Applied Mechanics and Materials Vol. 610 (2014) pp 954-958
Online available since 2014/Aug/11 at www.scientific.net

doi:10.4028/www.scientific.net/AMM.610.954

Submitted: 21.05.2014
Accepted: 26.05.2014

Dynamical Traffic Engineering in Software-Defined Network

Yifan Yu, *Yong Li, Depeng Jin

Department of Electronic Engineering, Tsinghua University, Beijing 100084 China

*liyong07@tsinghua.edu.cn

Keywords: traffic engineering, software-defined network

Abstract. Software-Defined Networks (SDN), as newly proposed network architecture, has a great potential in optimizing network traffics. In SDN, the control plane is separated from the data plane. With the help of the centralized controller, we can gather information of the network in real time. In this work, we propose a practical two-stage approach for traffic engineering that takes advantages of SDN. The approach not only assures every newly injected flow gets a suitable route that does not have too much payload on it, but also schedules the overall flows so that they are distributed more equally in the network. Furthermore, we demonstrate its efficiency in terms of port speed and compared it with port speed under the default routing decision. We also use linear programming to find the optimal solution and compare it with our result.

Introduction

The Internet users have been increasing explosively in recent years, which left a huge amount of traffic on the worldwide network. However, the capacity of network links did not increase as fast as the traffic did. Consequently, the Internet Service Provider (ISP) failed to provide a network that is fast enough to satisfy the great need of their users. One alternative is that the ISP raises the Internet speed by exploiting the current links more efficiently. Traffic engineering is a way to solve this problem. It puts every flow in the network on the links systematically. As a result, flows will be transmitted at a high speed, which improves network utilization and enhances user experience.

However, there exists two difficult points in traffic engineering.

1. The route of a packet is decided by the status of part of the network, which usually leads to a locally optimal route rather than a globally optimal one.

2. The route of a packet is barely decided by the topology of the network, which means if the traffic changes dramatically, the network will fail to attain an overall load balance.

To solve the first problem, we use SDN as our network architecture. The switches in SDN forward packets according to its flow table, which are totally controlled by the controller. The controller is able to find the best path for each packet as it has the overall topology of the network.

We use a dynamical approach to fix the second problem. The controller will fetch port and flow statistics from switches routinely. Then we can calculate the speeds of ports and flows based on these real-time statistics and decide the route of packets using the speed information.

To demonstrate our approach's efficiency, we build a SDN network using floodlight controller and change the controller to implement our approach. We compare our result to the situation of using floodlight's own forwarding module. Given that we can use linear programming to calculate the optimal distribution of flows, we also compare our result with the optimal result.

Problem Describe and Linear Programming Solution

Consider a network with a topology *G (V, E)* contains a set of flows *F*. Each flow has a number of loop-free routes from its source to its destination. The problem is to find an optimal path for each flow so that the max link utilization can be minimized.

Previous work [1] has discussed similar problem and has given the detailed formulations. The constrain is the link utilizations and the variables are the flows along each link. We can use linear programming to solve this problem.

The problem of the linear programming is that it takes too long to calculate the optimal path for each flow, which may lead to its result no longer suit the current network because the flow speeds are changing. In this work, we propose a heuristic algorithm that has a dynamic attribute. The algorithm can respond to flow speed's change rapidly.

Software-Defined Approach for Traffic Engineering

Every packet in the network has its source switch and destination switch. The controller's task is to pick an optimal route from all the routes between these two switches. Every route contains a list of switch port tuples. We define each switch port tuple's utilization as the sum of transmit speed and receive speed divided by its max speed. Then we classifies all the routes into a certain numbers of levels, like *L1*, *L2* and so on, based on the max utilization of the route's switch port tuples. For example, if the max utilization is under 50%, the route is a *L1* route. If the max utilization is between 50% and 80%, it is a *L2* route and so on. We can set more levels if we want to describe the routes more precisely.

In order to gain the real time speed of ports and flows in the network, we set a repeatedly executed task that fetches the port and flow statistics from each switch. We can calculate the real time speed using the reports returned to the controller.

In our method, we separate our algorithm into two stages. The first stage is called New Flow Injection. For each newly set up flow, if it is a TCP or UDP flow, we will give the new flow a proper route in this stage. The second stage is called Overall Adjustment. Every flow that has been deploying in the first stage and its route will be saved in our database. We will seek a more optimal route for each flow in the database. The second stage will be executed routinely to keep up with the new status of the network.

New Flow Injection. The first stage deals with the newly injected flows. First, we find all the routes between the source switch and the destination switch. Then we classify these routes into different levels based on their max utilizations. We begin our search from the lowest level *L1*. If there exists at least one *L1* route, we choose the route that has the lowest average utilization in *L1* level to deploy the new flow on and save our decision to the database. If the set of *L1* route is empty, we raise our level to *L2* and do the same choosing. We repeat this step until we have raised the highest level. The detailed process of this stage is showed in Figure 1.

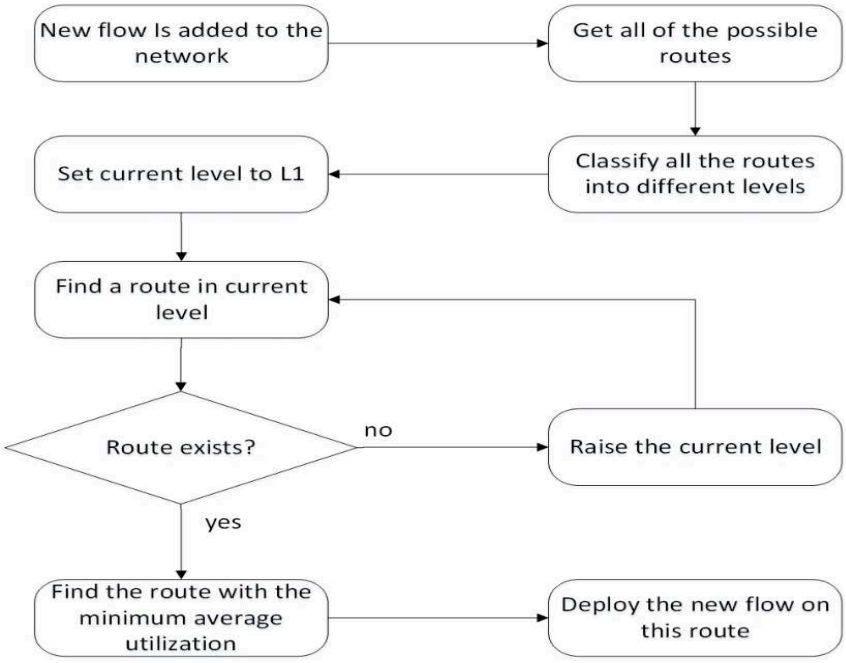


Figure 1. New Flow Injection

Overall Adjustment. The second stage rearranges all the flows that saved in the database. For each flow in the database, we first calculate its route level and current speed. Then we find all the routes between the source switch and the destination switch. Since we did this step in the first stage, we can use the result from it and do not need to calculate the routes again. Then we classify these routes. But this time, we add the flow's current speed to a route when we calculate its max utilization. Then we do the same search as the first stage did but we set our highest level to the flow's current level. At last, we update the database with the new decision. The detailed process of this stage is showed in Figure 2.

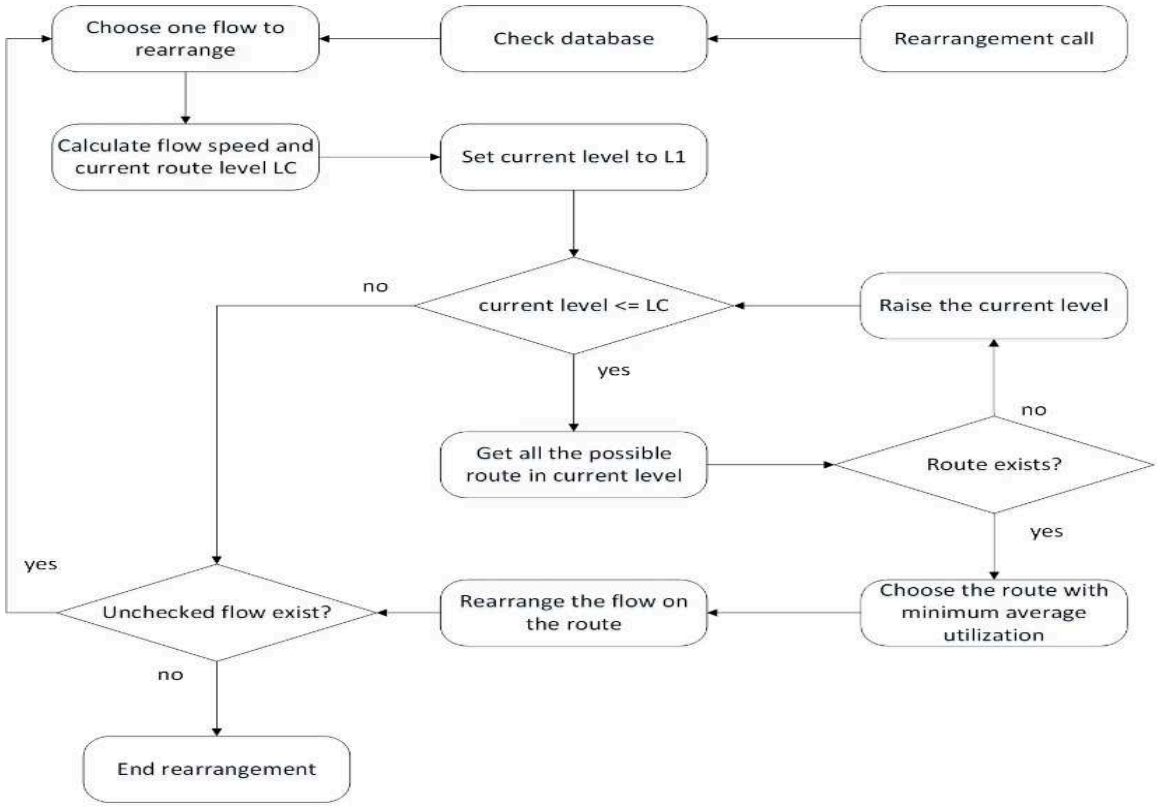


Figure 2. Overall Adjustment

Experimental Results

We built a SDN to test the performance of our approach. We used floodlight as the controller of SDN, NetFPGA and OpenvSwitch as the switches of the network. Figure 3 shows the topology of our network. On each host, we installed libnet, a software that can send packets in a high speed, to simulate crowded network. The host1 and the host3 are the sending host and the others are the receiving host. On each sending host, we set up one dynamic flow and two static flows. Each receiving host also receives three flows, one is dynamic and the others are static.

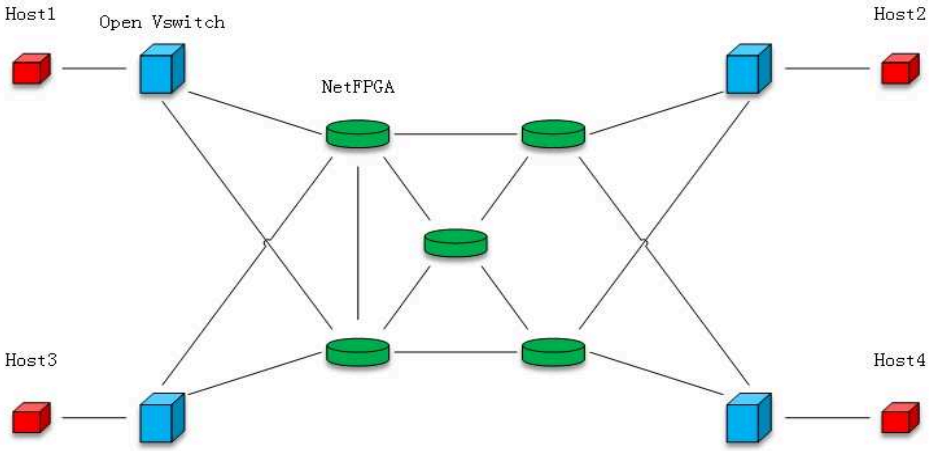
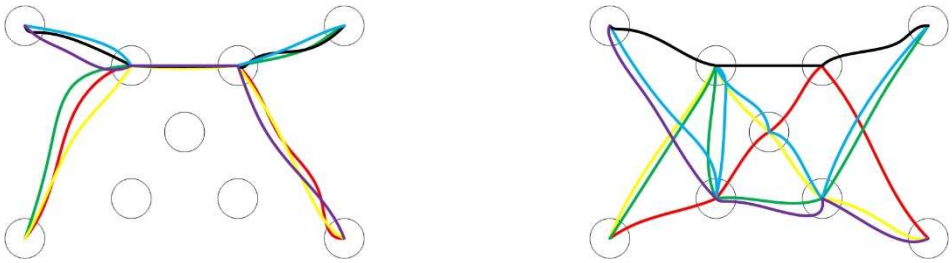


Figure 3. Network Topology

We take a few steps to test the performance of our approach. First, we do not add the traffic engineering module when we launch the controller. The packet's route will be decided by floodlight's default forwarding module. Then we begin to send 6 flows from host1 and host3 and meanwhile we take down the route of each flow and the speed of each port in the network. After this, we reboot the controller with the traffic engineering module. We send the same flows and take down the routes and port speeds as well. Figure 4a shows the route when we used the floodlight forwarding module. Figure 4b shows the routes after we used the traffic engineering module.



(a) Using Default Forwarding (b) Using Traffic Engineering

Figure 4. Flow Distribution

Figure 5 shows the average speeds of each switch port tuple under the situations of using default forwarding, using our proposed traffic engineering and using linear programming traffic engineering. The difference of speeds between each port decreases after we add traffic engineering module.

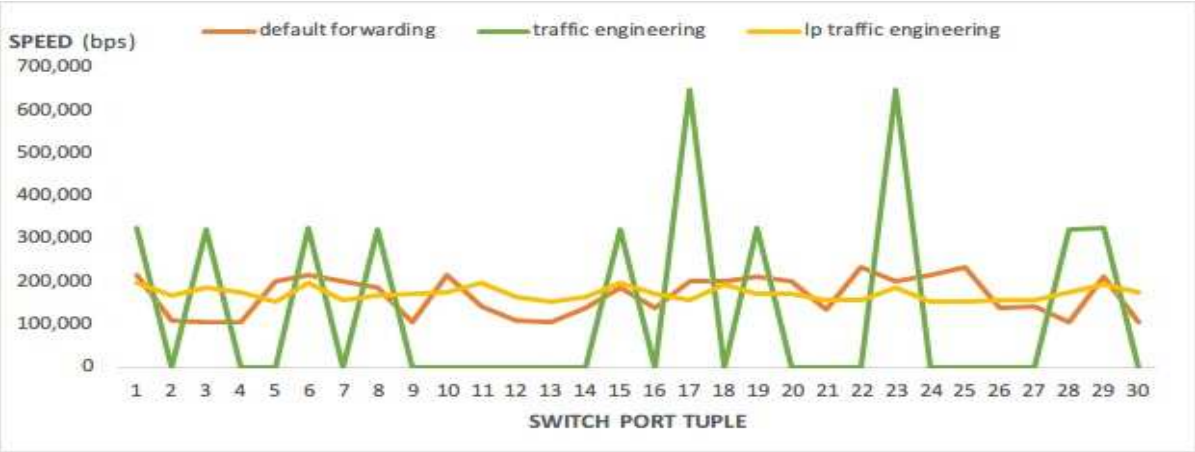


Figure 5. Port Average Speed

Figure 6 shows the standard deviations of port speeds under these three situations. The standard deviation stands for the degree of the equality of link utilizations. Flows are distributed more evenly if the standard deviation is smaller.

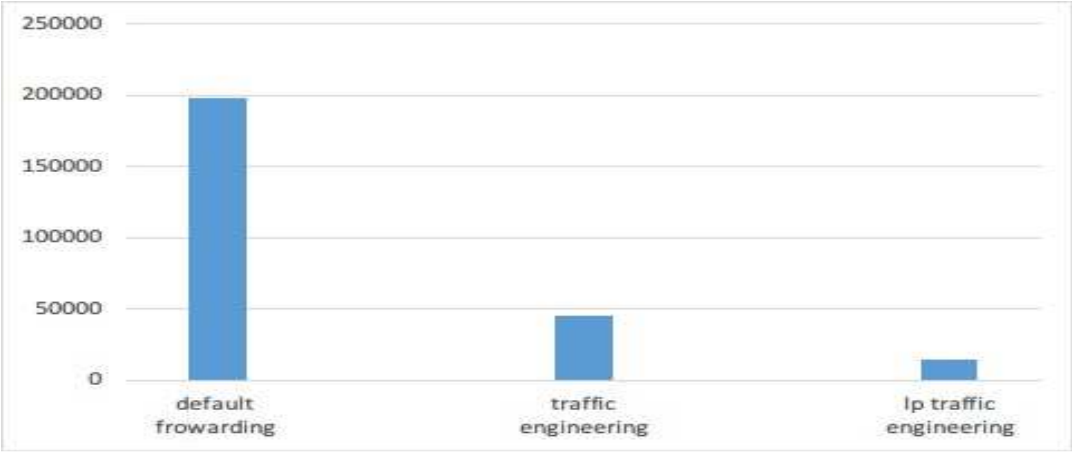


Figure 6. Standard Deviation

We also use iperf to test loss packet rates. We inject 4 flows which have a speed up to 100M and then use iperf test loss packet rate in different speeds. Figure 7 shows the loss packet rates of using and not using our approach.

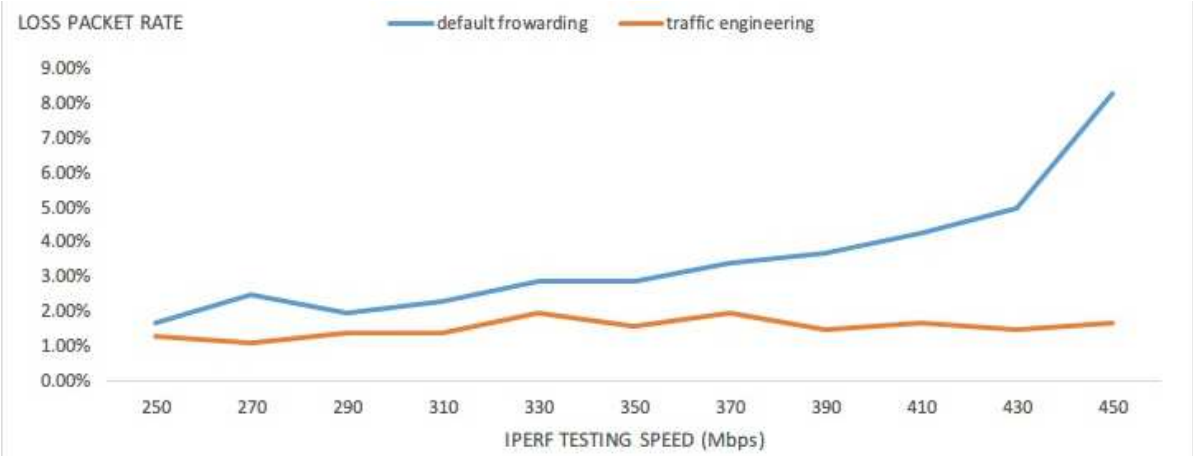


Figure 7. Loss Packet Rate

References

[1] Heller, B., Seetharaman, S., Mahadevan, P., Yiakoumis, Y., Sharma, P., Banerjee, S., & McKeown, N. 2010. ElasticTree: Saving Energy in Data Center Networks. In NSDI (Vol. 3, pp. 19-21).

Mechanics, Mechatronics, Intelligent System and Information Technology
10.4028/www.scientific.net/AMM.610

Dynamical Traffic Engineering in Software-Defined Network
10.4028/www.scientific.net/AMM.610.954