Dynamic Load Balancing using Software Defined Networks

Senthil Ganesh N
M.Tech Final Year
SRM University
Chennai

Ranjani S
Assistant Professor
SRM University
Chennai

ABSTRACT
In the today's modern age the usage of internet has been increasing tremendously. Hence high network traffic which requires many services such as DNS to control. To solve this load balancing are being used. But dedicated load balancers are expensive and quickly become a single point of failure and congestion. But this can be improved for better working. In my approach software defined network using OpenFlow protocol was implemented to improve the efficiency. In the Future Internet, Software-Defined Network (SDN) is seen as one of the most promising paradigm. By using this technique the network becomes directly programmable and agile. Here the http requests from different clients will be directed to different pre-defined http servers based on round robin scheduling. Round robin scheduling is easy to implement and are best to be used in geographically distributed web servers.

Keywords: Load Balancing, SDN, OpenFlow, Round Robin.

1. INTRODUCTION
Load balancing is a significant technology and it can help save power and improve resource utilization in network [1]. The current network infrastructure relies on a vertical architecture based on a variety of software and hardware, thus creating a scenario with little flexibility. Such lack of flexibility has undesirable effects recently coined as Internet ossification. SDN is a new paradigm that breaks the vertical integration between the network control plane and its data plane. SDN users can composite application programs run on top of the controller to monitor and manage the whole network in a centralized and real-time manner [7]. The SDN controllers decides how to manage the packet or which task has to perform such that which packet can be dropped or can be added a new flow entry in order to forward similar packets in the future.

A typical load balancing technique is to use a dedicated load balancer to forward the client requests to different servers, this technique requires dedicated hardware support which is expensive, lacks of flexibility and is easy to become a single point failure [13]. Here we describe the implementation of a dynamic load balancing algorithm to distribute the different traffic flows carried by a network through the different parallel paths between source and destination. OpenFlow is the most common protocol used in SDN networks which are used to communicate the controller with all the Network Elements.

1.1 SOTWARE DEFINED NETWORKS
The traditional network system has the control plane and data plane together. Whereas the SDN approaches to build a computer network which separates and abstracts the network into control and data plane. The data plane does an operation of transferring the packets through the network. Unlike traditional networks, the underlying switches do not implement the control plane. The control plane with its intelligence are able to instruct the data planes over the network.

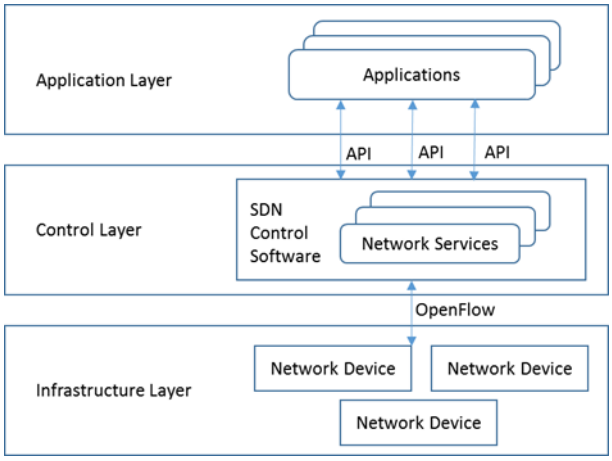

Fig 1: SDN Architecture

The control plane is a software or logical entity, which processes all the routing decisions taken by the data plane. Hence the network becomes directly programmable and agile.

The fig.1 shows the basic architecture of SDN. The separation of the application logic from the forwarding substrate greatly facilitates the deployment and operation of new services and enables SDN to react gracefully to changing network demands and modifications to the substrate topology [5]. OpenFlow is the most common protocol used in SDN networks which are used to communicate the controller with all the Network Elements.

1.2 Traditional Routing Vs SDN Routing
An OpenFlow network is formed by one or more OpenFlow-switches which contain a Flow-table with the de-fined flow entries and the actions to be performed. The main difference between the design of the traditional distributed network and an OpenFlow network lies in the structure of the control plane network. For traditional distributed architectures like the Border Gateway Protocol (BGP), Open Short Path First (OSPF) and the Intermediate System to Intermediate System (IS-IS), the control topology is peer to peer. It means that each forwarding device listens for events from its peers and makes independent decisions based on the local view [12].

1.3 Load Balancing
Load balancing can be achieved not only by specialized software or protocol, but also by installing specialized gateway or load balancers [1]. The first load balancing technology is DNS. DNS load balancing is a simple and

effective method but is it is not possible to distinguish between other servers. There once a server breaks it need more refresh time for it to work again normally. Round-robin works by responding to DNS requests not only with a single IP address, but a list of IP addresses of several servers that host identical services. The IP addresses are returned from the list in the basis of round robin fashion. Geographically distributed web servers are best served by applying DNS load balancing round robin server content distribution.

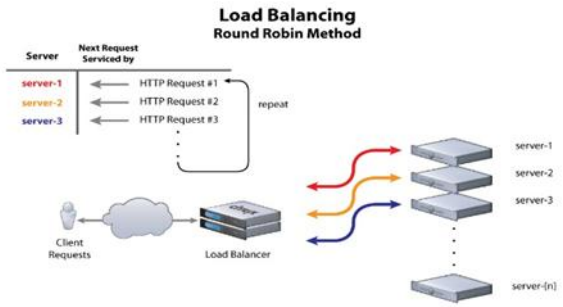

Fig 2: Load Balancing

In dynamic Load Balancing the weights are based on continuous monitoring of the servers and are therefore continually changing. This method distribute connections based on various aspects of real-time server performance analysis, such as the current number of connections per node or the fastest node response time.

2. TECHNICAL BACKGROUND

2.1 SDN Controllers

Five topmost open source controllers in terms of their usage have been analyzed here: POX, Ryu, Trema, FloodLight, and OpenDaylight [10]. To understand and realize the SDN concept we must choose a suitable controller.

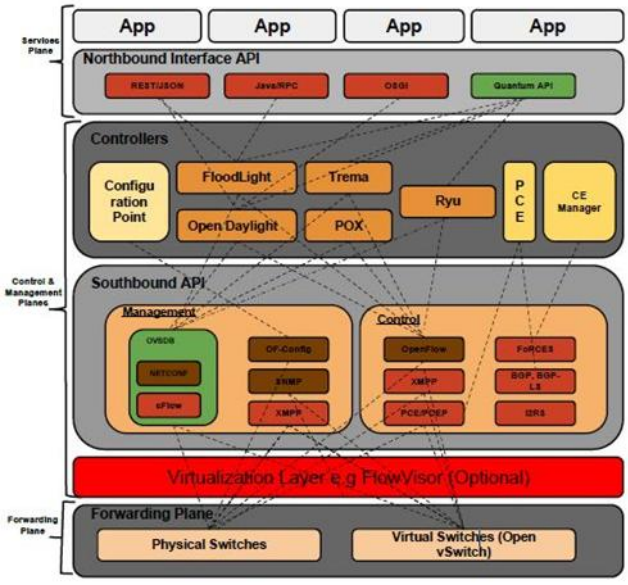

Fig 3: Interfaces of SDN Controllers

Here we have chosen POX controller for our application. It has an efficient performance for the rapid development and prototyping of network control software. POX has a Pythonic OpenFlow interface. This has been used for lot of researchers in universities as it is very easy for programming. It supports OpenFlow v1.0. It can run anywhere and can be bundled with install-free PyPy runtime for easy deployment. It's used to explore prototype distribution, SDN debugging, network virtualization, controller design, and programming models.

2.2 OpenFlow

OpenFlow provides a mechanism for SDN. While OpenFlow was first proposed as a way to enable researchers to conduct experiments in campus networks, its advantages lead to its use beyond that. OpenFlow's central-control model can avoid the need to construct global policies from switch-by-switch configurations, and can also support near-optimal traffic management [6]. The behavior of a forwarding device can be summarized in two steps: (1) When it receives a packet that does not match a certain entry in its routing table, it contacts the controller that defines how the packet should be forwarded or discard the packet; (2) when the received packet matches a rule in its routing table, the corresponding action in the forwarding table is performed [11].

2.3 Mininet

Mininet creates a realistic virtual network, running real kernel, switch and application code, on a single machine (VM, cloud or native), in seconds, with a single command. Mininet is also a great way to develop, share, and experiment with OpenFlow and Software-Defined Networking systems. The figure 4 shows the creation of the custom topology.

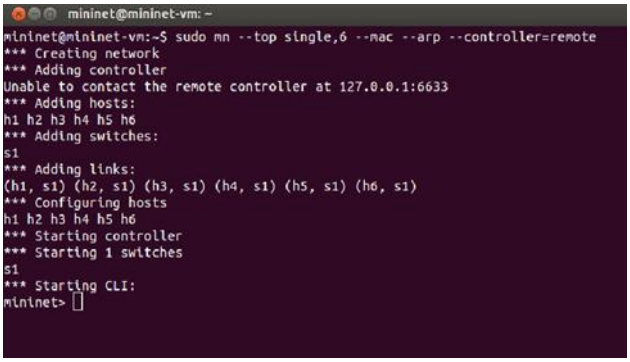

Fig 4: Mininet Topology Creation

3. IMPLEMENTATION AND EVALUATION

```
1: // N is the size of circular ACK re-sequencing buffer
2: wSeqNum = (cumulative ACK sequence number) mod N
3: if wSeqNum > left then
4:    left = wSeqNum        // in-sequence, new ACK
5:    put ACK at the end of the re-sequencing buffer
6:    release held ACK(s) up to left point with a peak rate
7: end if
8: if wSeqNum == left then
9:    switch RPC state      // duplicate ACK
10:       case NORMAL        // re-sequencing
11:          hold ACK with an identified number, break
12:       case LOSS          // no re-sequencing
13:          release ACK to a sender, break
14:   end switch
15: end if
```

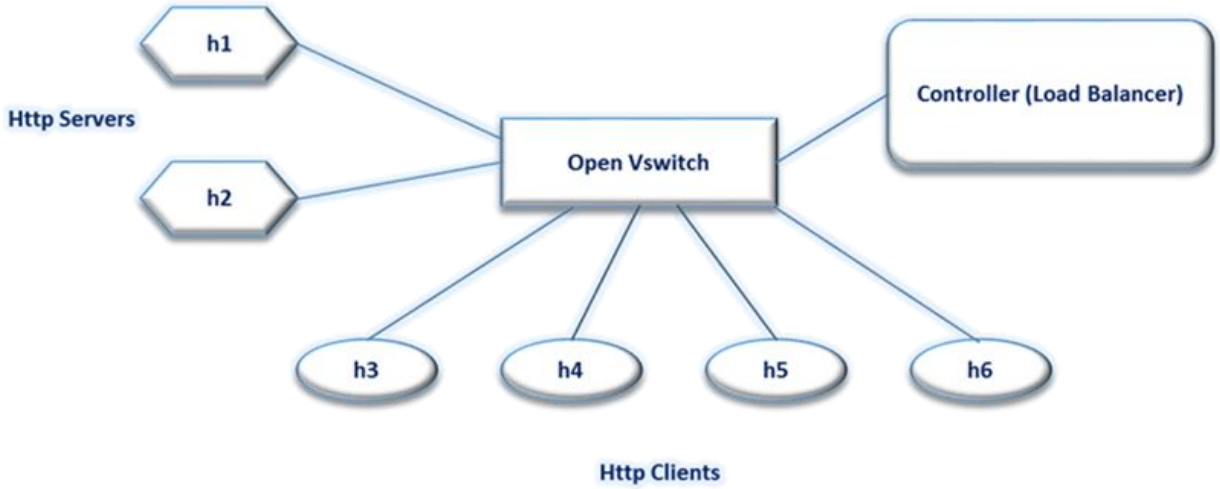

Fig 5: Custom Topology

Figure.5 shows the network topology which we have taken for evaluation. POX controller is used for implementing the load balancing. Mininet is used for creating a virtual network topology. Here we have considered a topology with 6 host nodes. Where two nodes are taken as http server other 4 as http clients. Depending upon the arrival of the traffic to the server from the client nodes the server are scheduled. Clients access the service through a single public IP address, reachable via a gateway switch.

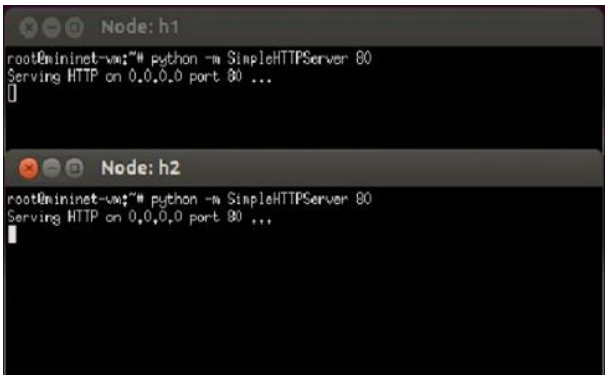

Fig 6: Server Set up

The load-balancer switch rewrites the destination IP address of each incoming client packet to the address of the assigned replica. The round robin policy uses a circular queue to decide where to send a request. The load-based policy sends a request to the server with the lowest load, where load is defined as the number of pending requests.

After creating a custom topology using Mininet. Then the servers has to be set up. In node 1 and 2 the HTTPServer is made up with the port number of 80. The servers should set up with individual IP address. This has been shown in the figure 6. IPerf is a popular network tool that was developed for measuring TCP and UDP bandwidth performance. The user is able to perform a number of tests that provide an insight on the network's bandwidth availability, data loss, delay and jitter.

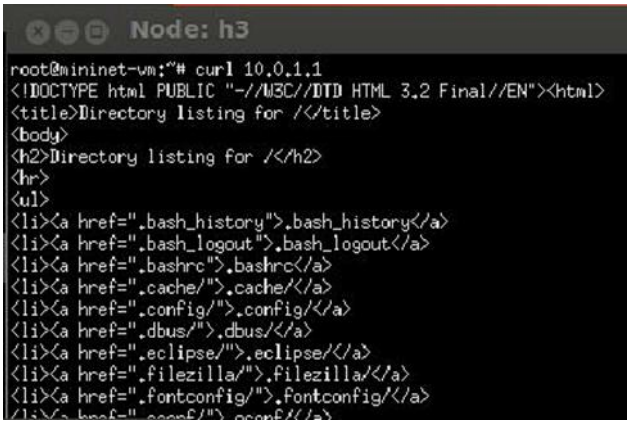

Fig 7: Traffic from Client Node

Next, by using the curl command the traffic is sent to the server. The curl command is used from all the four HTTPClient nodes. The figure 7 show the clients which sends the traffic to the servers. This gets the webpage from the IP of the server. Thus by using the round robin algorithm the client gets its server in a circular manner. Figure 8 shows directing of traffic from different nodes to the servers. All the 4 client nodes are given request to the server. The server starts to direct the traffic alternatively.

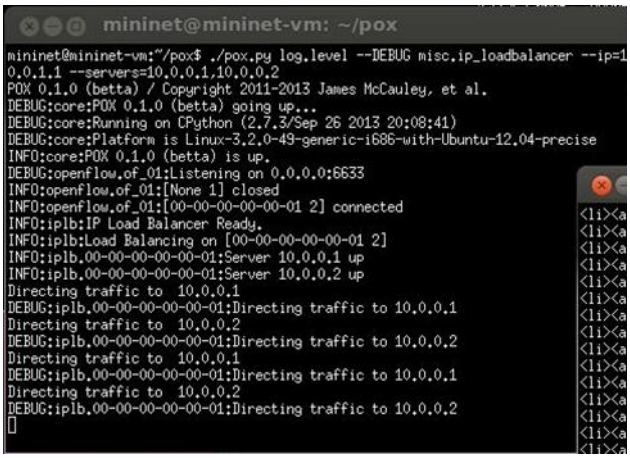

Fig 8: Directing Traffic

4. CONCLUSION AND FUTURE WORK

Irrespective of the developments in the IT industry Software Defined Networks is an evolving computing paradigm has influenced every other entity in all the private sector. Round robin DNS load balancing works best for services with a large number of uniformly distributed connections to servers of equivalent capacity. The project can further extended work by investigating more related load-balancing algorithms and multicast algorithms for SDN. Also in future load balancing can be evaluated by considering more parameters.

5. REFERENCES

[1] Yuanhao Zhou, Li Ruan, Limin Xiao, Rui Liu. 2014 A Method for Load Balancing based on Software Defined Network. School of Computer Science and Engineering Beihang University, Beijing, China.

[2] Richard Wang, Dana Butnariu, and Jennifer Rexford. OpenFlow-Based Server Load Balancing Gone Wild. Princeton University; Princeton, NJ

[3] Martí Boada Navarro, 2014. Dynamic Load Balancing in Software Defined Networks. Aalborg University.

[4] Zdravko Bozakov and Amr Rizk, 2013 Taming SDN Controllers in Heterogeneous Hardware Environments. Leibniz Universit¨at Hannover, Germany.

[5] Yannan Hu, Wendong Wang, Xiangyang Gong, Xirong Que, Shiduan Cheng, 2012. Balanceflow: Controller Load Balancing For Openflow Networks. Beijing University of Posts and Telecommunications, Beijing, 100876, China.

[6] Jehn-Ruey Jiang, Widhi Yahya and Mahardeka Tri Ananta. 2011. Load Balancing and Multicasting Using the Extended Dijkstra's Algorithm in Software Defined Networking. National Central University Jhongli City, Taiwan.

[7] Junjie Zhang; Kang Xi; Min Luo; Chao, H.J. Load balancing for multiple traffic matrices using SDN hybrid routing.

[8] Rahamatullah Khondoker, Adel Zaalouk, Ronald Marx, Kpatcha Bayarou. Feature-based Comparison and Selection of Software Defined Networking (SDN) Controllers. Fraunhofer Institute for Secure Information Technology Rheinstr. 75, Darmstadt, Germany.

[9] Felipe Alencar, Marcelo Santos, Matheus Santana, Stenio Fernandes, How Software Aging Affects SDN: A View on the Controllers. Universidade Federal de Pernambuco (UFPE) Recife, Brazil.

[10] G. Araniti+, J. Cosmas_, A. Iera+, A. Molinaro+, R. Morabito+, A. Orsino+. OpenFlow over Wireless Networks: Performance Analysis. University Mediterranea of Reggio Calabria, Italy.

[11] Zhihao Shang, Wenbo Chen, Qiang Ma, Bin WU. Design and implementation of server cluster dynamic load balancing based on OpenFlow. Lanzhou University Communication Network Center Lanzhou,China.