Load Balancing Utilization in Data Center Networks

Navjeevan Jot Singh Salana, Adarshbir Kaur
Department of Electrical Engineering, San Jose State University, San Jose, California 95192

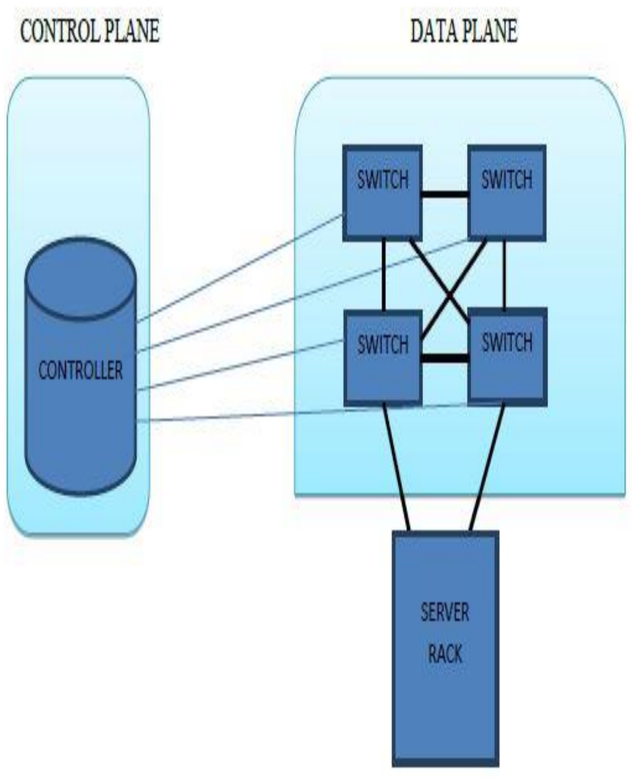
Introduction

The data centers have load balancers to distribute the incoming traffic to various servers present in the Server Racks. Load balancer is a key element of the data center networks. The topology build is done on network simulator 'mininet' and then traffic is passed through topology. There are various topologies used like fat tree, mesh and tree. The POX controller is installed which is Python-based. It is followed by an analysis of all applied load balancing techniques with different queuing times[1] and coming up with a conclusion/analysis of the implemented scenarios. The protocol that is used in this project is the SDN oriented OpenFlow.

Protocol

The project involved the implementation to be done in SDN. SDN is software defined network which divides control plane that decides the route for data traffic and data plane that forwards data traffic to the destination. SDN separates these two platforms i.e. control plane from the data plane and then allow these two to communicate with each other using some methods or protocol

The protocol used in context of this project is the OpenFlow[2]. OpenFlow is the most fundamental and widely used protocol in SDN and allows to set the routing/switching protocols in network.

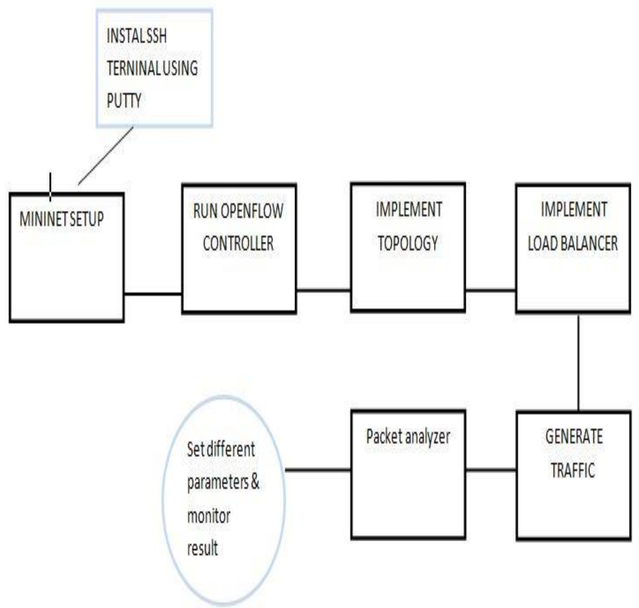


Simulation

Tools used:

- Mininet
- SDN Controller (POX)
- Wireshark
- Topology Script (Python based)
- Traffic Generation Tools

There needs to be a controller running at port 6633, and the topology binds to it remotely[3].



Flow of the project

Invoking the SDN Controller POX:
~$./pox.py log.level –DEBUG misc.RoundRobin forwarding.l2_learning

Implementing Mininet topology
~$sudo mn --custom ~/mininet/custom/fattree.py --topo fattree --mac --arp --switch ovsk --controller remote,ip=<ip address of POX controller>

Implementing Network Capture
~$sudo Wireshark (Capture eth0,eth1 and l0)

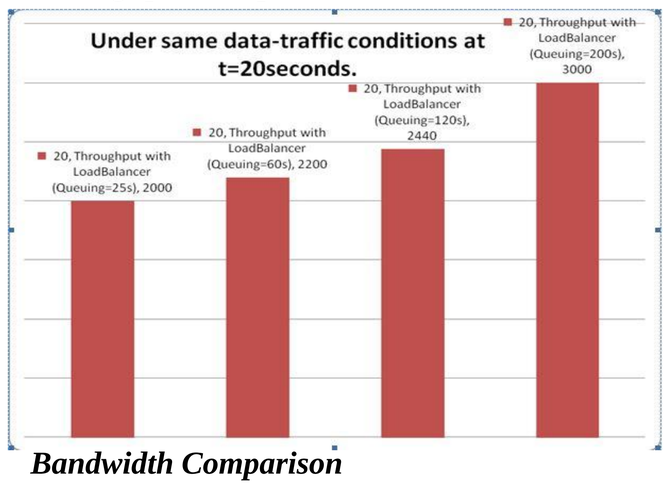
To generate traffic, various tools were used like
i) Iperf
ii) HTTP downloads
iii) ICMP packet transfers
iv) TCP download

There were four different scenarios implemented.
1) Implementation of Load Balancer with 25s queue
2) Implementation of Load Balancer with 60s queue
3) Implementation of Load Balancer with 120s queue
4) Implementation of Load Balancer with 200s queue

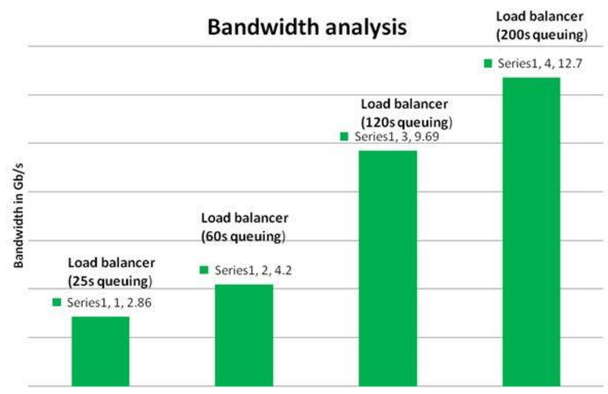
Results

The implementation of different scenarios follows the analysis of which queuing is better

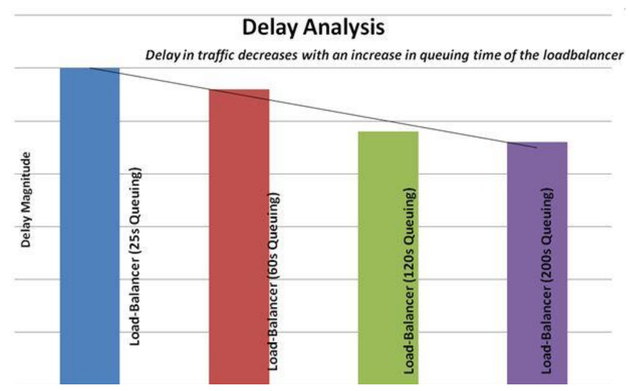
Throughput Comparison



Bandwidth Comparison



Delay Comparison:



Conclusions

Through the implementation of this project, throughput and bandwidth increases with increase in queue value of round robin flow scheduling method and the delay/latency decreases with higher queue time. The change in the throughput level is because with the higher queuing the load balancer gets the higher scheduling time to pass the requests to servers in a round robin load-balancing fashion. As a result, the bandwidth also increases

Key References

[1] "Dynamic Distributed Flow Scheduling with Load Balancing for Data Center Networks" by Sourabh Bharti, K.K.Pattanai

[2] "Open Flow based Load Balancing for Fat-Tree Networks with Multipath Support" by Yu Li and Deng Pan.

[3] "A Round Robin Load Balancing and Redundancy Protocol for Network Routers" by Abddallah, S

Acknowledgments

We would like to thank Prof. Frank Lin for the help and guidance in each step of the project. His expertise and immense support helped us to complete this project successfully.

For further information

For topology code, load balancer script, analysis report, please contact Navjeevan Jot Singh Salana (singh.navjeevanjot@gmail.com) and Adarshbir Kaur (adarshbir.kaur@sjsu.edu)