Project Proposal

“FlowQoS: Not every flow is the same”

October 22, 2015

Group #20

Dhruv Sharma (A53101868)

Feichao Qian (A53091625)

Frederik Nygaard (U07002090)

Robert Jenkins (A53063069)

Introduction

In modern times, user devices connected to broadband access networks, run an assortment of applications that exchange network traffic with remote servers and other devices over the internet. However, traffic from one application might not share the same characteristics as the traffic emerging from another application. To a large extent, it is the end-user's requirements that play a significant role in deciding the nature of such applications and hence the kind of network traffic they send and receive. For instance, a user's expectation from a VoIP call, video streaming and gaming applications is that their experience remains seamless and high quality during their use of the application, requiring the traffic to be sent out at near constant rates along low delay and low loss paths over the network. Whereas, in certain use cases such as data backup to the cloud and system updates, it is expected that the operation is completed eventually even when the user is not actively using the application. Traffic from such applications, in contrast to VoIP and video streaming, does not come with any hard deadlines or network requirements.

While the nature of traffic varies so widely, the network devices today are, to a large extent, agnostic to such subtleties in the nature of network traffic and tend to handle it in the same way. Even though later kind of applications discussed above do not face any issues that directly affect the user, the former kind might be impacted severely due to the effect on network dynamics leaving the user with a sour experience. Research over past many years has resulted in the use of various metrics such as packet delays, jitter, available bandwidth, frame rate etc. to quantify user's experience in some way.

For this project, we focus on FlowQoS [1] solution that utilizes SDN and traffic policing in virtual switches to achieve network isolation in traffic flow types. We further provide an alternate implementation using Linux traffic control features to achieve the same goal with possible optimization in the use of available bandwidth.

Related Work

There is significant previous work for QoS in IP networks [2,3,4], traffic shapers [5,6], and traffic flows classifiers [7,8]. Kim et al. presented a solution that sets rate shapers at the edge switches and uses a QoS control framework for management of OpenFlow networks [9]. In contrary, FlowQoS does similar automated traffic shaping at a single gateway and offloads application identification to the controller. There are also many other related work [10,11,12,13,14]. They focus on other different aspects of the problem, while we focus on providing usable QoS for broadband access networks.

Other approaches to explore QoS in home networks, like Yiakoumis et al.   proposed letting users notify the ISP about their bandwidth needs for a given application, which transfers provision from the home to the ISP's last mile [15]. Georgopoulos et al. proposed an OpenFlow-assisted framework that improves users' quality of experience(QoE) in home networks, which is performing on each device instead of per-application or per-flow QoS [16]. Mortar et al. developed Homework [17], which allows users to monitor and control per-device and per-protocol usage, but it does not provide QoS support or perform any application classification.

System Design

FlowQoS implements traffic shaping at the first hop router in broadband access networks. The router, implemented as a custom hardware component using Raspberry Pi, embeds a pair of virtual switches (OpenvSwitch), one receiving ingress traffic to the router from the user device(s) while the other forwarding egress traffic towards the ISP network. For applying different traffic policing limits on different traffic types, the switches are connected multiple virtual links (veth pair), each dedicated to a certain class of traffic. The virtual switch ports on the egress switch use the rate limiting feature of OpenvSwitch which is based on the concept of token buckets similar to the ones used in Linux kernel implementations.

For classifying the network flow to a certain kind, FlowQoS uses the ingress virtual switch in OpenFlow mode to forward the first packet of every flow to an externally deployed SDN controller for analysis. The egress virtual switch, preconfigured with the policing rates, runs in the standard learning switch mode forwarding all traffic that it receives.

At the SDN controller, the flow classifier module analyzes any packet in two steps. First, it categorizes a packet as HTTP(S) based on the destination port being 80 (or 443) and later, for confirmation of the exact type, it matches the CNAME field from the packets it received during the DNS resolution phase before the flow started. If the flow doesn't qualify as an HTTP(S) flow i.e. web traffic, it uses the libprotoident library for analysis. The libprotoident library uses additional data from forward and return traffic packets such as the first few bytes and payloads of the first forward and reverse packets for the connection. Since, such analysis cannot arrive at a decision without delay, the flow is forwarded as uncategorized traffic type along one of the links and later rerouted to the right port once the confirmation is made.

Contribution

During this project, we plan to deploy the FlowQoS implementation of pair of virtual switches for improving QoS, within a virtual machine (emulating our end-host) with internet connection instead of the discussed hardware implementation. We plan to validate the results presented by FlowQoS using similar tools and scenarios that they use. Finally, we intend to develop an SDN controller and agent application that utilizes Linux's advanced routing and traffic control to implement the idea of FlowQoS while overcoming the limitation of under-utilization of available bandwidth. In addition to it (if time permits), we plan to use the NetFlow protocol supported by the OpenvSwitch to collect traffic volume statistics about traffic types and suggest (and possibly) automate optimal distribution of bandwidth share among them.

Experimentation Details & Project Milestones

Oct 30: Setup the test environment in the VM using OpenvSwitch pair and virtual links and deploy FlowQoS controller modules successfully to classify and forward traffic.

Nov 5: Get initial experimentation results for bitrate oscillation and throughput for the HTTP based adaptive streaming (DASH) using the DASH-JS player.

Nov 12: Get experimentation results for a RTSP based video streaming server-client pair (Gstreamer RTSP with VLC/openRTSP client) as well as round trip delay and jitter for a VoIP (Skype) call.

Nov 20: Setup required Linux Traffic control structures such as HTB classes and queuing disciplines in place. Implement/extend rate control module of the SDN Controller (POX) to direct rate control and flow re-routing commands to end-host agent.

Nov 25: Implement end-host rate control agent to receive commands from the controller over a separate TCP connection (non-OpenFlow) and set or update qdisc class rates as well as mark packets for a particular flow using iptables for the correct qdisc class forwarding. Begin structuring the poster.

Nov 30: Finish experimentation using similar test setup as earlier with new design to validate that it improves over FlowQoS by utilizing complete bandwidth in the absence of competing traffic from other types.

Additional milestone (if time allows): Collect traffic volume statistics from OpenvSwitch using NetFlow and automate reassignment of bandwidth limits based on past average usage per traffic type.

References

[1] Seddiki, M. S., Shahbaz, M., Donovan, S., Grover, S., Park, M., Feamster, N., & Song, Y. (2014). FlowQoS: Per-Flow Quality of Service for Broadband Access Networks. Proceedings of the Third Workshop on Hot Topics in Software Defined Networking - HotSDN ’14, 207–208.

[2] C. Aurrecoechea, A. T. Campbell, and L. Hauw. A survey of qos architectures. Multimedia systems, 6(3):138–151, 1998.

[3] D. McDysan. QoS and traffic management in IP and ATM networks. McGraw-Hill, Inc., 1999.

[4] P.Newman,W.Edwards,R.Hinden,E.Hoffman,F.C.Liaw, T. Lyon, and G. Minshall. Ipsilon flow management protocol specification for IPv4 version 1.0. 1996.

[5] Meraki Traffic Shaper. http://goo.gl/IL24ek.

[6] PacketShaper. http://www.bluecoat.com/ products/packetshaper.

[7] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. Blinc: multilevel traffic classification in the dark. In ACM SIGCOMM Computer Communication Review, volume 35, pages 229–240. ACM, 2005.

[8] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield. Class- of-service mapping for QoS: a statistical signature-based ap- proach to IP traffic classification. In Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, pages 135–148. ACM, 2004.