2016

Derrick D'souza
Krishna Prabhu Sundharan
Savithru Lokanath
Vivek Mittal

Interdisciplinary Telecom Program
University of Colorado Boulder

Faculty Advisor:
Dr. Levi Perigo
Professor
Interdisciplinary Telecom Program
University of Colorado Boulder

Industry Advisor:
Rob Hagens
VP of IP Architecture
Zayo Group

Abstract - Internet Service Providers (ISP) and networking companies are apprehensive about implementing a Software-Defined Network (SDN) as a universal solution because of the lack of quality research done in the field. This paper focuses on Quality of Service (QoS) as an influential factor differentiating a service provider's perspective in choosing between a centralized SDN versus a distributed traditional routing network. An application prone to latency and jitter can severely impact its performance leading to a dismal customer experience. Most service providers today need a network with fast provisioning and quick support. SDN aims to provide this capability to the service provider with a valid exhibit of QoS. The proposed prototype of the SDN demonstrates comparable QoS performance to traditional networks. This paper finds ways to mitigate latency and suggests improvements in QoS using a custom built OpenFlow controller based on Floodlight. The networking tool iPerf is used to measure the QoS performance characteristics. The results indicate capital cost savings of 20% and operational cost savings close to 43% upon migration to a white box SDN from traditional networks.

Keywords - Software-Defined Network; Quality of Service, Floodlight; OpenFlow; Mininet; Open vSwitch; jitter

I. Introduction

SDN has been considered revolutionary in the networking industry. While the technology is widely poised to replace the traditional hardware-based networking infrastructure, the rate of adoption of SDN by ISPs and networking companies has not taken off as originally predicted [1]. To gain more insight into this problem, we conducted a survey among 50 engineers in top networking companies and the result of this survey is shown as a pie chart representation in Figure 1.

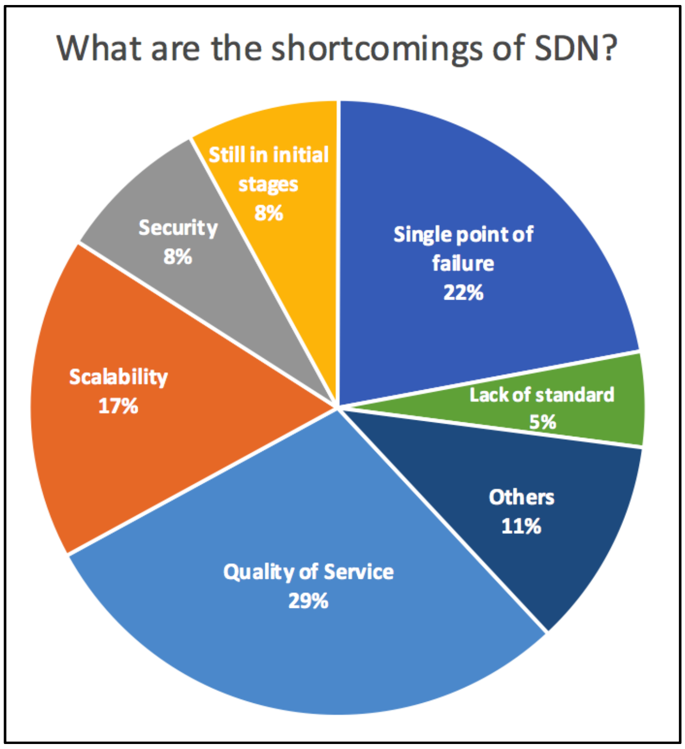


Figure 1. Challenges faced with implementation of SDN conducted among network engineers

The results indicate that 29% of respondents, the largest group, felt that QoS is the biggest obstacle to implementing SDN. This reason was followed by 22% of the engineers choosing a single point of failure, 17% selecting scalability and 8% choosing security as the primary concern with implementing SDN. About 8% also felt that SDN is in its initial phase, illustrating that it needs more time to become industry ready. In addition to this perception, 5% were concerned that there is no standard for governing SDN, and the remaining 11% had miscellaneous reasons. Overall, this study helped demonstrate the importance of further research into QoS in a SDN infrastructure.

II. Research Question And Problem Setting

Research question: How might QoS efficiency and efficacy be achieved when running on SDN rather than on traditional routing devices? Is the cost and effort involved to achieve this setting worth the technical risks of such a centralized network approach compared to traditional hardware-based networking?

Our study focuses on facilitating the decision-making process for service providers. This process must weigh benefits and disadvantages of centralized control of the network with lower capital and operating costs with distributed control over a traditional routing network that typically involves greater costs. Only a few service providers have slowly started testing these SDNs in their lab environments; however, they do not perceive the necessity to move these tests into a production environment soon. This hesitation might be the result of skepticism toward QoS in the customer base. This research hopes to solve this dilemma by providing a comparison of the improvements of QoS in a SDN versus traditional networks. Additionally, this study provides an initial analysis of the proposed SDN with graphical and analytical models to identify whether savings are available using the prototype developed for the analysis.

Given that research in QoS remains nascent, our proposal offers service providers flexibility and heightened efficiency through SDN. Our research hypothesizes that service providers can implement SDN to save Capital Expenditure (CapEx) and to reduce Operating Expenditure (OpEx). This proposal includes a demonstration using a SDN prototype and validation of QoS using standard networking tools.

Problem setting: To simulate a dynamic QoS setting in a traditional network, we configured multiple parameters on a vendor-to-vendor basis. Current methods configuring QoS are time-consuming and do not contain a mechanism for centralized intelligence to configure QoS based on changing network conditions. This issue is addressed by a SDN approach aiming to simplify the configuration process and to facilitate a uniform QoS setting across the network. Since QoS is a relatively new topic, documentation is scarce. As a result, ISPs are still apprehensive about implementing a SDN model in their network [2]. In our project, we have addressed this problem by providing a stable solution viable in the production environment. We have also achieved the primary goal of the SDN model - to reduce the CapEx/OpEx significantly through the utilization of open-source projects/tools.

To achieve these cost-savings, we first simulated the network using Mininet and Open vSwitch (OVS) connected to a custom built controller. After achieving the desired results, we implemented the same setting in a physical environment consisting of two Raspberry Pis, each running OVS and both connected to the custom controller; two end-hosts acting as Real-time Transport Protocol (RTP)/data server and client; and a Netgear switch connecting the Pis to the controller. Typically, the cost of setting up this network is approximately $385, which is as illustrated in Table 1.

Equipment	**Cost**
x86 Server (x1)	$200
Raspberry Pi (x2)	$100
Cables and Connectors	$50
Netgear Switch (x1)	$35
Custom Controller (x1)	$0
Ubuntu OS	$0

Table 1. Capital Expenditure

This is economical when compared to the cost of a traditional vendor network. After pushing in the QoS configurations through Representational State Transfer (REST) calls, the SDN controller makes intelligent decisions based on the changing network parameters such as bandwidth, delay, jitter, latency, and classification of traffic. These parameters are measured using standardized tools such as iPerf and Wireshark.

II. Research Sub-Problems

A. How do we demonstrate QoS on a traditional routing network model?

The QoS in a traditional networking model can be achieved based on different parameters. The first step is to recognize the best possible method to prioritize one flow by limiting the throughput of other flows. Cisco has recommended the use of traffic shaping and traffic policing to achieve the QoS functionality in an efficient and effective way [3]. The challenge is to understand the behavior and characteristics of applications that are prone to delay and are loss-sensitive. The QoS demonstration in a traditional networking model includes two traditional routers and two end hosts, of which one will act as a RTP/data server and the other acts as a client. Another important point is that traditional networking devices used in this experiment are production grade, whereas the devices that are used to test SDN performance are not because of lower processing power. Hence, the tests run on the traditional networking devices will be scaled down to match the peak performance throughput of the SDN devices.

B. How do we demonstrate QoS on a simulated SDN model?

The Floodlight controller has been predominantly used as the standard SDN controller by the industry, especially the major service providers. The Floodlight controller highlights the ease of setting up SDN with minimal dependencies and supports a broad range of virtual and physical OpenFlow switches. One challenge is to develop a working application using the Floodlight controller to demonstrate QoS. Floodlight has an open-source project to demonstrate the use of SDN Application Programmable Interfaces (APIs) and a Graphic User Interface (GUI). The source code for rate-limiting the interfaces using Floodlight is freely available and can be modified to demonstrate improved QoS [4]. The demo for SDN includes two Raspberry Pis running OVS, which are configured using the modified Floodlight controller. The current implementation uses Yet Another Next Generation (YANG) models with REST APIs. The existing APIs require modifications to incorporate QoS as an additional functionality. Tools for QoS testing are not well-defined or standardized. Although there are many licensed tools, to keep the costs low, this project uses open-source tools to conduct these tests. Many open-source tools are available such as those developed by Stanford University to measure QoS in various sections of the demo [5].

C. What technical aspects of the design must be addressed to improve QoS?

In a SDN, the concept of QoS is still new. Not many controllers support the QoS feature, and this deficiency has resulted in the applications being developed without many of the standard features required to implement a true QoS functionality. A software-controlled network is likely to encounter latency depending on how far the controller is from the physical devices [6]. A smaller network may ensure a better QoS, but realistically, this is not the case in an enterprise network. A design analysis of a distributed software network architecture with multiple controllers in a clustered environment is required before implementing QoS in SDN. Additionally, the interoperability in a hybrid network model (SDN + traditional network) is to be analyzed and improved since there is no standard present to implement QoS in a hybrid network model.

III. Literature Review

A. Measuring QoS

Network monitoring tools are crucial in a research project that deals with measuring QoS. Stanford University has developed a series of network monitoring tools that can be used to monitor various parameters of information that are transmitted over a network. These capabilities can be used to measure QoS [5]. In this paper, a list of tools called the perfSONAR-Performance Toolkit (pS-Performance Toolkit) is used for network monitoring and measurement of QoS. These tools can be loaded on the end devices such as the hosts to check the effectiveness of the imposed QoS [7].

B. QoS in Traditional Devices

The Cisco whitepaper "Quality of Service Networking" lists the different techniques to implement QoS for Cisco devices [8]. This whitepaper lists the various parameters that must be considered in delivering information at a designated QoS: 1) It classifies QoS through identification and marking techniques between end-to-end network elements, 2) Within a single network element consisting of queuing, scheduling, and traffic-shaping tools, and 3) Using policies and management functions to control end-to-end traffic across a network of many devices. This whitepaper mentions the service levels for defining QoS as best-effort service, differentiated service (soft QoS), and guaranteed service (hard QoS).

C. Floodlight Controller

Floodlight is an open-source SDN controller that uses OpenFlow as its southbound protocol to configure physical and virtual network elements. It also facilitates Apache-licensed features which enable an individual to use Floodlight for any purpose. An Asia-Pacific Advanced Network (APAN) whitepaper proposed multiple ways for implementing QoS in SDN. One way is to use Type of Service (ToS) bits and queuing packets, and the other way is through using Differentiated Services Code Point (DSCP) values on OVS [9].

D. QoS in Multimedia

Egilmez and Dane in their paper on QoS describe how OpenFlow is adapted to enable QoS routing to achieve dynamic optimization of routes for efficient multimedia delivery [10]. This paper discusses classifying traffic depending on its type for example data or video. The video packets are allowed to use a well-defined QoS path while the data packets take the shortest path as learned via routing. OpenQoS demonstrates dynamic routing on OpenFlow-enabled switches. This concept can be applied to any device running OVS; such as Raspberry Pi in our case.

E. Traffic Shaping using QoS

Voicu, in his presentation, discusses ways and techniques to implement QoS in OVS by traffic shaping [11]. OVS demonstrates substantial QoS support in the form of Ingress Classless Queuing Disciplines (qdisc), Hierarchical Fair-Service Curve (HFSC), and Hierarchical Token Bucket

(HTB). It supports dynamic bandwidth adjustment. This presentation provides a framework for the network tests and analysis to be performed in order to demonstrate QoS. The native support for policing in OVS makes this a good choice for describing QoS.

F. Capital Expense Savings

Naudts explains how SDN can save an organization's CapEx in his article [12]. Since SDN is vendor neutral, it is not necessary to rely on a single vendor to setup your network. A combination of products from different vendors can be integrated into the environment. A centralized controller software is used to configure all the nodes in the network and the network nodes now do not need different vendor Operating Systems (OS) and therefore, expensive software licenses. This significantly reduces the CapEx for an enterprise to deploy a network.

G. Operational Expense Savings

A recent Forrester survey points that a typical IT department spends eighty percent of their time on operations [13]. Different QoS maps can be assigned to the traffic flow dynamically, based on varying network conditions resulting in less time spent on configuration, troubleshooting and monitoring of traffic. By deploying a SDN, enterprises can see the whole network from the controller's perspective and can therefore configure end-to-end QoS in the most optimal manner. This ensures that the time and money spent on operating and maintaining (OpEx) is minimal.

H. Challenges Faced in Implementing SDN

SDN is a relatively new practice, and the cost of migrating to this new technology is high. An important aspect of this proposal is performance versus stability. QoS is an important parameter that defines the performance of a network [14]. Hakiri, Gokhale, et al. describe the problems arising due to network resource allocation while centralizing the control plane [15]. They describe how a SDN can best be utilized using a hybrid model approach with a well-defined APIs to enable QoS. However, human presence to define these parameters affects the flexibility of the network. From these papers, knowing about the challenges of implementing SDN helped us in developing a solution that can overcome these shortcomings.

IV. RESEARCH DESIGN AND METHODOLOGY

A. How do we demonstrate QoS on a traditional routing network model?

QoS in a traditional network can be implemented through various methods. A few examples of these are prioritizing traffic based on flow priority, bandwidth, source and destination address, ToS values and DSCP values.

To implement QoS in traditional networking devices, we used two Cisco routers connected to two end hosts acting as RTP/data server and client. We first created two class maps to classify the traffic. The primary purpose of a class map is to match incoming or outgoing packets to the specified class. In our case, we specified RTP and data traffic as two different traffic classes to be used as a matching criterion.

After the creation of the class map, we created the traffic policy and associated the traffic policy with the traffic class. Based on these classes, we assigned priority to the class of traffic such that the RTP traffic has been given a higher priority than data. The final step of the configuration was to attach a traffic policy to the egress interface of the routers [16].

After implementing QoS on Cisco routers, we enabled queuing to generate potential congestion. During a period of congestion, QoS based on configured class maps provided differentiated services in such a way that critical applications, such as RTP traffic, are prioritized over data. The demonstration of QoS in the traditional network was measured and documented in the results section to compare its performance with a SDN model.

B. How do we demonstrate QoS on a simulated SDN model?

Similar to the traditional networks, QoS in a SDN model can be configured based on various parameters such as source and destination address, DSCP values, and ToS bits.

To demonstrate QoS in SDN and achieve end-to-end implementation, we modified the source code of the Floodlight controller. As explained previously, Floodlight has a QoS application that can only rate limit interfaces but cannot implement QoS functionality through the setting of DSCP or ToS values.

As shown in Figure 2, a custom built controller running on a UNIX host is connected to two machines running OVS. These machines now emulate the switching and routing functionalities and are connected to the two end-hosts. We used Video LAN Client (VLC) stream server to generate RTP traffic on the host acting as the server. To obtain uniform results, we used a video file with a size of two gigabytes to be delivered to the client as a RTP/data stream. On the client side host, we fetch the RTP stream. Similarly, we created a data stream between the same end hosts. We used a UNIX-based tool called Secure Copy (SCP) and initiated a data transfer between the server and the client.

Initially, we used a SDN network simulator called Mininet to simulate the network topology. Mininet automatically installs OVS without any additional configuration. Since the bandwidth between virtualized end hosts created in Mininet is in the order of Gigabits Per Second (Gbps), we need to rate limit the interfaces. In order to achieve this, we configured a queue limited to 50 Megabits Per Second (Mbps) on the interface using an OVS utility called OVS-VSCTL (Open vSwitch Virtual Switch Controller). The OVS-VSCTL program maintains an active connection with the OVS configuration database. Any changes made through OVS-VSCTL will be applied to the running configuration on the OVS [17]. We then add a flow entry using the queue created between the ports connecting the server and client by making a REST call to the custom controller. The changes made to the running configuration can be verified in Mininet by running the iPerf tool. We initiated RTP/data streams from the server to the client and observed the results.

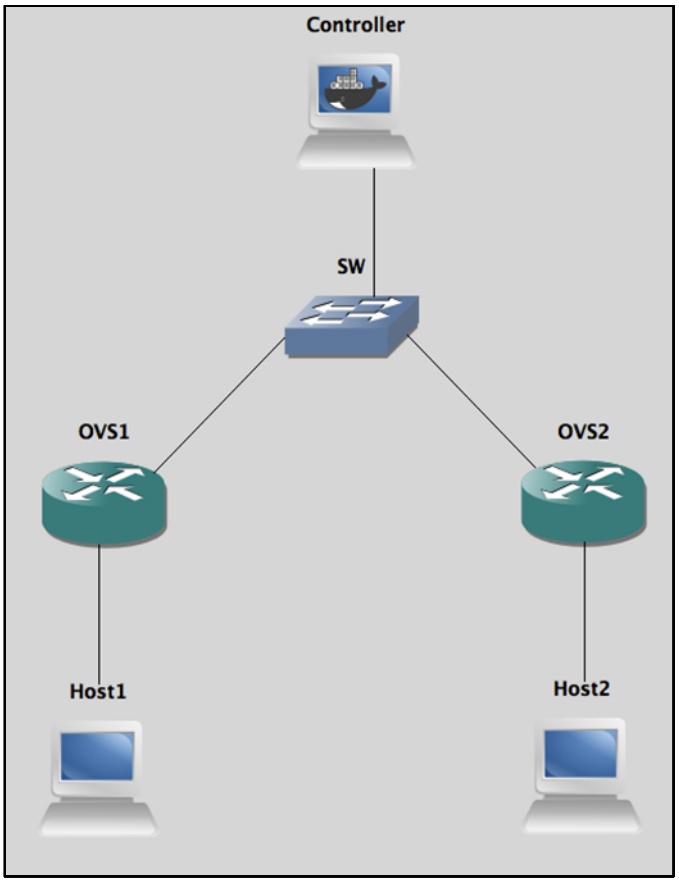
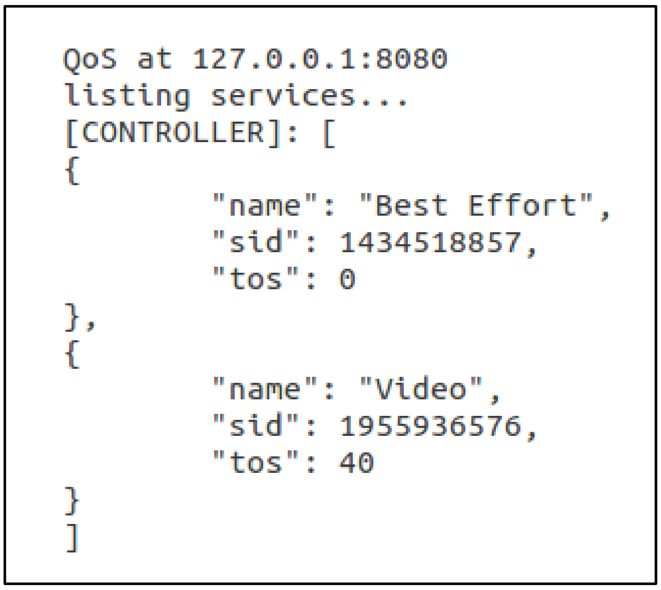


Figure 2. Topology

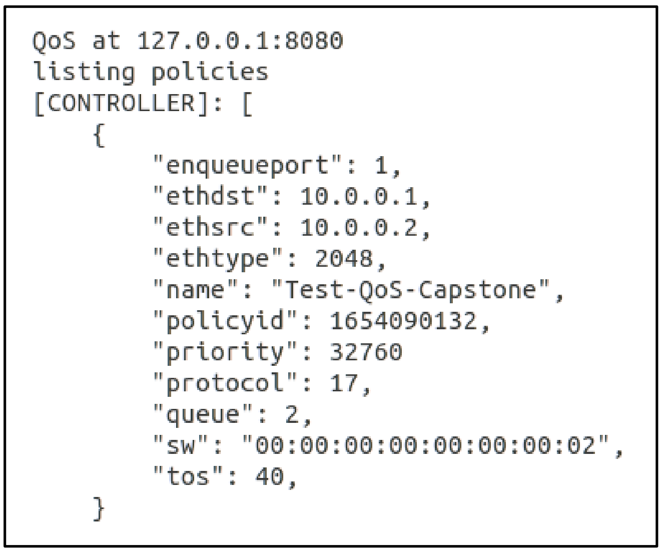
The next step is to create two services based on DSCP/ToS bits. These bits are assigned to RTP/data traffic and are used to differentiate these two Class of Services (CoS). We assigned a DSCP value of "40" which translates to a ToS value of "101000" for the RTP traffic, or a video stream in our research. The class maps generated by the QoS application on the controller is shown in Code Snippet 1.

Next, we created a QoS policy with a source address of the stream server and the destination address of the client with the service policy applied to it. This is shown in Code Snippet 2. We initiated the same RTP/data streams from the server to the client and observed the change in result with QoS enabled.



```
QoS at 127.0.0.1:8080
listing services...
[CONTROLLER]: [
{
        "name": "Best Effort",
        "sid": 1434518857,
        "tos": 0
},
{
        "name": "Video",
        "sid": 1955936576,
        "tos": 40
}
]
```

Code Snippet 1. Differentiation of traffic through class maps



```
QoS at 127.0.0.1:8080
listing policies
[CONTROLLER]: [
    {
        "enqueueport": 1,
        "ethdst": 10.0.0.1,
        "ethsrc": 10.0.0.2,
        "ethtype": 2048,
        "name": "Test-QoS-Capstone",
        "policyid": 1654090132,
        "priority": 32760
        "protocol": 17,
        "queue": 2,
        "sw": "00:00:00:00:00:00:00:02",
        "tos": 40,
    }
```

Code Snippet 2. Application of QoS policy

Since Mininet is an emulated environment, it does not provide true performance characteristics. Hence to perform true analysis and to keep the CapEx low, we replaced the existing setup with two Raspberry Pis running OVS. Raspberry Pis are miniature CPUs designed to achieve multiple functionalities and cost less than $35 each. The project leverages the capability of the Raspberry Pi to transform it into a virtual switch using OVS loaded on the open-source Raspbian OS [18]. Raspberry Pis are now performing the same functionality as of a Cisco router used in a traditional network. This can be extended to be used in a production network through the utilization of OVS on standard x86 hardware and white-box switches. Performance analysis for QoS is conducted on Mininet and Raspberry Pis and is documented in the results.

C. What technical aspects of the design must be addressed to improve QoS?

The design can include a distributed architecture consisting of a number of custom-built controllers in a clustered setup to address performance and scalability issues related to the implementation of QoS in an enterprise SDN. The controllers must synchronize with each other to understand the underlay network. Although the exchange of information can be accomplished using an Interior Gateway Protocol (IGP) such as Open Shortest Path First (OSPF), Intermediate Systems-Intermediate Systems (IS-IS), and Internal Border Gateway Protocol (iBGP), there are many issues such as redistribution, metric cost, etc., associated with these network protocols and involves excessive manual configuration changes. One of the reasons behind the formulation of SDN is to solve this problem. The problem can be addressed with certain Network Function Virtualization (NFV) technologies that will create virtualized instances of network elements on a standard x86 hardware. This overlay network can exchange information easily through several overlay protocols such as Virtual Extensible Local Area Network (VXLAN) and Multiprotocol BGP (MP-BGP) [19].

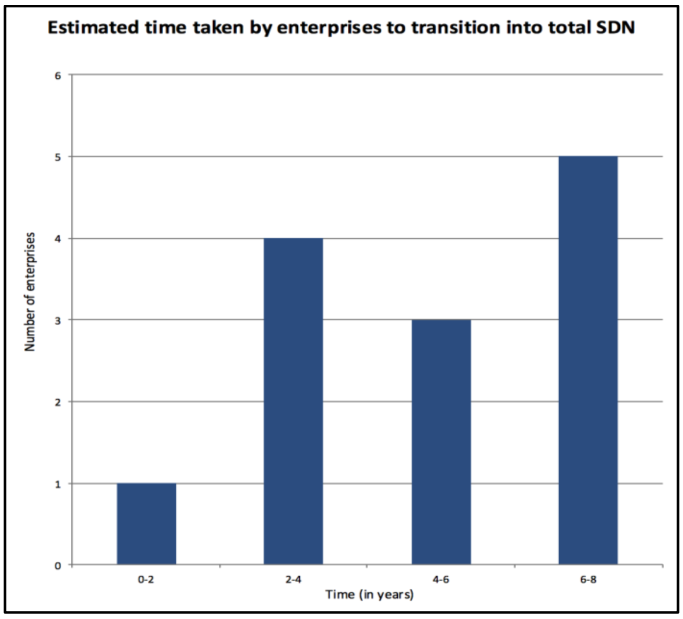


Figure 3. Time taken to transition into total SDN

The protocols used in a SDN are either OpenFlow or Open vSwitch Database (OVSDB). These two protocols are primarily utilized by the northbound APIs to interact with the southbound interfaces on the devices. This setup will work fine in a pure SDN consisting of all OpenFlow/OVSDB capable devices. As per the survey results shown in Figure 3 conducted among a number of networking companies, we found out that more than fifty percent of the enterprises are not willing to invest in a SDN solution right away. Instead, they are ready to implement a hybrid architecture consisting of both SDN and traditional networking elements. To provide a solution, we evaluated different hybrid technologies presently in the market. Some of the technologies we evaluated were OpenStack, CloudStack, and VMware and it was found that they cannot offer a total end-to-end QoS. Through the evolution of NFV, vendors such as Cisco and Brocade are developing virtualized versions of network devices (Vyatta vRouter, Nexus 1000V) which do not require dedicated hardware and powerful ASICS [20]. The virtualized routers that run on standard x86 hardware utilize the traditional Command Line Interface (CLI) and can provide the same QoS functionality of traditional networks. These are comparable to devices constituting the traditional network elements.

V. Research Results And Discussions

After collecting key performance parameters from both traditional network and SDN, we model the data for a graphical representation. Latency is a crucial parameter that we used for comparison.

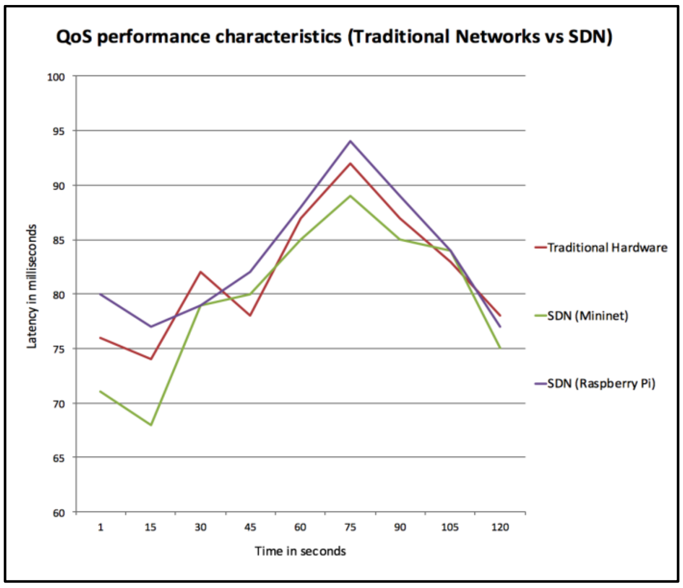


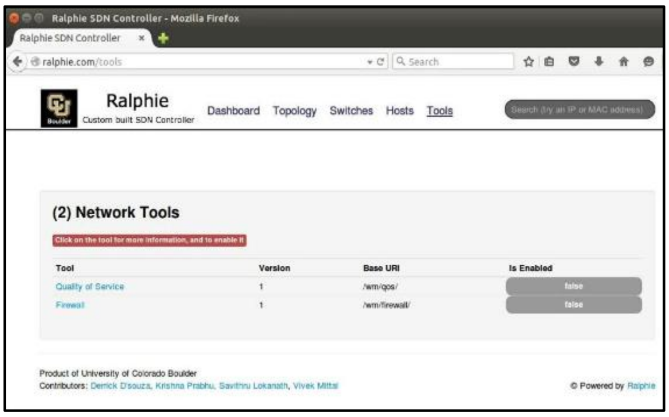
Figure 4. QoS performance characteristics

As depicted by the graph in Figure 4, we decided to include the SDN model with Mininet, whose parameters were measured under emulated conditions to ensure we have a benchmark to compare the traditional and SDN models. Ideal latency figures were achieved when using a SDN model with Mininet. These low figures of latency are not practically achievable in a physical network because of the latency introduced by routing and switching overhead. The traditional network model using Cisco routers represented by the red line on the graph initially had a low latency figure of about 75ms when compared to the SDN model. When the RTP burst traffic was introduced, as seen at around 25-40s mark, the SDN model outperforms the traditional model by about 2-3ms. This difference is attributed to the OVS performance of looking up the flow table. With time, the traditional network offers better QoS performance characteristics than a SDN model. This difference can be attributed to the performance limitations of the hardware that is being used (Raspberry Pi) to run the OVS daemon. Some of the limitations that we encountered were limited speed settings on the Pi's interface, no support for multi-threading and low Random-Access Memory (RAM).

Although the QoS performance characteristics in a SDN is not better than a traditional routing model, it provides similar stability. QoS in a SDN is still in its infancy, and there are techniques that the SDN model cannot address such as AutoQoS and Modular QoS. Hence, we modified the source code of the Floodlight controller and added features that will improve configuring QoS.

Some of the features are,

- Accept DSCP and ToS values as an input parameter to configure QoS
- Accept source and destination Internet Protocol version 4 (IPv4) and Media Access Control (MAC) address to achieve end-to-end QoS
- Modular encoding of QoS configuration through the use of JavaScript Object Notation (JSON) modeling language.
- Ease of configuring QoS through a custom GUI


Ralphie SDN Controller - Mozilla Firefox
Ralphie SDN Controller
ralphie.com/tools
Search
Ralphie
Custom built SDN Controller
Dashboard
Topology
Switches
Hosts
Tools
Search (try an IP or MAC address)

(2) Network Tools

Click on the tool for more information, and to enable it

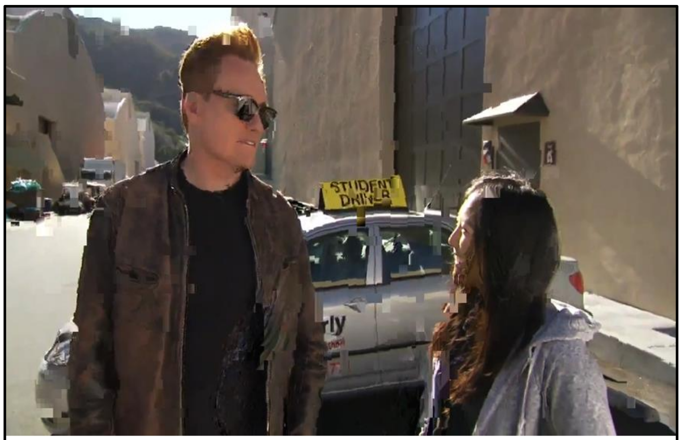
Tool	Version	Base URI	Is Enabled
Quality of Service	1	/wm/qos/	false
Firewall	1	/wm/firewall/	false

Product of University of Colorado Boulder
Contributors: Derrick D'souza, Krishna Prabhu, Savithru Lokanath, Vivek Mittal

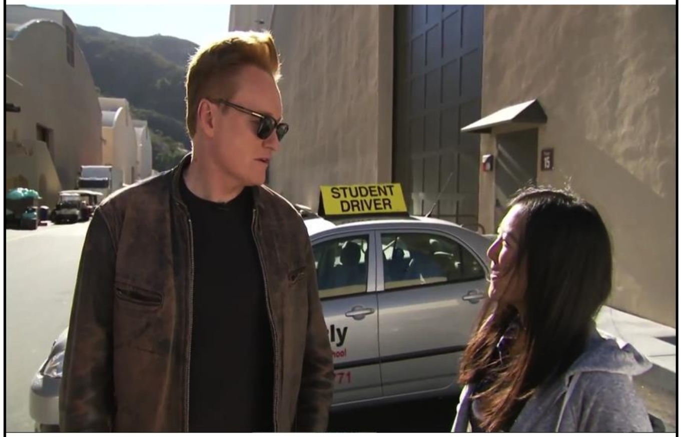
© Powered by Ralphie

Figure 5. Screenshot of the custom controller

Figure 5 shows the screenshot of the GUI for the custom built controller used in the project. QoS module can be enabled by clicking on the Tools tab located on the controller's dashboard. Figure 6 shows the screenshots of a video playing without and with QoS in a SDN model. Figure 6.a depicts the distorted video output that is caused due to the lack of prioritization of video traffic over data. After enabling the QoS module in the custom built controller, we were able to see a significant difference in the output of the video stream, which is as seen in Figure 6.b.



a. A video streaming without QoS



b. A video streaming with QoS

Figure 6. Comparison of a video playing without and with QoS

The ideal solution to solve the problems related to interoperability in a hybrid network is to develop an orchestration tool which understands both traditional and SDN protocols. The solution can be implemented by designing a generic YANG service model in the orchestration layer and have the tool push out configurations based on the hybrid underlay network. Orchestration tools must use southbound protocols such as OpenFlow, OVSDB to configure SDN devices and Simple Network Management Protocol (SNMP), Network Configuration (NETCONF) to configure traditional network devices.

Item	Cost per device ($)	Quantity	Total Cost ($)	Type of device
Cisco Catalyst WS-C6506-E	4235	20	84700	Switch
Cisco 3945-V/K9	5147	8	41176	Router

Table 2. Cost of setting up an enterprise network with traditional networking hardware

Item	Cost per device ($)	Quantity	Total Cost ($)	Type of device
Pica8 P-3290	3510	28	98280	Generic Switch
CumulusOS	3300	1	3300	License

Table 3. Cost of setting up an enterprise network with SDN

We performed a cost-benefit analysis to realize the monetary saving for enterprises when they make a switch from a traditional hardware-based networking infrastructure to SDN. For the analysis, we considered an enterprise serving about four hundred nodes using eight routers and twenty switches. We calculated the CapEx that will be spent by the organization on traditional networking hardware and is shown in Table 2. We analyzed the equivalent CapEx for SDN to support such an enterprise and is documented in Table 3. From the analysis, we estimated that the enterprise would be saving about $25,000 or about 20% on CapEx. The enterprise would also be saving a considerable amount of about $10,000 per year on OpEx which is estimated to be around 43% to 45% savings compared to traditional networks. This can be attributed to better network management practices, as SDN reduces the OpEx by converging the processes into a centralized platform through automation [21].

VI. Conclusion And Future Research

In our research paper, we conducted several tests on traditional networks and SDN to compare their QoS performance. We wrote code to improve QoS in SDN. From the results we obtained, we conclude that the QoS in SDN can be improved to provide production grade performance by running the QoS application that we developed on standard x86 hardware or white-box switches.

The code developed in our project can be integrated with other applications to provide low latency and give priority to a certain type of traffic. Implementation of our code in switches that are in compliance with the industry-standard white-box devices will provide a better marketing model. Future work can begin by understanding the architecture of several white-box switches available on the market and find out a way to seamlessly integrate our application. The devices that can be considered are switches from Pica8 and BigSwitch Networks [22]. These devices can run custom code and provide excellent performance characteristics. Open-source OS such as Cumulus OS, Raspbian or any other UNIX distribution can be installed on these white-box switches. Therefore, enterprises can now leverage their automation requirements on these devices through the utilization of good DevOps practices [23].

Further tests can include, increasing the number of devices in the network to have more traffic that represents a production environment. The code can also be adapted to work with different controllers.

VII. Acknowledgement

We would like to thank our faculty advisor Dr. Levi Perigo for helping us throughout the tenure of this project with his expertise and constant guidance. We would also like to extend our gratitude to Mr. Rob Hagens, VP of IP Architecture in Zayo Group for guiding us through our project plan. We would like to thank the network engineers at CableLabs, Comcast, Google, and Visa for their valuable inputs that helped make this project possible.

We extend our sincere thanks to the Interdisciplinary Telecommunications Program at the University of Colorado Boulder for providing us access to the networking laboratory and equipment for conducting our research. Lastly, this project would not have been possible without the support of our Director, Dr. David Reed and his constant encouragement to relentlessly innovate to achieve results.

References:

[1] D. Geer, "Five reasons IT pros are not ready for SDN investment", SearchSDN, 2016. [Online]. Available: http://searchsdn.techtarget.com/feature/Five-reasons-IT-pros-are-not-ready-for-SDN-investment. [Accessed: 10-Apr-2016].

[2] M. Chubirka, "Are you ready for an SDN deployment?", SearchNetworking, 2016. [Online]. Available: http://searchnetworking.techtarget.com/tip/You-may-not-be-ready-for-SDN-deployments. [Accessed: 11-Apr-2016].

[3] "Implementing Quality of Service." Cisco. Cisco, 15 Feb. 2008. Web. http://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-packet-marking/13747-wantqos.html [Accessed: 29-Mar-2016].

[4] “Lsoapi,” OPNFV Wiki. [Online]. Available at: https://wiki.opnfv.org/lsoapi. [Accessed: 18-Oct-2015].

[5] “Network Monitoring Tools ,” Network Monitoring Tools. [Online]. Available at: http://www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html. [Accessed: 27-Feb-2016].

[6] N. Heresy, "The Scaling Implications of SDN", Network Heresy, 2011. [Online]. Available: https://networkheresy.com/2011/06/08/the-scaling-implications-of-sdn/. [Accessed: 12-Apr-2016].

[7] “Internet2 Performance Tools,” Performance Tools. [Online]. Available at: http://www.internet2.edu/products-services/performance-analytics/performance-tools/. [Accessed: 15-Nov-2015].

[8] “Quality of Service Networking.” [Online]. Available at: http://docwiki.Cisco.com/wiki/Quality_of_Service_Networking. [Accessed: 14-Dec-2015].

[9] R. Wallner, and R. Cannistra, “An SDN Approach: Quality of Service using Big Switch’s Floodlight Open-source Controller,” Proceedings of the Asia-Pacific Advanced Network, vol. 35, 2013, Pages 14-19, ISSN 2227-3026 doi: http://dx.doi.org/10.7125/APAN.35.2 [Accessed: 7-April-2016].

[10] H.E. Egilmez, S.T. Dane, K.T. Bagci, and A.M. Tekalp, “OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks,” in Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific , pp.1-8, 3-6 Dec. 2012 [Accessed: 16-Nov-2015].

[11] R. Voicu, “Traffic shaping with OVS and SDN,” Jan-2015. [Online]. Available at: https://indico.cern.ch/event/376098/contribution/24/attachments/749534/1028288/ovs-rv_20150602_v1.pdf. [Accessed: 16-Nov-2015].

[12] B. Naudts, “Evaluating the impact of SDN on CapEx and OpEx,” IEEE, 2015. [Online]. Available: http://sites.ieee.org/sdn4fns/files/2013/11/SDN4FNS_panel_presentation_Bram_Naudts.pdf. [Accessed: 29-Oct-2015].

[13] A. Kindness, “Forrester Research : Research : Is Software-Defined Networking Ready For The Enterprise,” Forrester.com, 2015. [Online]. Available: https://www.forrester.com/Is+SoftwareDefined+Networking+Ready+For+The+Enterprise+Part+1+Of+3/fulltext/-/E-RES111821. [Accessed: 29-Oct-2015].

[14] S. Sezer, et al “Are we ready for SDN? Implementation challenges for software-defined networks,” in Communications Magazine, IEEE , vol.51, no.7, pp.36-43, July 2013 doi: 10.1109/MCOM.2013.6553676 [Accessed: 16-Feb-2016].

[15] A. Hakiri, A. Gokhale, P. Berthou, D. Schmidt, and T. Gayraud, “Software-Defined Networking: Challenges and research opportunities for Future Internet,” Computer Networks, vol. 75, Part A, 24 Dec. 2014, Pages 453-471, ISSN 1389-1286 doi: dx.doi.org/10.1016/j.comnet.2014.10.015 [Accessed: 16-Nov-2015].

[16] "Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2 - Configuring the Modular Quality of Service Command-Line Interface [Cisco IOS Software Release 12.2]", Cisco, 2016. [Online]. Available: http://www.cisco.com/c/en/us/td/docs/ios/12_2/qos/configuration/guide/fqos_c/qcfmcli2.html. [Accessed: 11-Apr-2016].

[17] "Open VSwitch Manual." Open VSwitch. Web. http://openvswitch.org/support/dist-docs/ovs-vsctl.8.txt [Accessed: 10-Oct-2015].

[18] "Raspberry Pi FAQs - Frequently Asked Questions", Raspberry Pi, 2016. [Online]. Available: https://www.raspberrypi.org/help/faqs/#introWhatIs. [Accessed: 12-Apr-2016].

[19] "VXLAN/EVPN: Standards based Overlay with Control-Plane - WKSB Solutions", Wksbsolutions.com, 2016. [Online]. Available: http://wksbsolutions.com/?p=1755. [Accessed: 22-Mar-2016].

[20] K. Tolly, "Cisco, Brocade virtual router review", SearchNetworking, 2016. [Online]. Available: http://searchnetworking.techtarget.com/tip/Cisco-Brocade-virtual-router-review. [Accessed: 22-Mar-2016].

[21] "Software Defined Networking (SDN) - Overview", Cisco, 2016. [Online]. Available: http://www.cisco.com/c/en/us/solutions/software-defined-networking/overview.html. [Accessed: 12-Apr-2016].

[22] D. Geer, “White-box switching: Three paths to network programmability,” SearchSDN, 2015. [Online]. Available: http://searchsdn.techtarget.com/feature/White-box-switching-Three-paths-to-network-programmability. [Accessed: 01-Dec-2015].

[23] Cumulus Networks, “The first, true Linux OS for data center networking,” 2015. [Online]. Available: https://cumulusnetworks.com/media/cumulus/pdf/misc/Cumulus-Linux-Datasheet.pdf. [Accessed: 01-Dec-2015].