Implementation of Failure Link Recovery through Spanning Tree Protocol in SDN

[1]Shuruthi.S.S, [2]Kannan.S, [3]Sanjayprasad.J
[1]Student, [2]Student, [3]Student
National engineering college, kovilpatti

Abstract - **The advancement of Computer Networks revolutionized the communication technology and paves a better and efficient way for sharing data. Traditional network device, design and deployment were remarkable, but in this era, it is not sufficient. A software defined networking is an emerging technology that is best suited and seems to be satisfied and further development is on the way to meet up the requirements such as high availability, programmability, and cost efficiency of the increased networking devices, and that is what people also waiting for. Because of its flexibility and availability, SDN is under deployment in a carrier grade network. High availability requires faster recovery of failed links. Hence, several methods were proposed and implemented by research scholars. One of the basic link failure recoveries is providing an alternate path for a source to reach its destination. In this paper a rerouting i.e. Providing a dynamic, alternate path to alleviate link failure is implemented in mininet version 2.2.2 which is installed on an Ubuntu 18.04 Bionic host. SDN usually comprises devices with a separated data plane and control plane and a special device called controller act as a brain. In this experiment, Pox controller is used, and the experimental topology is viewed via miniedit-a python program which acts as a GUI for creating topology and the interrupts less flow of ping is also shown. For the improved mechanisms of failure recovery, this implementation serves as the base.**

keywords - **software defined networks, mininet, recovery of link failure.**

I. INTRODUCTION

Computer networks pave the way to share the data among the users where they are connected, for example, Printer is a resource which is installed on one computer, now the other users can access the same printer if they are connected in a single network. This avoids the need to install printer on individual systems and it also reduces the cost of installing printers in all computers.

Computer networks also provide reliability, scalability. The data in the network go through two planes, namely, data plane and control plane. The data plane will forward the packets and the control plane is the logic that controls the forwarding behavior. In traditional network the control plane is located within a switch or router which is inconvenient for the administrators to dictate the traffic flow, but in software defined networks, the control plane becomes software based that can be accessed through a connected device. This created a revolution where an administrator can control the traffic from a centralized interface resulting in virtualization of the entire network.

II. SDN AND MININET

In SDN, the data plane is decoupled from control plane. Due to the presence of a controller which acts as the brain of SDN that controls, data packets resulting in faster convergence compared to conventional networks. When the manufacturer tries to implement more features over the network devices to make it more intelligent, it becomes increased in size and even single network device has become more expensive. But in SDN, the control plane from the network devices are extracted and turned into a separate device called controller. The controller tells the switches where to send packets. It reduces the configurations in individual network devices**.** Any communication between network devices must go through the controller. the simple controller which is connected with network devices is shown in fig1.

Pox is one such framework which can be used as SDN controller. It is mostly used for researching and teaching about software defined networking and network application programming. It consists of pox components, python programs that implement network functionality in SDN.

Mininet, an emulator, creates a realistic virtual network such as emulated host, emulated links, and emulated switches. Each host has its own Ethernet interfaces (virtual). It also consists of network simulator called Miniedit that enables complex topology testing, rapid prototyping without the need to wire up a physical network. It also paves the way for concurrent developers to work independently on the same topology. Open Flow protocol is associated with SDN for the purpose of determining path across network switches.

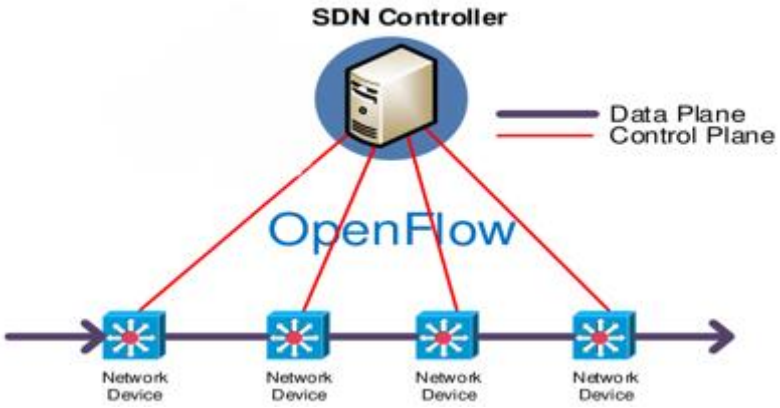

fig 1. SDN Controller and other network devices

III. LINK FAILURE

Link failure causes packet loss, service unavailability, routing instabilities, degradation in quality of service. Detouring the disrupted flows from failed link to an alternate path. Distributed recovery makes delay in finding the failure. The centralized design of SDN reduces the delay in failure detection where the controller immediately modifies the flow entry of each switch. Once a failure is detected, individual routers, updates the forwarding path entries affected by the failure and inform the updated changes to the neighboring nodes.

The alternate path selection can be viewed through the spanning tree protocol. The spanning tree protocol is usually used to prevent the loops in the network. The redundant path is considered to be a backup link through which the loops are eliminated. In this experiment, the redundant path is considered as the alternate path once the link between 2 hosts is down the redundant path is chosen to reach the host. This provides the alternate path for the failed link.

IV. EXPERIMENTAL SETUP

The experiment is carried out in a ubuntu 18.04 host machine where the mininet virtualization tool is installed. To implement the topology, in one terminal run miniedit.py which is shown in fig 2, a python program which consists of toolbar at one side. The toolbar consists of hosts, legacy switch, openflow switches, controller, netlink etc. In another terminal mininet is executed. The topology created in miniedit is shown in the fig 3. The controller has to be configured as remote controller and in the preference submenu of edit make sure to check the start CLI box. This will enable the functions in the mininet console window. The IP addresses of host h1, h2, h3, h4, h5 and h6 are configured as 10.0.0.1,10.0.0.2, 10.0.0.3,10.0.0.4, 10.0.0.5 and 10.0.0.6 respectively.

The functionality of pox controller can be done with python programs simply called as pox components. Open a new terminal and run the pox controller along with its components as shown in fig 4. To view the network events, it is needed to execute the controller in debug mode.

The forwarding.l2_learning component makes openflow switches act like ethernet learning switches that creates flows with idle timeout values. The openflow.discovery component uses link layer discovery protocol messages sent to and received from open flow switches to discover the network topology. The component openflow spanning_tree --no-flood --hold-down prevents the loops and to ensure no packets are flooded in the network before the component creates the spanning tree. The log.level --DEBUG component is used to specify the amount of detail they will see in the log information produced by the pox. The pretty_log provide attractive and readable log output on the pox console. The host_tracker component keeps track of hosts in the network.

The stimulation has been initialized, at the pox controller console and the openflow events being handled by the pox controller is seen, which are displayed as pox log messages. The pox SDN controller connecting to switches and setting up the spanning tree is seen. With the click of Run option in the miniedit, the whole design will come to alive. The output can be viewed in the pox console window.

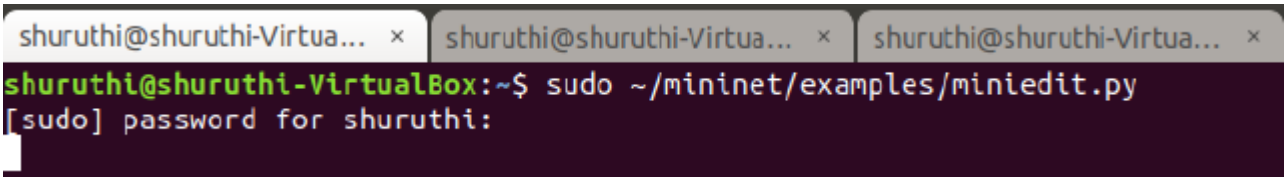

Fig 2. Command to execute miniedit.py

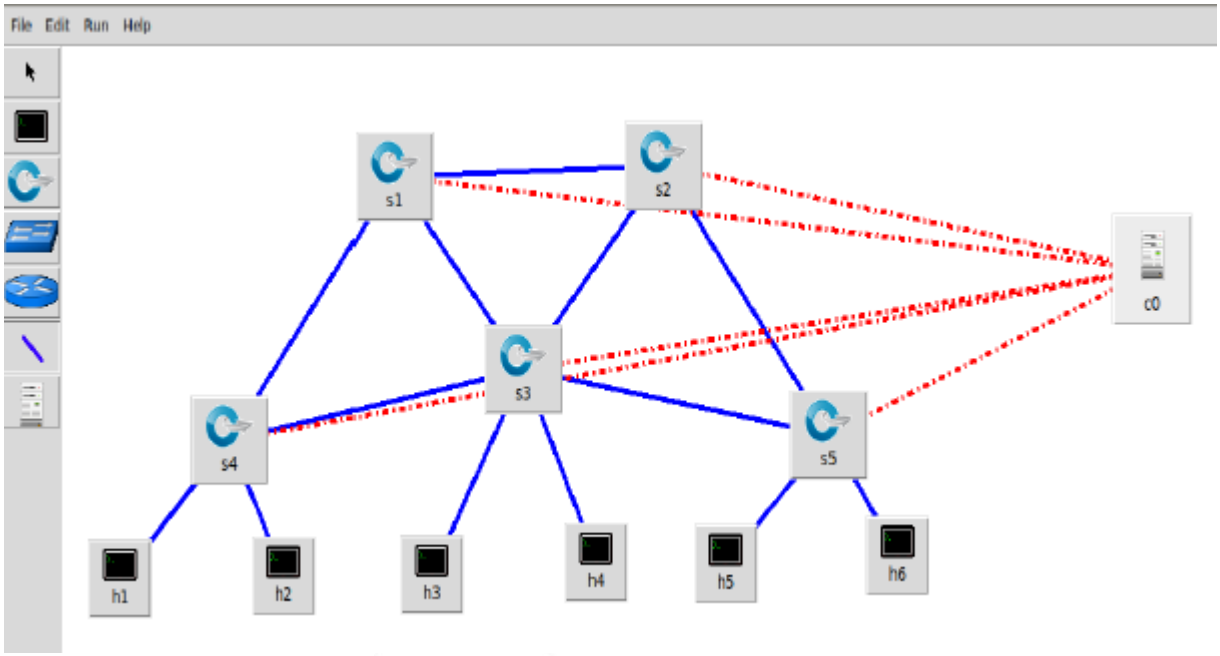

Fig 3. Topology before link failure

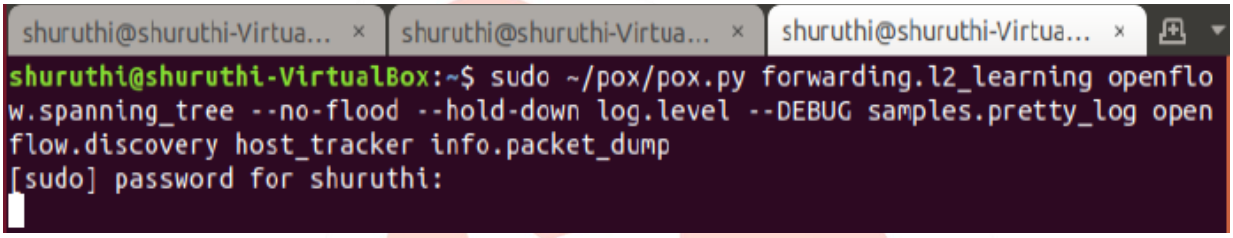

Fig 4. Run pox controller along with its components

V. RESULT

In the host h1, right click and go to the terminal, an x-term terminal pops up. In the terminal ping the host 6 by entering ping 10.0.0.6 as shown in fig 5, which is the corresponding IP address of host h6. The result of ping command is shown in the figure. Now, the links between the switches is made to down by right clicking the link and choosing the link down option (the dotted line in the topology represent link down and the solid line represent link up which is shown in fig 6). It is observed the openflow_discovery packets in the pox console window has shown up but still the ping command in the host terminal is alive.

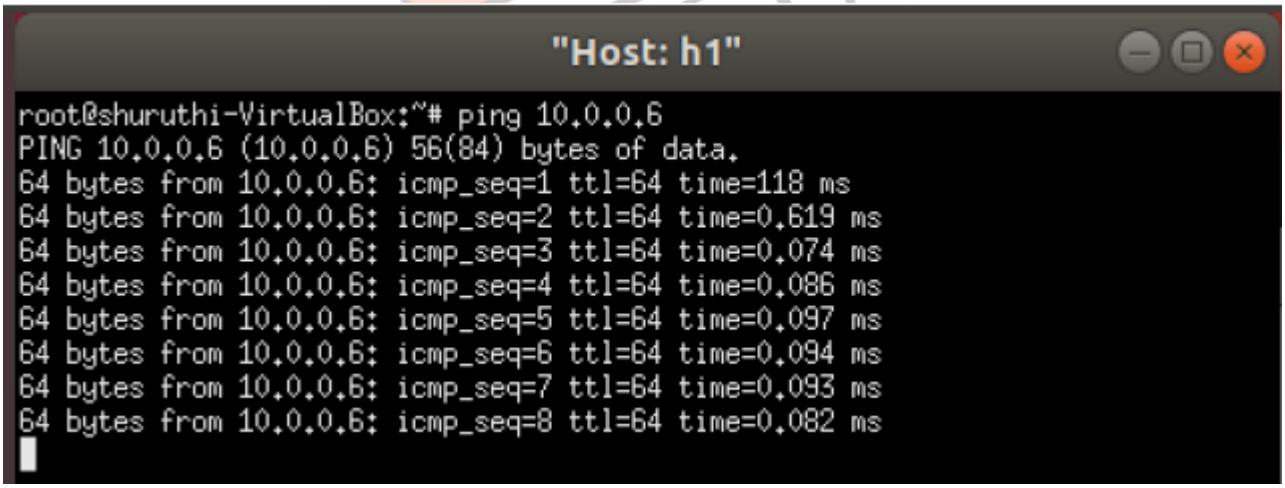

Fig 5. Ping the host h6 from h1

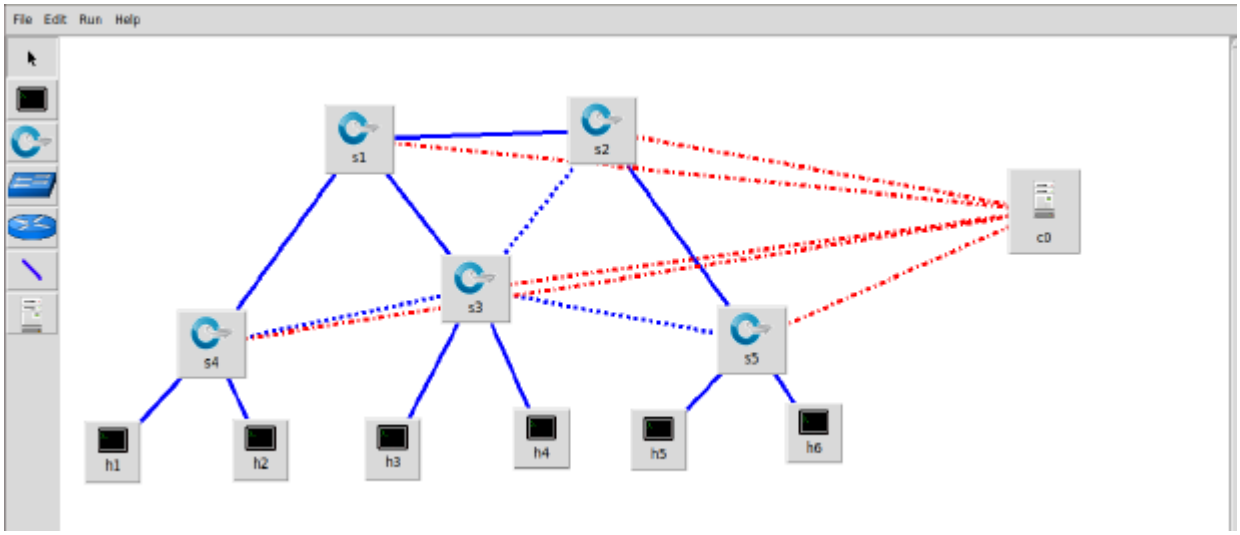

Fig 6. Dotted line represent link down

VI. CONCLUSION

Even after the link is made down, the host is still reachable. This is because the spanning tree component has the redundant path to reach the destination. Hence the host is finding another way to reach the destination through openflow.discovery component and forwarding.l2_learning component whose logs where shown in the terminal as shown in fig 7. For every 10 seconds (by default) the flows in switches are getting updated. Since the path is kept as a backup that act as an alternate path to reach the destination. SDN plays an important role with its flexibility, availability and fast recovery from link failure.

```
[dump:00-00-00-00-00-04 ] [ethernet][ipv4][icmp][echo][56 bytes]
[forwarding.l2_learning ] installing flow for 2e:8f:8b:fa:1d:67.1 -> b2:39:aa:02:b9:89.3
[openflow.discovery     ] link timeout: 00-00-00-00-00-03.3 -> 00-00-00-00-00-04.2
[dump:00-00-00-00-00-04 ] [ethernet][arp]
[forwarding.l2_learning ] installing flow for b2:39:aa:02:b9:89.3 -> 2e:8f:8b:fa:1d:67.1
[dump:00-00-00-00-00-01 ] [ethernet][arp]
```

fig 7. Logs of openflow.discovery and forwarding.l2._learning components

References

[1] Gopi.D., Cheng.S., & Huck,R.. Comparative analysis of SDN and conventional networks using routing protocols., in press.
[2] https://www.comparitech.com/net-admin/software-defined-networking/
[3] https://www.rfwireless-world.com/Terminology/traditional-networking-vs-software-defined-networking.html
[4] Software Defined Networks: A Comprehensive Approach by Paul Goransson, Chuck Black, Timothy Culver - Book
[5] https://en.wikipedia.org/wiki/Carrier_grade.
[6] mininet.org/walkthrough