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 8, 2016

1 Abstract

CONTENTS

2 INTRODUCTION

This project focused on research 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]. 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.

3 TECHNIQUES FOR IMPLEMENTING QOS IN SDN

This report summarizes the techniques that were researched in this independent study.

3.1 AN SDN APPROACH: QUALITY OF SERVICE USING BIG SWITCH'S FLOODLIGHT OPEN-SOURCE CONTROLLER

Wallner et al. use OpenvSwitch and Floodlight OpenFlow controller to define and manage quality of service in software defined networks via rate limiting/traffic shaping and DSCP diffserv (Differentiated Services Code point). This is achieved by bandwidth control among the OVS ports for rate limiting and using the 8 Type of Service bits in existing IP header for implementing differentiating class of service. They use standard OpenFlow protocol specifications like "enqueue" for queuing traffic or "network ToS" to rewrite the ToS field in the IP header. The SDN controller runs modules for flow matching and classification, insertion and deletion of rules in flow tables, network topology updates and handling QoS policies. The QoS policies are maintained using two python applications "QoS Manager" and "QoS Path" in the control plane that use the northbound interface and assign policies like queuing policy or differentiating ToS policy to different flows. The queues inside the switch are configured by an administrator but OpenFlow version 3.0 introduces "OF-Config" that will make this process easier [2].

3.2 CONTROL OF MULTIPLE PACKET SCHEDULERS FOR IMPROVING QOS ON OPENFLOW/SDN NETWORKING

This paper improves upon the approach for packet scheduling in Wallner et al. [2]. Ishimori et al. present QoSFlow using multiple packet schedulers of Linux kernel e.g., Hierarchical

Token Bucket, Randonly Early detection and Stochastic Fairness queuing to overcome packet scheduling issues. They argue that only bandwidth guaratees and FIFO scheduling is not enough to provide QoS guarantees in SDN. Their module adds OpenFlow data path extensions with queues located in the kernel space and use Netlink and sockets to connects kernel space to the userspace. This enables the mapping of QoS messages from OpenFlow to Netlink messages (Netlink messages are accepted by Linux kernel for network resource management from userspace). Using a strong traffic control system of Linux, traffic shaping is done through hierarchical token bucket algorithm, packets in queues are sent out the transmission channel via round robin algorithm and congestion avoidance by avoiding saturation of queues is achieved by Randonly Early detection [3].

3.3 On QoS management in SDN by multipath routing

Chemeritskiy et al. talk about quantitative quality of service evaluation of a network that is giving the application 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. All these requirements need network resources that cannot be allocated beforehand to meet QoS efficiently. They propose MPRSDN (Multi Path Routing Software Defined Network) which is a best effort approach because they do not reserve resources beforehand so that they can be efficiently utilized. It does not require any special hardware and just needs the hosts to install the software agent for MPR. They want to avoid controller interference because it violates end to end principle. So implement something with only the help of application at end hosts. They split and balance a flow among a set of alternative paths using MP agent which splits tcp session into multiple virtual sessions at the end host by starting internal sockets. Those sockets set up individual connections in the network which still doesn't ensure usage of different paths but we can change that using SDN. 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. In this way, each subflow establishes a connection with unique pair of L4 addresses and each treated as an independent ordinary flow where we can also have priority in the sub flows using higher TOS/DSCP. 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 [4].

3.4 Realizing the quality of service (QoS) in software-defined networking (SDN) based cloud infrastructure

Govindarajan et al. present a special QoS Controller called Q-Ctrl by exploiting the fact that OpenFlow lets administrators decide the path which the flows will take through the network. Q-Ctrl uses a virtual overlay via OVS over the physical network controlled via an SDN controller. Since SDN controller itself does not provide mechanisms for QoS management and policy maintenance, they provide a reference architecture, a QoS lifecycle control system and Q-Ctrl system implementation details. Q-Ctrl can be implemented in direct, controller or simulated

environment depending on the requirements. It uses "Topology manager" and "Topology information" for maintaining network topology information and assign network links to different applications based on that information. The QoS lifecycle describes how a particular QoS flow is added into a queue, how specific QoS rules are added in flow table, modified and removed. The architecture was implemented in a cloud data center environment to effectively manage and allocate virtual machines' bandwidth in the data center [5].

3.5 OpenQoS: an openflow controller design for multimedia delivery with end-to-end quality of service over software-defined networks

Traditional QoS architectures like Diffserv (soft QoS guarantees) and IntServ (hard QoS guarantees) do not have a complete picture of overall network resources and are based on a hop by hop routing architecture. OpenQoS is a controller design for multimedia apps specifically where timely delivery is preferred over reliability usually without affecting other types of traffic. The incoming flows are split in to data flows and multimedia flows so that multimedia QoS flows can be dynamically routed while the rest of the data flows remain on the shortest optimal path for them. Their implementation consists of three interfaces. A standard Controller-switch interface, A Controller-controller interface in case of multiple controllers in large network management and a Controller-service interface for service providers to specify policies for their flows. The controller itself has functional modules of topology management, route and flow management, route calculation, call admission and traffic policing [6].

Multimedia traffic is 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.They propose to use MPLS headers and for making routing decisions, collect current global network state information like delay, bandwidth and rate of packet loss). They define a NP-Complete problem which minimizes the cost function subject to a delay variation of a max D and propose to use Lagrangian Relaxation Based Aggregated Cost (LARAC) algorithm for route calculation when a Packet_in event arrives. The routes are recalculated if topology manager indicates changes or QoS policies are changes. The route management module in controller collects up to date network information from switches by sending Feature_Request and Feature_Reply of OpenFlow protocol. Then links are assigned states of "congested" or "uncongested" and decisions made based on cost fucntion. Route managers keep track of flows already in switches and delete if a route is now congested or insert if a previous congested route now uncongested [6].

3.6 MonSamp: a distributed sdn application for QoS monitoring

MonSamp is a controller application specially dedicated to monitoring the QoS of flows in software defined networks. They rely on traffic sampling for the purpose of QoS implementation for flows by taking out a subset of traffic for each and migrate those flows to some commodity server for a later QoS monitoring. They do not actually implement any QoS but provide an efficient approach for traffic monitoring in order to get more information about link and network parameters that may or may not affect QoS [7].

3.7 AN OPTIMIZATION FRAMEWORK FOR QOS-ENABLED ADAPTIVE VIDEO STREAMING OVER OPENFLOW NETWORKS

Egilmez et al. use dynamic QoS optimization for multimedia traffic i.e., no resource reservation/priority queuing and constrained shortest path calculation using two level QoS flows i-e use a video format like MPEG-4 which encodes videos in a base layer and enhancement layers. Then the base layer is served highest QoS and the enhancement layers are best service causing a trade off decision between performance and cost. The controller identifies the level 1 QoS (base layer), level 2 QoS (enhancement layers) and normal traffic flows and updates the switches using a "service mgmnt" module and "route mgmnt" + "route calculation" modules. Similar to [6], the "route calculation" uses Constrianed Shortest Path problem (Lagrangian Relaxation Based Aggregated cost algorithm (LARAC)) to minimize cost function subject to delay variation. The estimation of bandwidth and packet loss are done intermittently using the switches and passed to "route mgmnt" [8]. The estimations are done using the following ways

- Packet loss measures estimated using per-flow counters in the OF switches
- Link bandwidth by experimentation/manually
- Delay is calculated by average observed delay using time stamping methods
- Delay variation (jitter) is the first derivative of the above delay

The above characteristics can be scaled by a factor according to the application QoS requirements [8]. They used LEMON* for test network implementation. LARAC algorithm already implemented in LEMON*.

**"LEMON stands for Library for Efficient Modeling and Optimization in Networks. It is a C++ template library providing efficient implementations of common data structures and algorithms with focus on combinatoric optimization tasks connected mainly with graphs and networks." [9]*

3.8 AUTOMATED AND SCALABLE QOS CONTROL FOR NETWORK CONVERGENCE

Kim et al. develop some QoS extensions to OpenFlow 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. The controller creates slices in the network and each slice is assigned to traffic of different applications according to their performance requirements. They specify as special QoS API; an extension to OpenFlow which is responsible for mapping flows to rate limites and priority queues for enforcing bandwidth and delay guarantees. The flows are differentiated for QoS services in reactive mode by the controller by inspecting the first packet for each flow and adjusting the configurations in the queues in switches for that flow. An adaptive aggregation module lessens the load of the controller by reducing the packet_ins to the controller and aggregating flows into groups [10].

3.9 ENHANCING QUALITY OF SERVICE IN SOFTWARE DEFINED NETWORKS

This thesis report implements a Qos management and orchestration architecture that integrates with SDN infrastructure. The author formulates the problem of guaranteeing good QoS in terms of packet loss and delay taking into account constraints on network i-e maximum packet loss/delay acceptable by the service and the bandwidth available on links. The model is based on "multi-commodity flow constrained shortest path problem (MCFSP)" which 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 controller modules are the "Network Topology Mapper" which maintains information about network topology and links, a "Static Path Inserter" which inserts the flow rules into the flow tables, the "Network Status Collector" derives network parameters about bandwidth, latency and packet loss, a "Dynamic Path Inserter" installs the rule found by the "path finder" which implements the "MCFSP" model for finding the best path as decribed above using the information from the "Network Status Collector". They implemented the controller modules in JAVA and uses AMPL and CPLEX for solving the mathematical dynamic path finding [1].

3.10 APPLICATION-AWARE DATA PLANE PROCESSING IN SDN

The extended architecture of SDN in Mekky et al. implemented as a prototype on Open vSwitch reduces switch-controller delay by keeping most of the processing in data plane thereby eliminating unnecessary detouring of traffic or the need to track flow state between different applications. Examples of applications that can be implemented in software can be QoS applications, NAT, Firewall, TCP splicing etc. The central idea of the implementation is that logic is not kept restricted to the controller only. Instead, all the OpenFlow enabled switches maintain local application specific processing logic and have stateful application processing capabilities and also utilize L4-L7 header information. OVS was chosen as the platform because it allows extension of the data path and introduction of application processing capabilities in it and also provides opportunities to make a modular design and interface. The userspace flow table pipeline is modified by replacing the last table with a special "app table". The table miss rules which specified the action "Trap to the controller" are replaced with "go to app table" action and hence intercepted instead of being sent to the controller. The app table matches packets to actions like a standard unmodified OpenFlow table but the actions are special corresponding to application processing actions. These actions direct packets to application instances running inside the OVS. The SDN controller can also write and delete rules from the app table like a standard flow table. The support for these special actions comes from "vendor actions" supported by OpenFlow protocol. In case a table miss happens in the app table as well, a packet_in event is sent to the controller. This allows control applications to run inside the OVS datapath increasing performance significantly [11].

REFERENCES

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

[2] 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.

[3] 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.

[4] 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.

[5] K. Govindarajan, K. C. Meng, H. Ong, W. M. Tat, S. Sivanand, and L. S. Leong, "Realizing the quality of service (qos) in software-defined networking (sdn) based cloud infrastructure," in *Information and Communication Technology (ICoICT), 2014 2nd International Conference on*, pp. 505–510, May 2014.

[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] D. Raumer, L. Schwaighofer, and G. Carle, "Monsamp: A distributed sdn application for qos monitoring," in *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*, pp. 961–968, IEEE, 2014.

[8] 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.

[9] COIN-OR, "Lemon graph library." https://lemon.cs.elte.hu/trac/lemon.

[10] 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.

[11] H. Mekky, F. Hao, S. Mukherjee, Z.-L. Zhang, and T. Lakshman, "Application-aware data plane processing in sdn," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, (New York, NY, USA), pp. 13–18, ACM, 2014.