PROTOCOL BASED SWITCHING IN SDN

Anil Kumar Reddy Kona, Noorinbano Shaikh, Prof Dr. Chao-Li, Tarng

Department of Electrical Engineering, San Jose State University, San Jose, CA, 95112

SAN JOSÉ STATE UNIVERSITY

Introduction

The Traditional networks were designed to forward packets from source to destination using the shortest route possible. Routers and switches were mostly agnostic to the applications being served by the network. Today's networks are evolving to be more application aware that changes the user experience for the better and reduces operational costs.

This project shows how the Software-defined network (SDN) architecture allows service providers to build networks with increased application awareness, which can be built into the network by developing SDN controller applications that keep track of application-level characteristics and use that intelligence to provision flow into the network switches.

When the client transmits an IP packet, the switch inspects the packet and depending on the policy/rule installed in it, forwards the packet on a particular route. If the switch doesn't have any policy/rule installed, it sends the packet to the controller. The controller inspects the packet header and/or the payload, determines the type of packet (TCP, UDP, HTTP, etc.), and installs a policy/rule on the switch instructing it to forward packets along a particular route

System Requirements

Our proposed system works on a c++ and would require a controller that is c++ compatible. We would require an SDN architecture layout with virtual Switches and Hosts. The hosts would have to be able to send numerous kinds of packets representing various kinds of layer 4 to layer 7 protocols.

The c++ program would have to have cases where it can handle each of these protocol packets using the intelligence in the controller. We would also require a network topology creation software such as Mininet where we may create our network design with various possible routes to the destination

We would require a protocol to connect the SDN topology created on Mininet with the controller so that the controller can give switching guidance to the switches on the network.

Lastly we would require to generate many type of packets at one end which can be sent by the sender to check if the controller code functions properly.

Technologies Used

1.Software Defined Networking:
SDN is the latest technology in the networking world and it is our belief that it is the technology that will overhaul all other technologies in the near future. With SDN we can replace the components such as routers and switches and create a mere software that functions like these devices.

Function:
Whenever a packet arrives at a forwarding device in SDN the device either already has an update as to what action has to be taken on the packet or must consult the controller for instructions, The forwarding device is enabled with the ability to cache the instructions of the controller so that it need not consult the controller for instructions for similar packets. The cached information expires after a while or is updated by the controller

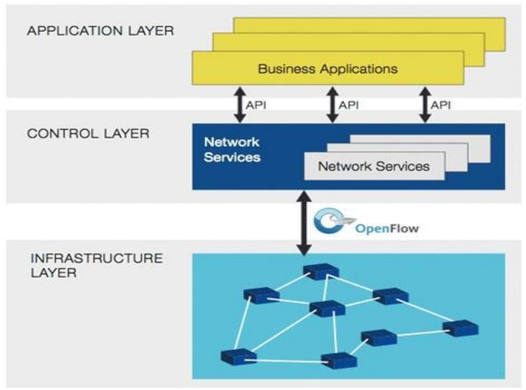

Figure 1. SDN Architecture

2. Open Flow:
This is an open standard that works at the interface between the controller and the forwarding device such as the switch. It is usually a feature present on commercial network devices such as switches, routers etc. It is presently being used by most vendors.

3. POX Controller:
The inbuilt controller in Mininet which is the main platform for network device layout creation is POX. It basically works to support programs written in Python for the Network Control Software .It works on the Open flow protocol.

4. Mininet :
It is a software used to create a virtual topology of a network consisting of switches and hosts using simple commands. The switches are controlled by a controller that makes all the switching decisions based on the program fed into it.

Project Design

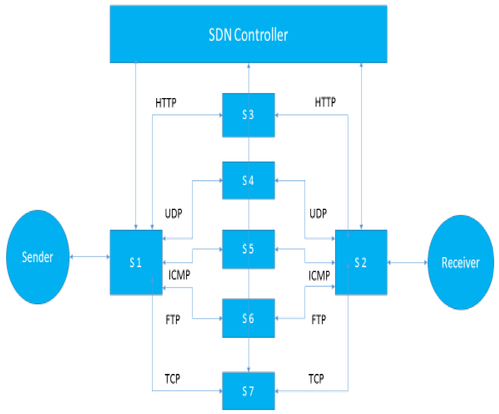

Figure 2. Project design.

Implementation

Our implementation comprises of a code which performs switching on the network topology we created on the mininet.

The switching is performed on packets on the basis of the contents in the header of the packets .Since different type of protocol packets have different fields in their header we can differentiate between the type of protocol used or the application the protocol belongs to on the basis of the contents in its header.

The controller identifies HTTP packet with its destination port 80, UDP packet with its protocol number 17, ICMP packet based on its protocol number 1, FTP packet with its destination port 21, TCP packet based on its protocol number 6 and routes the packets through switches S3, S4, S5, S6, S7 respectively.

Only the first packet of a particular type of flow goes to the controller and the controller configures rules for these packets on the vswitch. Once the rules are configured on the switch, the packets directly go through the previous path for these packets.

Results

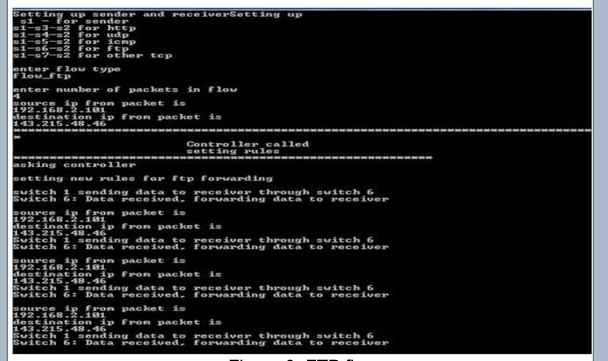

Figure 3: FTP flow

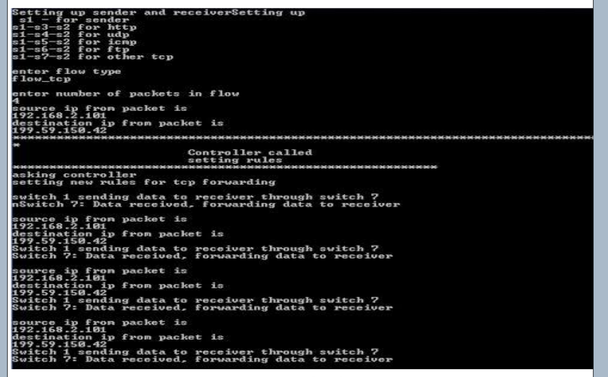

Figure 4: TCP flow

Conclusions

We can conclude saying that the implementation is a prototype of how an application aware system would route packets. Such an implementation is not yet available in the SDN environment and a with its implementation a great deal can be changes would come about in the way packets are switches/routed. With the background of our code and many additional features we can accomplish routing that is more intelligent and gives excellent performance networks

Key References

[1] "Software Defined Networking" by Thomas D. and Ken Gray. Ebook URL: http://pro-networking-h17007.external.hp.com/images/solutions/technology/openflow/sdn-architecture.png

[2] "Monitoring, Managing, and Securing SDN Deployments " A white paper by Gigamon inc., URL:http://www.gigamon.com/stuff/contentmgr/files/1/9de51bbdfa058aaab338aaaeac76956c/download/wp_monitoring_managing_securing_sdn_deployments_3106.pdf