DEPARTMENT OF COMPUTER SCIENCE | UNIVERSITY OF KENTUCKY

Quality of Service in Software Defined Networks

MASTERS PROJECT REPORT

SUBMITTED BY : Sumaira Shamim

ADVISOR : Dr. Zongming Fei

March 22, 2016

CONTENTS

1 INTRODUCTION

This project focuses on research and proof of concept about how OpenFlow and Software Defined Networks can be used to provide Quality of Service. The main goal of Software defined networks is to help create a smarter network then we have today. The concept of splitting the control plane and the data plane and increasing programmability of the network by taking the intelligence to a central controller allows us to make our networking entities more application aware. The advantages of a smarter network are better utilization of network resources, adaptability, lower cost and better services.

The decoupling of the control plane and data plane allows a high degree of control over flows that pass through the switches via intelligent controller applications. The increasing advent of real time applications also bring with them the need of demanding Quality of Service guarantees and the network protocols need to have the capability of meeting these requirements [1]. The quality of service evaluation of a network is to make certain that the application receives its due set of connection parameters according to the QoS requirements of that application. The parameters can be throughput, End to end delay, Jitter (deviation from average end-to-end delay) and number of packets lost or damaged in the channel [2].

We need to utilize OpenFlow protocol in SDN for providing dynamic QoS guarantees for different classes of flows with different service requirements by making dynamic routing decisions or efficient priority queuing.

2 RELATED WORK FOR IMPLEMENTING QOS IN SDN

This report summarizes the techniques that were researched for literature review on the project.

2.1 RATE LIMITING AND TRAFFIC SHAPING

The rate limiting or traffic shaping technique is achieved by bandwidth control among the OVS ports for rate limiting. Standard OpenFlow protocol specifications like "enqueue" can be used for queuing traffic based on inspection of flows by the SDN controller in order to assign different queuing QoS policies for different flows. The queues inside OVS itself are configured by an administrator of OpenFlow versions less than 3.0 while versions 3.0 and above introduce "OF-Config" that will make this process much easier [3].

Since only bandwidth guaratees and FIFO scheduling is not enough to provide QoS guarantees in SDN, using multiple packet schedulers of Linux kernel e.g., Hierarchical Token Bucket, Randonly Early detection and Stochastic Fairness queuing can be utilized to overcome packet scheduling issues. The strong traffic control system of Linux can be used to provide traffic shaping, queuing and congestion avoidance. This approach would require data path extensions for OpenFlow for kernel space queues [4]

Some QoS extensions to OpenFlow have been developed in [5] that have the capability of taking in high level QoS requirements of applications and automatically modify the QoS parameters on network devices in the form of rate limiters and dynamic priority assignment. This controller removes the need of manually configuring QoS requirements for each device in the network and centrally controls these configurations.

2.2 DIFFERENTIATED SERVICES CODE POINT (DSCP)

This technique is established by using the 8 Type of Service bits in existing IP header for implementing differentiating class of service. Standard OpenFlow protocol specifications allow rewriting of the Type of Service field in the IP header using "network ToS". This is achieved by the SDN controller after flow matching and classification according to different ToS polices for different flows [3].

2.3 MULTIPATH ROUTING

The idea of multipath routing is to split and balance a flow among a set of alternative paths using a Multipath agent which splits tcp session into multiple virtual sessions at the end host by starting internal sockets. The controller can intercept the first packet of each new connection and find out the application it belongs to by inspecting the payload and make sure the flows with the same connection take different paths. All the flows are multiplexed together via a multiflow agent at the destination host. A speciating routing module in the controller keeps track of the subflows, QoS policies and dynamic calculation of paths which serve the best QoS to the flows [2].

2.4 DYNAMIC ROUTING OF SPECIFIC FLOWS

This approach is designed for special QoS flows like multimedia flows where timely delivery is preferred over reliability usually without affecting other types of traffic. The QoS flows can be dynamically routed on different paths while the rest of the data flows remain on the shortest optimal path for them. The traffic can be differentiated by Traffic class header field in MPLS, TOS (Type of Service) field of IPv4 header, Traffic class field in IPv6 header, source IP address of a known multimedia server and TCP port numbers. Collection of current global network state information like delay, bandwidth and rate of packet loss is also required for dynamic routing [6].

Another approach for dynamic routing is by using two level QoS flows for multimedia like MPEG-4 which encodes videos in a base layer and enhancement layers. The controller identifies the level 1 QoS (base layer), level 2 QoS (enhancement layers) and normal traffic flows and updates the switches with calculated dynamic routes for level 1 QoS and level 2 QoS flows after estimating bandwidth and delays on the links [7].

Another technique in [1] tries to find an optimal path between two end points with the minimum cost (that does not increase the capacity of any link) after ruling out the links that

break specified QoS constraints.(i-e delay and packet loss do not increase a threshold).The architecture dynamically updates the network parameters to become aware of the status of the network and calculate the best route according to the QoS specified policies. The model is based on "multi-commodity flow constrained shortest path problem (MCFSP)" .

3 MOTIVATION FOR THE PROJECT

The project aims for setting up an experiment for implementing dynamic Quality of Service in software defined networks using GENI portal as the platform. The project serves to be a proof of concept on how the behavior of individual network flows can be changed from the default shortest forwarding path through Northbound API applications running on top of the SDN controller.

4 PROJECT DESCRIPTION

4.1 EXPERIMENT SETUP

The experiment is set up in GENI portal using Xen virtual machines as hosts and Open vSwitch nodes as virtual switches between them. The switches are configured with default shortest path routing rules for each of the hosts using an SDN controller. The SDN controller used is Floodlight open source controller which exposes a REST API for the northbound applications to use. The northbound QoS application for the experiment is developed in Python. It uses python packages like Networkx and untangle in order to construct the topology using the information provided by the controller REST API and RSpec document provided by the GENI portal.

4.2 THE NETWORK TOPOLOGY AND THE CONTROLLER

The topology constructed in GENI portal is shown in Figure 4.1. *May replace with a 4 switch topology figure later*. The topology is not linear and has multiple paths between a pair of hosts in order to have alternate paths for routing flows based on bandwidth. The nodes labeled as switches run Openvswitch on them configured to use an OVS bridge to connect to the hosts and other OVS nodes.
Each of the OpenFlow switches (OVS) have a Datapath id (DPID) associated with them which is used by the SDN controller to identify and control the switches. Using the web interface of the Floodlight controller, we can see the switches connected to the controller and the various information statistics related to them in Figure 4.2.

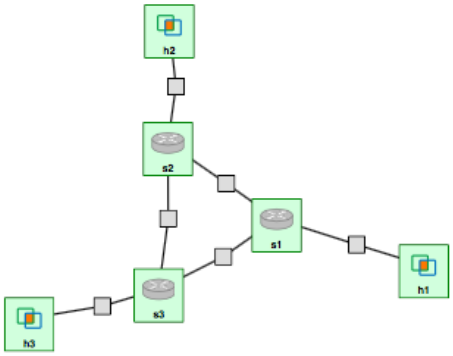

Figure 4.1: GENI topology

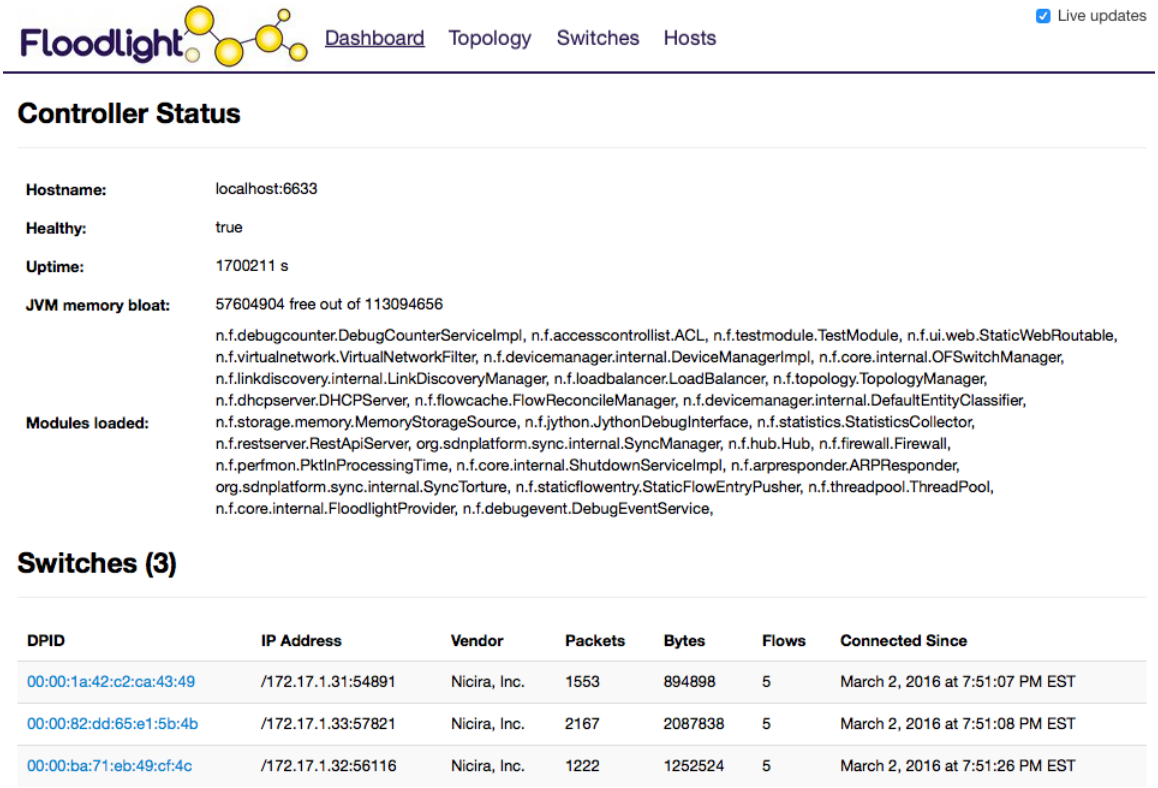

Figure 4.2: Floodlight dashboard

We can also see the OpenFlow rules that a switch is configured with in Figure 4.3. Right now, the rules shown are default shortest path rules that the floodlight controller pushed into the

switches using the static flow pusher REST API.

Flows (5)

Cookie	Table	Priority	Match	Apply Actions	Write Actions	Clear Actions	Goto Group	Goto Meter	Write Metadata	Experimenter	Packets	Bytes	Age (s)	Timeout (s)
45035998543468674	0x0	1	eth_type=0x0x800 ipv4_src=10.10.4.1 ipv4_dst=10.10.5.1	actions:set_eth_src=02:54:f4:fb:57:a6,set_eth_dst=02:0d:3a:65:56:ec,output=3	n/a	n/a	n/a	n/a	n/a	n/a	146	14308	1688306	0
45036000423048352	0x0	1	eth_type=0x0x800 ipv4_src=10.10.6.2 ipv4_dst=10.10.4.1	actions:set_eth_src=02:cd:ed:37:f7:f9,set_eth_dst=02:26:62:93:d0:5c,output=2	n/a	n/a	n/a	n/a	n/a	n/a	629	433106	1688306	0
45035998054472618	0x0	1	eth_type=0x0x800 ipv4_src=10.10.5.1 ipv4_dst=10.10.4.1	actions:set_eth_src=02:cd:ed:37:f7:f9,set_eth_dst=02:26:62:93:d0:5c,output=2	n/a	n/a	n/a	n/a	n/a	n/a	149	14602	1688306	0
45035999636593981	0x0	1	eth_type=0x0x800 ipv4_src=10.10.4.1 ipv4_dst=10.10.6.2	actions:set_eth_src=02:43:58:c8:59:43,set_eth_dst=02:cd:3b:33:87:63,output=1	n/a	n/a	n/a	n/a	n/a	n/a	625	432714	1688306	0
45036000137237139	0x0	2	eth_type=0x0x806	actions:output=controller	n/a	n/a	n/a	n/a	n/a	n/a	4	168	1688306	0

Figure 4.3: Rule table in OpenFlow switch

4.3 FLOW OF THE APPLICATION

The control flow of the application starts from constructing the network graph, polling the OpenFlow switches using the floodlight REST API to find active flows and then uses the statistics collector module of floodlight to query link statistics. Based on the statistics received, the application chooses a different but better path for active flows based on link bandwidth.

The design decisions taken are the following:

- The topology is constructed using Python Networkx library undirected graph. The information for the nodes and the edges is received from the GENI RSpec document as well as the Floodlight controller REST API calls.
- The bandwidth statistics from the statistics collector module of floodlight queries the links every 10 seconds.
- The different simple paths are only on an end-to-end basis for a pair of hosts. A simple path means that no node is repeated between a path between two hosts.
- A switch can only decide to change the path for a flow if the sender's IP is on the same subnet as the switch's interface.
- The active flows are decided by polling the switches every 10 seconds and comparing the last packets hit count with the new polled information.
- ***How the application selects the path based on bandwidth and list of simple paths between the two hosts***
- The QoS rule written by the controller in the switches for a particular flow has an idle timeout of 60 seconds.

REFERENCES

[1] F. Ongaro, "Enhancing quality of service in software defined networks," Master's thesis, University Of Bologna, 2014.

[2] E. Chemeritskiy and R. Smelansky, "On qos management in sdn by multipath routing," in *Science and Technology Conference (Modern Networking Technologies) (MoNeTeC), 2014 International*, pp. 1–6, Oct 2014.

[3] R. Wallner and R. Cannistra, "An sdn approach: quality of service using big switchâĂŹs floodlight open-source controller," *Proceedings of the Asia-Pacific Advanced Network*, vol. 35, pp. 14–19, 2013.

[4] A. Ishimori, F. Farias, E. Cerqueira, and A. Abelém, "Control of multiple packet schedulers for improving qos on openflow/sdn networking," in *Proceedings of the 2013 Second European Workshop on Software Defined Networks*, EWSDN '13, (Washington, DC, USA), pp. 81–86, IEEE Computer Society, 2013.

[5] W. Kim, P. Sharma, J. Lee, S. Banerjee, J. Tourrilhes, S.-J. Lee, and P. Yalagandula, "Automated and scalable qos control for network convergence," in *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, INM/WREN'10, (Berkeley, CA, USA), pp. 1–1, USENIX Association, 2010.

[6] H. Egilmez, S. Dane, K. Bagci, and A. 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, Dec 2012.

[7] H. Egilmez, S. Civanlar, and A. Tekalp, "An optimization framework for qos-enabled adaptive video streaming over openflow networks," *Multimedia, IEEE Transactions on*, vol. 15, pp. 710–715, April 2013.