Dynamic Flow Steering Application For SDN

Project Advisor: Professor Jonathan Ponniah

Electrical Engineering Department

Poddutoori, Puneeth (MS ElectricalEngineering
Nangunoori, Shivani (MS Electrica lEngineering)

Introduction

We have routing protocols that determine the shortest path to send traffic to the destination. This shortest path may not be suitable for intended customer applications which are delay-sensitive, while we have other routing paths available that are not utilized efficiently. Even if the performance is high in the beginning, the quality of experience at the customer end may get deteriorated from time to time.

To realize full potential of centralized control plane in Software Defined Networking (SDN), and to maximize the network resource utilization and offer new differentiating services, it is required to have an intelligent flow configuration module that dynamically reacts to network conditions. An application which grasps the real-time dynamics of the network can help the controller to make the most correct routing decisions in real-time. Dynamic flow steering application which runs on top of SDN controller, monitors the end to end and hop by hop network paths and reconfigure the flows in the flow tables in real time to meet the network demands.

The controller enables us to have centralized view of the network. The application gets the link delays by making use of additional TLVs in LLDPDU packets. In the event of PacketIn (incoming packets), the application uses the link delays to calculate the shortest path at that time and updates the flow tables in the routers. Also, the flows are updated frequently according to the real-time conditions.

.

Methodology

This project contains 3 phases:

1. Update LLDPDU packet format.
2. Link-Latency Measurement.
3. Get Shortest Paths and Update Flow Tables.

Phase 1 - Update LLDPDU packet format:

LLDP packets are used by network devices to get the neighbor information, their capabilities, etc. Also, it can be used to send and receive additional information such as time at which packet is generated.

The LLDPDU in the LLDP packet is modified by adding **Timestamp** field as an optional TLV (Type Length Value field) which can be used to calculate the delay. Figure 1 shows the format of LLDPDU.

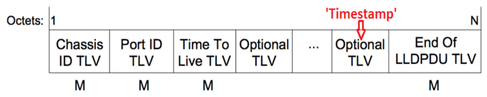

Figure 1 – LLDPDU packet format

Methodology

Phase 2 - Link-Latency Measurement:

The Mechanism for measuring link delay is described below.

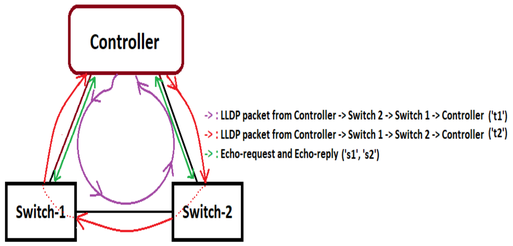

Figure 1 – Latency Measurement Mechanism

In order to get delay of each link in the topology, let us consider two switches which are directly connected namely switch-1, switch-2 and to measure the delay of the link connected to these switches, the SDN controller sends modified LLDP Discovery packet to switch-1, and is forwarded to switch-2 and then back to the controller. The difference of the time (when LLDP packet is received) and the time when it was generated (which can be obtained by parsing the packet) is stored. let' **'t1'** be the anti-clockwise roundtrip delay. In the same way we calculate the roundtrip delay (**'t2'**) in clockwise direction from switch 2 to switch 1 to controller.

At the same instant, controller sends ECHO request packets to switches and receives the echo reply packets from respective switches. And the difference in time of echo request generation and echo reply reception is **'s1'** for path controller<->switch1 and **'s2'** for path controller<->switch2. Link delay is measured using the formula below.

$$\textbf{Delay} = \frac{1}{2}(t1 + t2 - s1 - s2)$$

Phase 3 - Get Shortest paths and update flow tables:

The Underlying Network Topology information is obtained from the controller using APIs. The delays calculated for each link in the network are now stored in graph data structure using the topology (switches and links) information. And a dictionary, which takes end switches' dpid(s) as key and delay of the link as value is initialized.

To calculate shortest paths, we used python-networkx module which has a function that takes graph, source switch, destination switch and real-time delay metric (weight) as inputs and returns all possible paths to a list, the shortest path being the first one. We have included a new parameter **'k'** which can be used for load-balancing. We pick first **k shortest** paths from the list. In this way all possible paths from every switch to all other switches are stored in a dictionary.

In the event of 'incoming packet' the application uses this dictionary and instructs the controller to push (or) update the flow tables of all the switches involved in the (shortest) path. The Latency between links and shortest paths will be calculated frequently for every 10 seconds, though it can be customized to avoid instability.

Analysis and Results

We have implemented the project using RYU SDN Controller & Mininet. The application is run on the Controller. We tested it on the topology shown below:

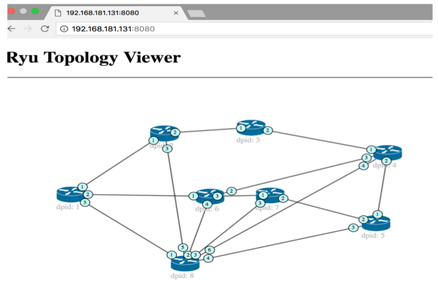

Figure 3 – Network Topology

The latency of all links are calculated using LLDP packet and topology information and are stored in graph. Using delay (as weight) and k = 1, the k-shortest path(s) are obtained for a source and destination and the flows are pushed by the controller to all switches involved in the path.

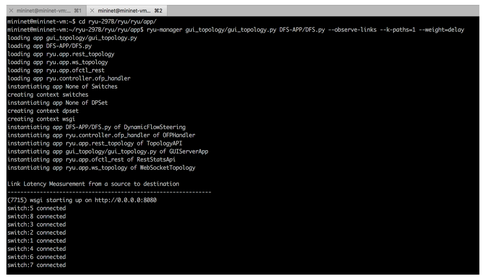

Figure 4 – DFS App run on controller

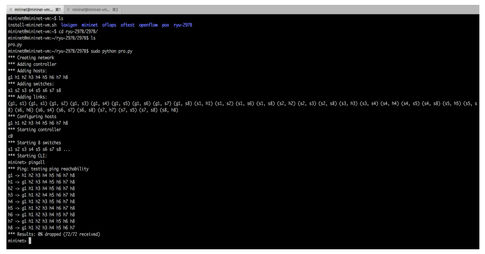

Figure 5 – Run Mininet to Emulate a network

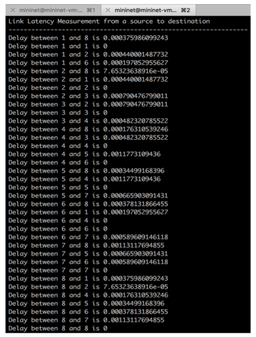

Figure 5 – Link Latency

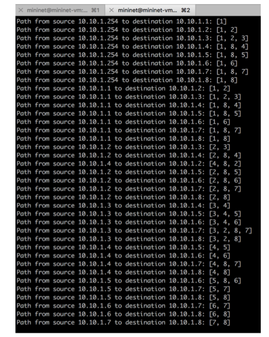

Figure 6 – shortest paths

Conclusion

In this project, we have implemented a SDN application which routes the traffic dynamically based on real-time delay of the links. It's intelligent flow configuration module makes it capable of reacting to real-time network conditions. It maximizes the network resource utilization and improves quality of service to the end users.

Key References

[1] Application Layer Flow Classification in SDN by Hamid Farhadi, Akihiro NakaoDate Published:02 December 2013 Publisher: IEEE

[2] Improved Video Throughput and Reduced Gaming Delay in WLAN Through Seamless SDN-BasedTraffic Steering by Wei Guo, V. Mahendran, Sridhar Radhakrishnan. DOI: 10.1109/CCNC.2017.8016293

[3] Traffic Engineering in Software Defined Networking: Measurement and Management by Zhaogang Shu, Jiafu Wan, Jiaxiang Lin, Shiyong Wang. DOI: 10.1109/ACCESS.2016.258274

[4] In IEEE Standard for Local and metropolitan area networks— Station and Media Access Control Connectivity Discovery, Link Layer Discovery Protocol(LLDP, IEEE 802.1AB and IEEE 802.3-2012 section 6 clause 79)

Acknowledgements

We would like to thank Prof. Jonathan Ponniah for his constant assistance, expert advice and encouragement through out the course of this project. He has helped a lot in understanding dymanic routing protocols and how dynamic metric exactly works. Also, he helped us in getting deeper by challenging us at each phase of the project and bringing the best out of us.