Server Load Balancing In Software Defined Networking Using A Hybrid Model

Name: Shouvik Chatterjee
Dept: CS
Email: h20200119@pilani.bits-pilani.ac.in
Institute: BITS Pilani

Name: Brijgopal Dixit
Dept: CS
Email: h20200129@pilani.bits-pilani.ac.in
Institute: BITS Pilani

Name: Keval Kulkarni
Dept: CS
Email: h20200136@pilani.bits-pilani.ac.in
Institute: BITS Pilani

***Abstract*—Today's networks have to handle enormous amount of traffic due to an exponential increase in the number of internet users. For exorbitant, inflexible traditional networks, it has become very challenging to manage the network flow. Traditional load balancer are vendor locked and due to its non-programmable nature network administrators does not have capability to build their algorithms by own. But in case of SDN load balancers, it provide the facility to build and design our own load balancing strategy on the centralized controller, thus making the SDN load balancer programmable and flexible. Here we have analyzed the performance of random, round-robin, least active connections (LAC) based strategy and weighted round-robin, and come up with a hybrid algorithm that combines LAC and Server Load Based strategies.**

Keywords— Software Defined Networking, Mininet, Load Balancer, POX, OpenFlow

I. INTRODUCTION

An exponential growth of internet users leads to various network problems such as congestion in the network and overloading of the server. Thus, a load balancer is used to improve the performance of the network by distributing the client requests to multiple servers. Conventional Networks have manual configuration, very inflexible to change, and require high operational expenditure for running the network. Hence new emerging field in the area of information technology is the Software-Defined Networking (SDN) where network control is separated from forwarding plane (data plane) and is programmable directly. OpenFlow protocol used in SDN helps in communication of control plane with data plane.

The layered architecture of SDN is as shown in Fig. 1, description of these layers are given below.

1) Application layer: SDN makes use of APIs to communicate with different applications and programs are installed in this layer. Applications may be business application and network management applications.

2) Control layer: It has a central part of the architecture controller. It takes Network related information from the hardware components and provides global view of the network to the application layer, It also manages network traffic by communicating with the infrastructure layer using Openflow protocol.

3) Infrastructure layer: Hardware components required to transfer packets over the network are present in this layer for example switches, routers etc.

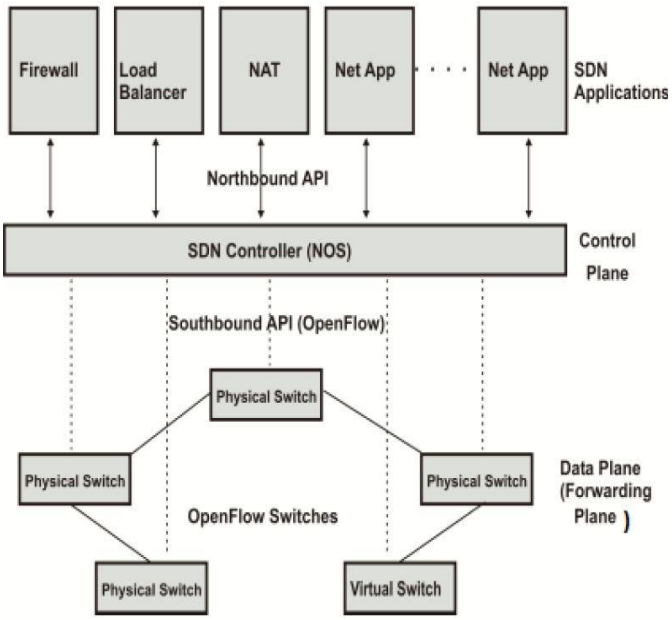


Fig 1: Architecture of Software defined Networking

II. ARCHITECTURE OF THE LOAD BALANCER

A system in which there are 3 servers and a single controller are considered for load balancing and distribution on these servers. There are N clients wanting server CPU time and other resources.

III. STUDY OF EXISTING LOAD BALANCING ALGORITHMS

A. Random Strategy:

As its name suggests, this algorithm selects a server on random basis, i.e. using an underlying random number generator. In cases wherein the load balancer receives a large number of requests, a Random algorithm will be able to distribute the requests evenly to the nodes. So like Round Robin, the Random algorithm is sufficient for clusters consisting of nodes with similar configurations (CPU, RAM, etc.).

B. Round Robin Strategy

Round Robin is the most widely used algorithm. It is easy to implement and easy to understand. In Random strategy, while user requests come to controller it randomly selects one of the available servers. But in round-robin fashion, it tries to distribute load uniformly to all servers. Results show that Round-robin is better than Random strategy in terms of response time, latency and throughput. Let's say we have 2 servers waiting for requests behind the load balancer. Once the first request arrives, the load balancer will forward that request to the 1st server. When the 2nd request arrives, that request will then be forwarded to the 2nd server. Because the 2nd server is the last in this cluster, the next request (i.e., the 3rd) will be forwarded back to the 1st server, the 4th request back to the 2nd server, and so on, in a cyclical fashion.

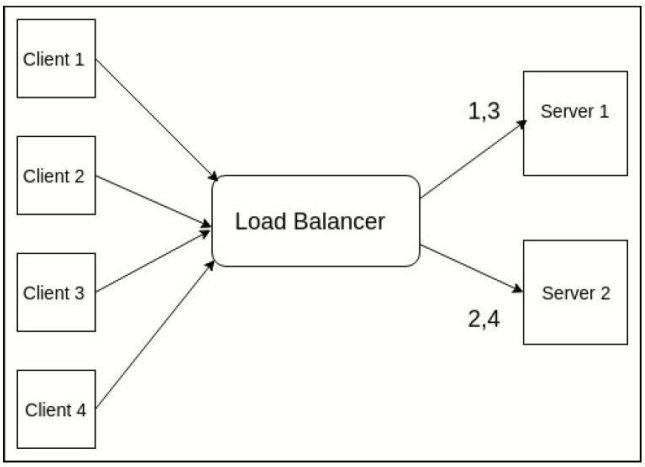


Fig. 3. Round Robin Load Balancer

Even though this method is simple to implement, it has some limitations, For example, if Server 1 has more CPU, RAM, and other specs compared to Server 2, then Server 1 will be able to handle a higher workload than Server 2. Unfortunately, a load balancer running on a round robin algorithm won't be able to treat the two servers accordingly. In spite of the two servers' disproportionate capacities, the load balancer will still distribute requests equally. As a result, Server 2 can get overloaded faster and probably even go down. The Round Robin algorithm is best for clusters consisting of servers with identical specs.

C. Weighted Round Robin Strategy

In this approach we can differentiate between heterogeneous servers. Example, if Server 1 is having higher specs than Server 2, the algorithm will assign more requests to the server with a higher capability of handling load. This algorithm is called as Weighted Round Robin [2]. The Weighted Round Robin is similar to the Round Robin in a sense that the requests are assigned to the servers in cyclical manner only, but with a twist. The server with the higher specs will be assigned a greater number of requests [5]. Here when we set up the load balancer, we assign "weights" to each node. The node with the higher specs will be given the higher weight. Weights are in proportion to actual capacities. For example, if Server 1's capacity is 2x more than Server 2's, then server 1 is assigned a weight of 2 and Server 2 is assigned a weight of 1. So when clients' requests arrive, the first 2 client request will be assigned to server 1 and the 3rd to server 2. If more clients come in, the same sequence will be followed. That is, the 4th, 5th, 7th, 8th will all go to Server1, and the 6th to Server 2, and so on. Capacity isn't the only basis for choosing the Weighted Round Robin (WRR) algorithm. Sometimes, if you want server one to get a substantially lower number of connections than an equally capable server for the reason that the first server is running business critical applications and you don't want it to be easily overloaded.

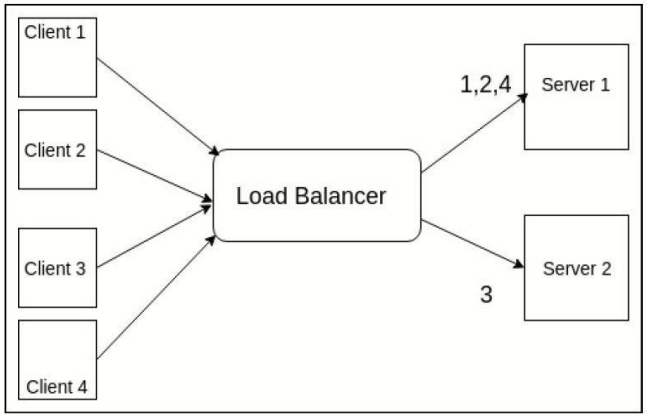


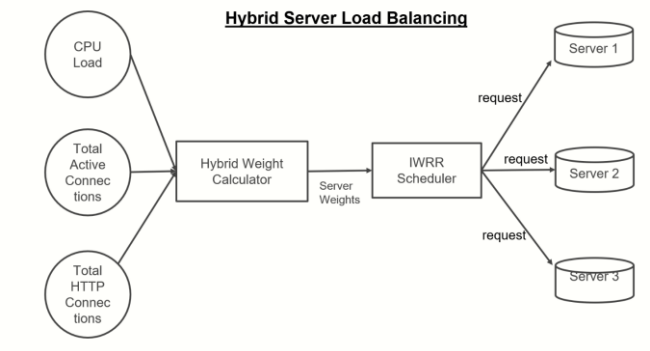
Fig. 4. Weighted Round Robin

D. Least Active Connections(LAC):

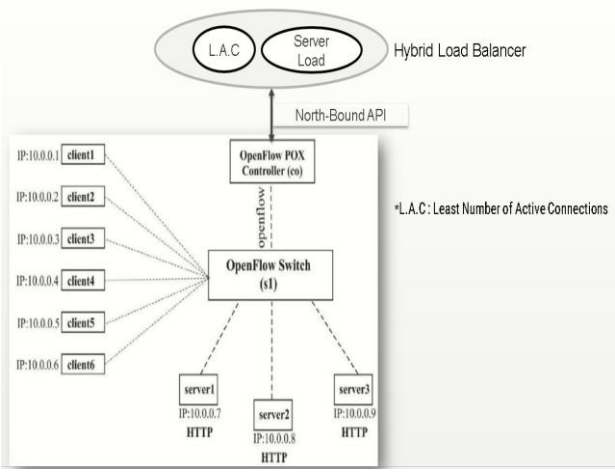
Consider an instance where, even if two servers in a cluster have exactly the same specs, one server can still get overloaded considerably faster than the other. The possible reason for this to happen is because clients connecting to Server 2 stay connected much longer than those connecting to Server 1. This can cause the total current connections in Server 2 to pile up, while those of Server 1 with clients connecting and disconnecting over shorter times, would virtually remain the same. As a result, Server 2's resources can run out faster. In such a situations, the Least Active Connections algorithm is a better fit. This algorithm takes into consideration the number of current connections each server has. When a client attempts to connect, the load balancer will try to determine which server has the least number of connections and then assign the new connection to that server.

IV. OUR PROPOSED HYBRID MODEL

Our proposed model is a hybrid of the Least Active Connection Based and the Server Load Based Strategies. Following Figure illustrates the proposed IWRR (Interleaved Weighted Round Robin) Model based on the hybrid parameters. The Hybrid model calculates the weights of each server based on 3 parameters, i.e. Total Active Connections, Total HTTP Connections and CPU Usage.

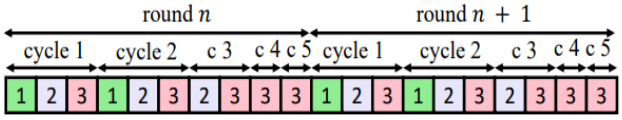


The following figure shows how the Load Balancing Algorithm is attatched to the POX Controller through the NorthBound API. The Clients send requests to the switch(Virtual IP), and the PacketIn Event will be generated to the controller. Then, at the controller, the hybrid algorithm(IWRR) functions) and routes the request to the best available server.



It has an edge over the existing. Interleaved Round Robin has an edge over Normal Round Robin in the following ways:

- WRR, a number of packets equal to the weight allocated to a flow can be served consecutively, which leads to a bursty service.
- the service is bursty because up to w_i packets can be served consecutively for client i, which can cause a large worst-case waiting time for other clients.
- With IWRR, a server i with weight w_i can service w_i requests per round and client can send up to one packet/Connection request at every emission opportunity.
- IWRR spreads out emission opportunities of each client in a round, which is expected to result in a smoother service and lower worst-case delays

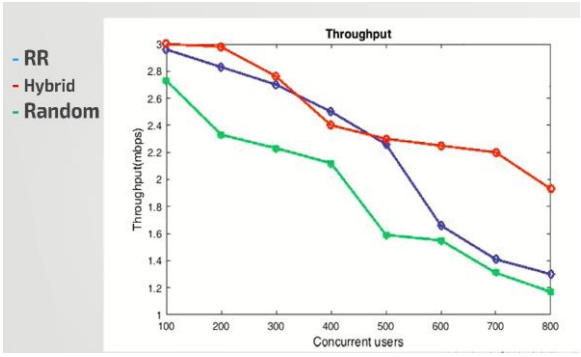


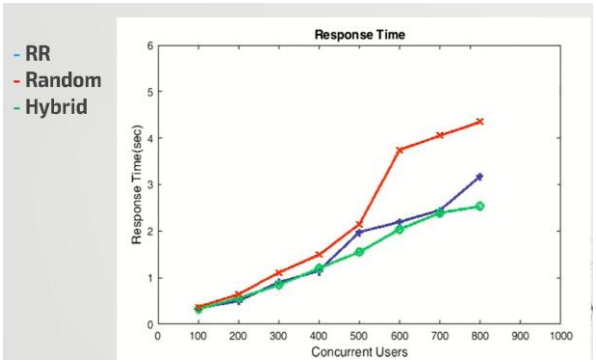
Reverse Proxy

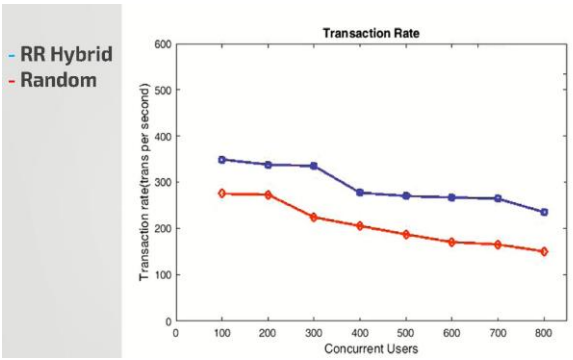
A reverse proxy is a server that sits in front of one or more web servers, intercepting requests from clients. This is different from a forward proxy, where the proxy sits in front of the clients. With a reverse proxy, when clients send requests to the origin server of a website, those requests are intercepted at the network edge by the reverse proxy server. The reverse proxy server will then send requests to and receive responses from the origin server

”.

V. PERFORMANCE ANALYSIS WITH GRAPHS







As we can observe, the performance of the Hybrid Model is the best among all the strategies used.

VI. EXPERIMENTAL SETUP

We have used Mininet for simulating the network. We created the topology as shown in the system model. We used

POX as a controller. The Linux utilities used were Netstat, Mpstat, and AWK. Python libraries used were OS, Subprocess, Time.
The commands used were:

- mpstat -P ALL | awk 'NR==4 { print $13 }‘
  - Gives the %idle time of CPU
  - 100-(above result) gives %CPU Usage

- netstat -an | wc –l
  - Gives the total number of active connections

- netstat -an | grep :80 | wc –l
  - Gives the number of active HTTP connections
  - :80 is to specify port 80 for HTTP

REFERENCES

[1] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, S. Uhlig, “Software Defined Networking : A Compre- hensive Survey,” Proceedings of the IEEE, Vol. 103, No. 1, January 2015.

[2] Wen-Hwa Liao, Ssu-Chi Kuai and Cheng-Hsiu Lu, “Dynamic Load- Balancing Mechanism for Software-Defined Networking,” *International Conference on Networking and Network Applications*, 2016.

[3] Sukhveer Kaur, Japinder Singh, Krishan Kumar and Navtej Singh Ghumman, “Round-Robin Based Load Balancing in Software Defined Networking,” Institute of Electrical and Electronics Engineers IEEE, 2015.

[4] Saket Bhelekar, Mrdvika Iyer, Gargee Mehta and Sheetal Chaudhari, “Dynamic Load Balancing Strategy in Software-Defined Networking,” International Conference on Trends in Electronics and Informatics ICEI, 2017.

[5] Walber Jose Adriano Silva, Kelvin Lopes Dias, Djamel Fawzi Hadj Sadok, “A Performance Evaluation of Software Defined Networking Load Balancers Implementations,”Institute of Electrical and Electronics Engineers IEEE, 2017.