Performance Measurements of an SDN Topology

Giovanni Pica
Department of Computer Science, La Sapienza Università Di Roma
pica.1816394@studenti.uniroma1.it

Abstract

SDN (Masoudi and Ghaffari, 2016) is the achronymous of the Software Defined Networking that is a structure designed for simplifying and improving network management with high flexibility by splitting control plane and data plane with the usage of a remote controller (e.g. OpenFlow). In this report there are some measurements concerning the Link Utilization, Latency and see what happens when the packets have a particular rate that could saturate some link.

Keywords: Ryu, Mininet, Utilization, Latency.

Introduction

In this report there are some measurements with some plots of two particular metrics that are Link Utilization and Latency as the λ incoming rate increases. The topology shown in 1 has been made with a python script and the command "*sudo mn –custom /path/to/topology.py*" and also have been added other flags such as "–controller remote –switch ovsk –topo topo –link tc,bw=10" to set respectively the controller, the switcher as Ovs, the topology based on the script and the link characteristics such as the bandwidth. And also I used MiniNAM (Khalid et al., 2017) that is a tool very useful to monitor where the packets go because it offers a GUI like miniedit of mininet but with the animations of the packets.

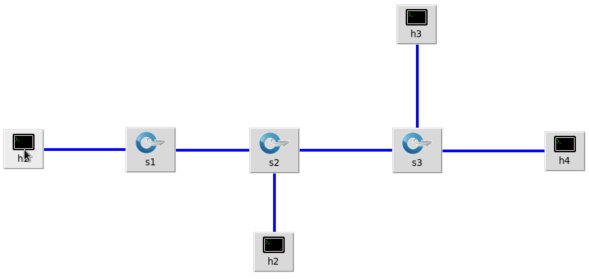

Figure 1: Topology.

After some measurements the best bandwidth for this project was 10Mbit/s as shown on the Plot 2. The test has been maded in combination with some values of the bandwidth (1Mb, 5Mb, 10Mb), the delay (0, 1ms) and the queue size (10, 20) with a total of 12 tests. The ones with the delay setted to 0 are faster (7 to 12 tests) and so I decide to take the one with the larger bandwidth and I decide also to not consider the queue size.

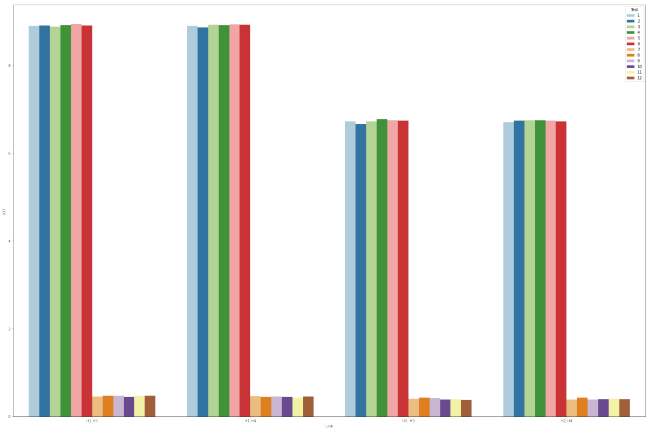

Figure 2: Plot of the best Bandwidth.

The simulation of the network is based on two parts, the mininet that provides the client to ping the packets between hosts and Ryu that is the server part with the command "*ryu-manager –observe-links /path/to/someswitch.py /path/to/gui_topology.py*" and the gui_topology is useful to see the installed flows on the switches. The installed flows are important because with those flows can be seen the byte_count to calculate the utilization with the formula:

$$U = \frac{\frac{byte\_count}{ping\_time}}{link\_bandwidth}$$

And for the latency can be taken the average RTT when the ping finish. The approaches that have been followed are:

- Simple Switch of the Ryu Application when the topology is the default topology in Figure 1.
- Spanning Tree (more details here https://osrg.github.io/ryu-book/en/html/spanning_tree.html) always in the Ryu Application when in the topology has been added the link (S1,S3) Figure 3, that is based on disconnect logically a port to avoid loops.

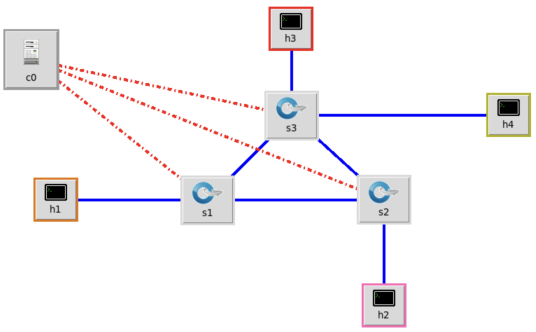

Figure 3: New Topology.

- Load Balancing based on alternative paths after some received packets, in this particular approach has been used the NetworkX (Hagberg et al., 2008) library to build a graph with the topology shown in Figure 3. The measurements have been taken with a ping of an half of the total packets and after that another ping after a "*ovs-ofctl del-flows*" because in this way the switch refresh the flows and learn a new path.

Some important characteristics of the packets:

- Packet size is 1000B, so in the ping command a flag "*-s972*" has been added because ping has 28B of header.
- Total packet number on all of those tests is 20 and is setted with the flag (always after the ping) "*-c20*".
- The λ rate is setted with the flag "*-i1*", in this example the interval is 1 second so the number of Bytes trasmitted in a second is $\frac{1pkt}{1s} = 1\frac{KB}{s}$ because packet size is 1000B.

Results and Comments

In this section are discussed the main results and performances of all of these approaches. First of all let's talk about the latency in the Plot 4.

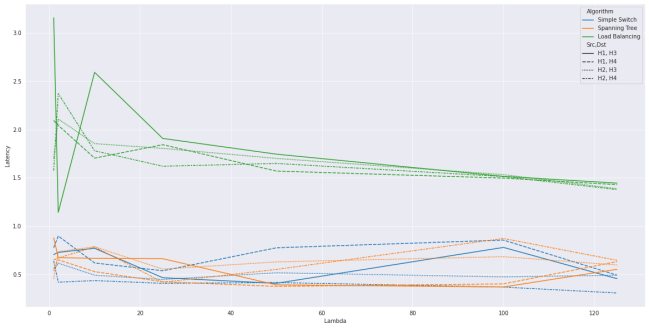

Figure 4: Latency expressed in milliseconds.

As can be seen the latency when the networkx library is used is worse than when is not used, because the switch will handle packet-in to redirect messages without flow-rules installed so the RTT will be higher. So when the new link has been added can be seen a worsening of the performance based on the latency. Now let's talk about the utilization that is the most interesting one. The utilization is measured end-to-end from h1 to h3 and only for the spanning tree to measure S1,S2 utilization h1 to h2. In the Plot 5 can be seen that the first two approaches will saturate the links for example S1,S2 and as can be seen the Spanning Tree doesn't solve the saturation problem because it uses only one link for example from h1 to h3 it uses the link S1,S3 and S1,S2 is never used. So in this particular approach it uses only one link because for example it has deactivated the link S2,S3 so it never pass on the path S1,S2,S3. But with the Load Balancing approach the link utilization is halved so it's a nice idea. Because the main idea of this algorithm is to alternate paths and not use only one of them.

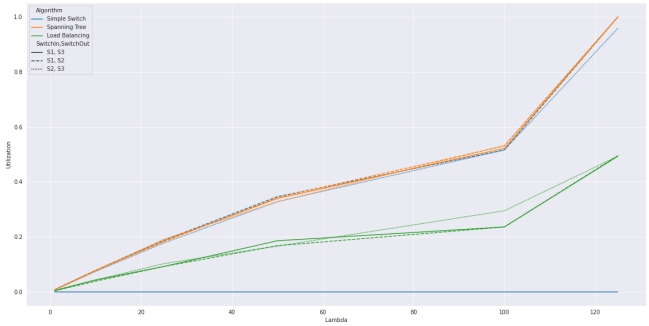

Figure 5: Utilization.

Conclusions and Future Works

I tried to implement a dynamic way to delete flows without the usage of the del-flows command of ovs and it works but the main problem is that each time the switch has the table flow empty and the RTT is very high like 40ms in average. A nice future work is to implement a Load Balancer Switch faster than this and also another good thing to do is to alternate the deactivation of the link in the Spanning Tree algorithm because in the Ryu implementation the link down doesn't work.

References

Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using networkx. In G. Varoquaux, T. Vaught, & J. Millman (Eds.), *Proceedings of the 7th python in science conference* (pp. 11–15).

Khalid, A., Quinlan, J. J., & Sreenan, C. J. (2017). Mininam: A network animator for visualizing real-time packet flows in mininet. *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, 229–231.

Masoudi, R., & Ghaffari, A. (2016). Software defined networks: A survey. *Journal of Network and Computer Applications*, *67*, 1–25.