**LAB 7: IP LOAD BALANCER**

**Objective:** A round-robin load balancer application deployed on RYU controller to perform load-balancing across servers using Mininet.

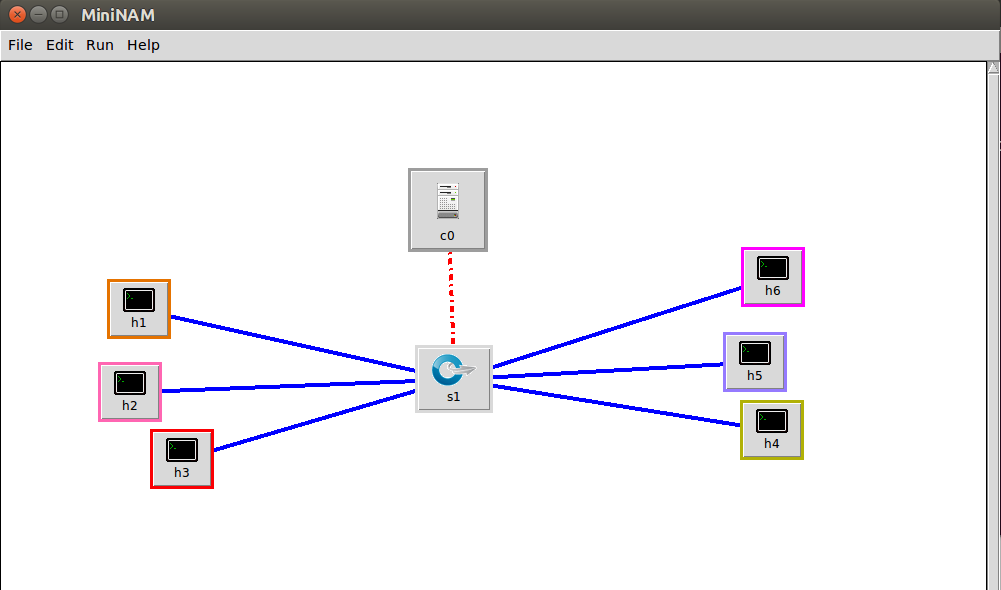
In this lab, the http requests from different clients will be directed to different pre-defined http servers. The server is chosen based on round robin scheduling.

PART 1: Stateless Round robin load balancer.

**Step 1:** Copy the part1.py controller file in the **</usr/local/lib/python2.7/dist-packages/ryu/app/>** directory. Then start the controller <**part1.py**>.

**Step 2:** Start a Mininet topology with 6 hosts, 1 switches and a controller.

***sudo python MiniNAM.py --topo single,6 --mac --controller remote --switch ovsk*** (**Paste the screen-shot of it**)



**Step 3:** Enable the OpenFlow version 13 on the switch.

***sh ovs-vsctl set Bridge s1 protocols=OpenFlow13***

**Step4:** Open the xterm for hosts h1, h2, h3 and start Simple HTTP server on port 80.

**Step 5:** Open xterm for the hosts h4, h5, h6. If you execute <**curl 10.0.0.8>** or <**wget 10.0.0.8>**, (10.0.0.8 being the virtual IP of the load balancer) every new request will be redirected to a different server by the load balancer. Sequence: h4/h5/h6 -> load balancer -> server. The server then gets back to the host h4/h5/h6 via the load balancer only. Sequence: h1/h2/h3 -> load balancer -> h4/h5/h6. Since, on every new request from one of the three hosts (clients), a new server is selected by the load balancer, like h1, then h2, then h3 and so on, this is called as a round-robin load balancer.

**Step 6:** Before executing ‘curl’ or ‘wget’, start Wireshark on the all the servers (h1, h2, h3). Then, execute either ‘curl’ or ‘wget’ from the client hosts i.e. h4, h5, h6 in any sequence. Then check whether all the clients are able to connect to the server. You will notice it doesn’t matter from which host you send HTTP traffic, it will always go to server H1 first and then H2 and so on. **Paste the screen-shot of xterm window of the h4, h5, h6.**

**Step 7:** Check the Wireshark windows you will see the TCP 3-way Handshaking and HTTP get message and **paste the screen-shot** of the same.

PART 2: Stateful load balancer

**Step 1:** Copy the part2.py controller file in the **</usr/local/lib/python2.7/dist-packages/ryu/app/>** directory. Then start the controller <**part2.py**>.

**Step 2:** Start a Mininet topology with 7 hosts, 1 switches and a controller same way as given above.

**Step 3:** Enable the OpenFlow version 13 on the switch.

**Step 4:** Open the xterm for hosts h1, h2, h3 and start Simple HTTP server on port 80.

**Step 5:** Open xterm for the hosts h4, h5, h6 and h7. If you execute <**curl 10.0.0.8>** or <**wget 10.0.0.8>**, (10.0.0.8 being the virtual IP of the load balancer) from any host, any request made by ‘curl’ or ‘wget’ command from h4 or h5 would be redirected by the load balancer to h1 web server, requests from h6 would be redirected to h2 web server and requests from h7 would be redirected to h3 web server. This re-direction to specific servers by the load balancer is done on basis of source IP of the clients h4, h5, h6 and h7 which can be modified in the controller program (part2.py line 194-205).

**Step 6:** Before executing ‘curl’ or ‘wget’, start Wireshark on the all the servers (h1, h2, h3). Then, execute either ‘curl’ or ‘wget’ from the client hosts i.e. h4, h5, h6 and h7 in sequence first in h4 then in h5 and so on. Then check whether all the clients are able to connect to the server. **Paste the screen-shot of xterm window of the h4, h5, h6 and h7**. Answer why h5 was not able to connect to the server.

**Step 7:** Check the Wireshark windows and **paste the screen-shot** of the TCP 3-way Handshaking and HTTP GET message.

This concludes your lab 7 i.e. Load balancer.