ENHANCING NETWORK AVAILABILITY USING SOFTWARE DEFINED NETWORKING

Sathish Manoharan, Mr.S.Saminathan*
M. Tech Student, Assistant Professor, SRM University, Kattankulathur, TamilNadu, India.
Department of Computer Science Engineering, SRM University, Kattankulathur, TamilNadu, India.
Email: samirmd007@gmail.com

Received on 04-08-2016

Abstract

In Traditional Networks the routers always calculate the shortest path for forwarding the packets. Availability of a service provider’s IP network can be thought of as the reach ability between the regional points of presence (POP). The Key Performance Indicator of the network is the availability of every POP to every other POP. During the network outages, the routers exchange route updates and takes additional time to converge. This process can be heavy when the network outages are frequent and bandwidth is limited. Whereas in Software Defined Networking, Controller enhances the network availability by knowing the global view of total resources available and so alternates the path without causing any network overhead. A centralized controller can compute the routes avoiding frequent route updates and hello packets. It also eliminates redundancy links, advanced middle boxes and separate load balancing equipment. So, the objective of the paper is to build a SDN router in place of traditional router and then enhance the network availability using SDN solutions such as SDN Security, Load Balancing, Multiple Controller Management and Policy Based Forwarding Approaches.

Keywords: Software Defined Networking, SDN Applications, Control plane vs Data plane, Real time SDN implementation, SDN Migration, Open-flow, Security, Load Balancing Application, Multiple Controller, Policy Based Forwarding.

I. Introduction

The scope of this paper will prove that we can segregate the software from hardware in network devices and the ability for consumers to pick and choose which software they need independent of hardware by migrating to Software Defined Networking which makes the network programmable and innovative. To enhance the network in software defined approach, first step is to replace all the networking devices into SDN platform. So a traditional router is
replaced by white box hardware switch (data plane) and SDN controller (control plane + management plane) with application running over/on it. Finally there is only one SDN controller to handle all the multiple forwarding devices.

The steps involved to enhance the network infrastructure in SDN approach: 1. Running Centralized Routing Application module in a SDN controller such as OSPF, BGP, etc application for multiple devices. This maintains a RIB for the entire topology and reduces the control traffic tremendously.

2. Securing the network devices from all kinds of attacks using SDN security application.

3. Sensing QoS and equally sharing the user traffic among the servers using LB application.

4. Multiple SDN controllers managing a single underlying topology with each having different set of policies based on the needs.

5. Policy Based Forwarding- Define any policy at any time to any node. It is the main aspect of the centralized approach which makes the network more flexible and agile.

II. Goals and Technologies

A. Goals of SDN: Network Operators depends upon Vendors to introduce new features. This causes huge delay from time of request to implementation. Thus there is no provision to deploy their user demands and business requirements immediately. These new trends demand the network with following:

1) Increased Agility and Speed and mobility

2) Replace complexity with operational efficiency

3) Enhancing the user experience with customer loyalty

4) Ensuring security associated to workloads

Software-defined networking (SDN) is a new approach to networking in which control plane is decoupled from hardware and given to a software application called a controller to simplify the management of networks with diverse functionalities and enable flexible deployments.

B. Scope of SDN technologies

SDN technologies can be classified based on the networking device plane they operate on. The scope of the work includes study and understanding of the underlying principles of the various technologies available for SDN. At present the scope will be to implement the technologies being defined by the ONF- Open-flow for the data plane, OVSDB for the management plane and Centralized Routing Control through BGP for control plane. The objective is to be at par with industry standards for SDN. The scope of this paper is to replace the traditional routers with SDN
routers and to enhance the network availability using SDN Security and Load Balancing. The resultant is to implement SDN approach in the ISP network.

III. Methodology:

Centralized Routing Application Module: Routing Application Module installs the default flows in hardware switches before any flows corresponding to route entries are updated. These default flows are required to forward OSPF/BGP, ARP and ICMP packets to controller to take proper action.

- Quagga sits on the top of the application and acts as a soft router which establishes BGP peering with neighbouring devices.

- Application uses the service of controller such as ARP handler and host service to know the list of hosts discovered by controller. This data is then used for identifying the next hop interfaces for FIB entries.

- Routing Application Module forms IBGP session with the Quagga. The EBGP routes learned by Quagga are pushed into RIB of the routing application. Thus Application forms RIB and FIB for each routes. This in turn are pushed into controller's configuration data-store.

- Finally these routes are pushed into controller's configuration data-store which in turn are dumped into hardware switches. So whatever routes learned from Quagga are loaded to controller's database.

A SDN Opendaylight Router can handle maximum number of 2 lakhs BGP routes whereas SDN ONOS Router can handle upto 50,000 BGP routes. The number of routes depends upon the controller and it's host machine resources such as CPU, RAM.

C. SDN Security

SDN Security does the following:

- Protect the Controller: if the SDN Controller goes down (for example, because of a DDoS attack), so goes the network, which means the availability of the SDN Controller needs to be maintained.
- Establish Trust: protecting the communications throughout the network is critical. This means ensuring the SDN Controller, the applications loaded on it, and the devices it manages are all trusted entities that are operating as they should.

- Create a Robust Policy Framework: what’s needed is a system of checks and balances to make sure the SDN Controllers are doing what you actually want them to do.

- Conduct Forensics and Remediation: when an incident happens, you must be able to determine what it was, recover, potentially report on it, and then protect against it in the future.

Defense4All is a security application which runs over the Opendaylight Controller. It offers DDoS attack detection engine and a traffic diversion mechanism based solely on the programmable characteristics of SDN enabled elements.

**Working Of Defense4All:**

Defense4All integrated with Opendaylight detects the attacks based on threshold settings and tells the ODL controller to install the flows to redirect DDOS traffic to AMS. Defense4all communicates with Opendaylight via Rest API. Through this API Defense4All performs two main task:

1) Monitoring behavior of protected traffic

2) Diverting attacked traffic to selected AMS

As from the diagram,

- Initially Host 1 can reach Host 2 as switch 2 is programmed such that Defense4all monitors Host 2 (protected node) and its behavior.

- When there is a DDOS attack on Host 2 from Host 1, defense4all application informs ODL to install the flows in switch 2 to redirect DDOS traffic to AMS (Host 3) and to retrieve the filtered traffic back from AMS (Host 4)

- When DDOS attack stops, additional flows installed by application is removed by controller in switch 2
D. Load Balancing

Like other application, Load Balancing Application runs over the Opendaylight controller. It measures any one of QoS parameter and reactively shares the load across the equally distributed network. Any one of the algorithm can be used for load balancing such as Round-Robin, Weighted Round-Robin, Least Count, Weighted Least-Count, etc. These algorithms will be used as a reference for real time implementation. This paper does not focus on load balancing algorithm but how to approach SDN based load balancing.

There are different type of QoS which can be measured such as

a) Bandwidth Utilization

To calculate available bandwidth, Iperf command is used. Iperf is a simple and very powerful network tool used for measuring TCP and UDP bandwidth performance.

b) Latency

Latency is defined as the time taken for a message to travel from source to the destination in the network. So pre-monitoring technique can be used to find the response time between source and destination. In pre-monitoring technique, whenever client asks for the service to the servers, SDN controller collects the status of each server first and then only it will allow packets to be transmitted that is if the number of users count (load) is more, controller will not add the flow or else it will redirect to the available server (load balancing) according to LB service.

c) Loss

Path loss rate plays a very important role in wired network. Path loss is the calculation of unwanted introduction of energy which interferes with the proper reception and reproduction of the signals during its journey from source to destination. The equation used to calculate rate of loss is as follows

\[
\text{Rate of loss} = \frac{\text{Input packets} - \text{Output Packets}}{\text{Input Packets}} \times 100
\]

Where Input packets – are the packets received at the destination & output packets – are the packet sent out of the host.

Load Balancing Application takes input from any of these parameter and install the flow-based rule such that flow with maximum loss is preferred less
E. Multiple Controller Approach

Unlike traditional networks, each network devices can be sliced or shared among the different vendors. For example, switch 1 can be used for both vendors equally i.e., 50 % of utilization by Vendor A and 50 % of utilization by Vendor B. This is called Network Virtualization. These can be achieved by either of these two concepts:

- With the network abstractions, there is a Virtual Tenant Network concept which enables to configure virtual network across multiple SDN controllers. This provides highly scalable network system. VTN can be created on each SDN controller. If users would like to manage those multiple VTNs with one policy, those VTNs can be integrated to a single VTN. As a use case, this feature is deployed to multi data center environment. Even if those data centers are geographically separated and controlled with different controllers, a single policy virtual network can be realized with VTN. Also, one can easily add a new SDN Controller to an existing VTN or delete a particular SDN Controller from VTN. In addition to this, one can define a VTN which covers both Open Flow network and Overlay network at the same time.

- Also there is a open source application called FlowVisor which acts as a transparent proxy between Open-flow switches and multiple Open-flow Controllers. It creates rich slices of network resources and delegates control of each slice to a different controller. Slices can be defined by any combination of switch ports (layer1), src/dst Ethernet address (layer 2), src/dst IP address (layer 3) and src/dst TCP/UDP port or ICMP code (layer 4). FlowVisor enforces isolation between each slice i.e., one slice cannot control another's traffic as mentioned in below figure:

![Figure 4](image)

F. Policy Based Forwarding

Policy Based Forwarding is the big impact of SDN which makes the network easily manageable, scalable and flexible. At any time, we can define the policies based on the user needs with respective to layer1, layer 2, layer 3 & layer 4 protocols and making the devices to react accordingly as per the rules. Open-flow switches act a complete forwarding element as it looks up into the header of all 4 layers in TCP/IP stack unlike the traditional switches/routers.

Policy based forwarding are of:
a) Source Identification

Packet-based (stateless) forwarding is performed on a packet-by-packet basis without regard to flow or state information. Each packet is assessed individually for treatment. As packets enter the device, classifiers, filters and polices are applied to it. Next, the egress interface for the packet is determined via a route lookup. Once the egress interface for the packet is found, filters are applied and the packet is sent to the egress interface where it is queued and scheduled for transmission.

Packet-based forwarding does not require any information about either previous or subsequent packets that belong to a given connection, and any decision to allow or deny traffic is packet specific. This architecture has the benefit of massive scaling because it forwards packets without keeping track of individual flows or state. Example: Segment Routing (MPLS based forwarding)

b) Flow Identification

Flow-based (stateful) packet processing requires the creation of sessions. A session is created to store the security measures to be applied to the packets of the flow, to cache information about the state of the flow (for example, logging and counting information), to allocate required resources for the flow for features such as Network Address Translation NAT, and to provide a framework for features such as Application Layer Gateways (ALGs) and firewall features.

The packet treatment in flow-based forwarding depends on characteristics that were established for the first packet of the packet stream, which is referred to as a flow. To determine if a flow exists for a packet, the system attempts to match the packet’s information to that of an existing session based on the following match criteria—source address, destination address, source port, destination port, protocol, and unique session token number for a given zone and virtual router.

Example: Block user Traffic

A Firewall is a network security application that is used to control the flow of ingress and egress traffic, usually between a more secure local area network (LAN) and a less secure wide area network (WAN). The system analyses data packets for parameters like L2/L3 headers (i.e. MAC and IP address) or performs deep packet inspection (DPI) for higher layer parameters (like application type and services etc) to filter network traffic. A firewall acts as a barricade between a trusted, secure internal network and another network (example: the Internet) which is supposed to be not very secure or trusted.
Algorithm for Block User Traffic based on policy-based forwarding:

```python
def block_flow(flowname, installflag, ethertype, nw_dst):
    flow = {}
    node = {}
    node['id'] = 00:00:00:00:00:01
    node['type'] = 'OF'
    if (installflag != ''):
        installflag = "true"
    flow['installInHw'] = installflag
    flow['name'] = flowname
    flow['etherType'] = 0x0800
    if (nw_dst != ''):
        flow['nw_dst'] = 10.0.0.2
        actions = "DROP"
        flow['actions'] = [actions]
    return flow
```

Host 1 will not able to ping Host 2. Whatever traffic going to destination 10.0.0.2 will be blocked.

**Conclusion**

SDN enables a programmable network control and offers multiple solutions to a variety of use cases. This paper helps the vendors and operators to understand the underlying Bottom-Up SDN solutions to scale out their vertically integrated stack into horizontal stack to achieve rapid innovation solutions with higher interoperability, thereby reducing both capital expenses (CapEx) and operational expenses (OpEx). It also helps to classify a technology as SDN if it is central network control, enabling Network Virtualization, traffic shaping, programmability in networks, service provisioning and thus allowing network engineers and administrators respond quickly according to the on-
demand business requirements. This paper proves that Software Defined Networking (SDN) have the reduced the complexity of traditional networks with higher agility, scalability and manageability. Thus proves the SDN advantage.

Reference

1. Nick Feamster & Ellen Zegura (Georgia Tech) and Jennifer Rexford (Princeton University) , December 2013. The Road to SDN: An Intellectual History of Programmable Networks.


Corresponding Author:
Sathish Manoharan*,
Email: samirmd007@gmail.com