**Active Virtual Circuit**

Husam Omar & Tariq Issa

CS Department, An-Najah National University

**ABSTRACT**

This paper presents new Software Defined Networking (SDN) Application. This application achieve the Active circuit path between two hosts with a specific bandwidth. The application relies on the Software Defined Network paradigm, the new approach in the networking world emerged to attain network programmability. This approach separated routing software and algorithms from network physical devices, a centralized control plane allows network operators to deploy advanced control and management strategies. The issue that the application solves is the limitation that caused by the traffic from the huge amount of users on the network. The solution makes use of the Dijkstra shortest path algorithm, to calculate a path that achieves specific bandwidth requirements between two hosts.

**1.INTRODUCTION**

Today computer networks are big, complicated, hard to manage and control. There are multiple computer networks equipment vendors, each have its own network hardware and software, which its source code and blueprints are copyrighted property in the commercial companies, in some certain cases the network operators have to change the way the networks acts to fulfill some requirements and achieve specific performance, which is hard to do in real life IP networks.

In spite of accomplishing a programmable networks, many approaches introduced and initiatives put forward, some of them stopped, other continued to develop. A solid example is software-defined networks (SDN), its widely adopted by industry enterprises.

SDN decoupled the control plane from the data forwarding plane, which are tightly coupled in the IP-core network devices, introduced centralized controlling were one controller manges multiple switches and make

In this paper we are implementing an Active virtual circuit between two hosts with a given amount of bandwidth to be reserved, the purpose of the project is to have high speed connection between two specific hosts. Moreover, it will reserved the best path that achieve the bandwidth requirements by using our algorithm.

This paper is organized as follows: section 2 contains related work, section 3 is about our working principle, while section 4 presents the results of our work, and section 5 concludes the paper

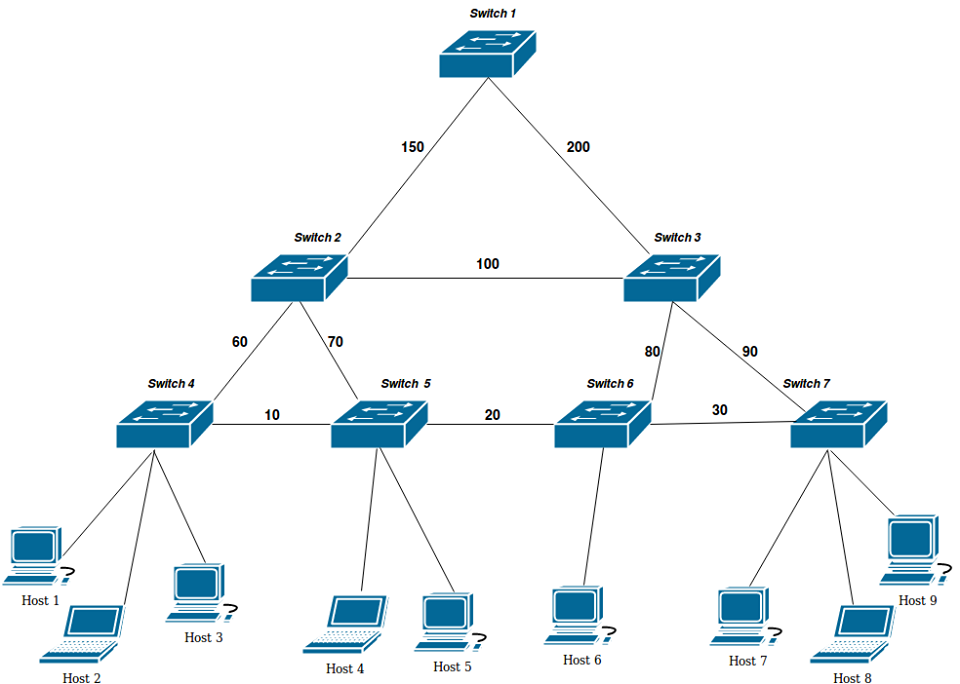
**2. RELATED WORK**

There have been some researching done in the field of prioritizing a connection and implementing QoS services.

Tomovic et al. [\*] .In there paper ‘SDN control framework for QoS’. Were working on an SDN control system that enables service differentiating and an automated way of network resources monitoring and reservation. There work consisted of four modules: resource monitoring, route calculation, call admission control and resource reservation. These functions works altogether to provides bandwidth guarantees for priority flows.

Davide Adami et al. There paper ‘Towards an SDN Network Control Application for Differentiated Traffic Routing’ was about creating an SDN application to control routing according QoS requirements. They introduced Deep Packet Inspection (DPI) module that detects the session starting message of the Session Initiation Protocol (SIP), alongside three other modules: Topology Discovery module, Traffic Statistics Handler module, Routing Engine module.

**3. WORKING PRINCIPLE**

Figure 1: Used network topology

**3.1 The Network**

To create a network, we used the mininet. It is a network simulation tool that provides virtual network devices, in the SDN we use the OpenFlow switch, the mininet provides the Open vSwitch that supports the OpenFlow protocol, it is an OpenFlow enabled switch, so we can make use of it to install flows entries on the flow table.

The topology we used is a tree like mesh network topology were every switch is connected to its parent and sibling, the topology built of seven Open vSwitches and seven hosts (Mininet virtual machines), the links between the switches have different capacities, and between each host and the corresponding switch connected to it is not set, so it will be defined regarding the machine resources, that is because we don’t want to get bottlenecks as our work is in the switches part of the network.

The SDN controller we use is Floodlight controller, it provides REST APIs that we use to control the traffic, take statistics and push flow entries to switches.

**3.2 The algorithm**

The algorithm starts with asking the user for the IP address of the start host, the IP address of end host, the bandwidth that the virtual circuit should fulfill. Using the APIs we find the switches connected to these hosts. Then we represent the switches part of the topology as graph using adjacency matrix representation, where the vertices are the switches, the edges are links between switches and the costs of the edges are the links capacities. After that we run Dijkstra shortest path algorithm, with an edit that the chosen edge is equal or greater than the bandwidth defined by the user. The Dijkstra algorithm takes a source node (the start switch) and calculates shortest paths to all other nodes, when the algorithm reaches the end switch, we terminate the execution and take the path of switches returned.

After that, we install the flow entries on the switches, each switch gets to entries. For example: if Dijkstra algorithm returns the path is switch number four (s2) and switch number five (s5), using some API we find the port in s5 that connects it to s6, and the port that connects it the start host, then we install a flow entry on s5, this entry checks if some packet in the switch pipeline with source IP matches the IP of the start host, and destination IP matches the IP of the end host, the switch forward the packet to the port that connects s5 to s6, the other flow entry we flip the source and destination IP address and forward the packet to the port that connects s5 to the start host. When done of installing the flow entries the virtual circuit is ready.

**4. Results**

The testing scenario is to measuring the bandwidth between the two hosts before and after running our algorithm, with and without traffic across the network.

First after creating the network in Mininet, while the still there is no traffic in the network, we use a utility called “iperf” that measures the TCP bandwidth between two host, our two in experiments are h1 and h7, the results shown in Table 1 (experiment 1) , then we generate traffic using “ping” utility, re-use “iperf” (exp. 2), then run our algorithm with bandwidth set to 50, re test (exp. 3)and finally, we delete the flow entries and the run out algorithm again but we set the bandwidth to 15 (exp. 4)

|  |  |  |  |
| --- | --- | --- | --- |
| Experiment | Bandwidth (Mb) | Download (Mb) | Upload (Mb) |
| 1 | - | 9.47 | 13.0 |
| 2 | - | 9.53 | 11.8 |
| 3 | 50 | 56.7 | 68.1 |
| 4 | 15 | 19.0 | 25.1 |

Table 1: results of experiments

**5. Conclusion**

An Active Virtual Circuit for two hosts Implementation was presented in this work, in which

QoS interaction between video applications and DiffServ-enabled networks was considered. The

proposed RPS plays a good bridging role in enabling the network to be application-aware and

provide better end-to-end video quality by using the same network capacity. The performance of

content-aware differentiation has been demonstrated in terms of the packet drop probability and

the throughput. Experimental results showed that the proposed framework has enhanced the end-

to-end video quality at the same packet drop rate.