IJCTA, 9(2) 2016, pp. 1259-1265
© International Science Press

Dynamic Scheduling for Traffic Management and Load Balancing using SDN

R. Subramanian*[*] and T. Manoranjithm[**]

ABSTRACT

Software Defined Networking (SDN) is an upcoming trend in computer networks which allows the entire network to be dynamically configured by a higher level programmable plane. The principle of SDN is to segregate the control plane from the data plane and have it available as a separate software module called SDN controller. This SDN controller is a programmable entity which allows to control the data flow through a higher level program. In traditional networks as the entire-view of the network is difficult to obtain, to address this problem, we propose a scheduling solution based on Software Defined Networking (SDN). In this paradigm, the network traffic is only handled by theOpen Flow switches, where the flow-handling decisions are determined by the central SDN controller. Based on loads of the server the paths can be dynamically adjusted with the global view i.e. load at the moment in the whole network. The proposed scheme can effectively balance the traffic and the overall network performance is also improved as well.

I. INTRODUCTION

The primary concept of SDN is to separate the control plane which is the decision making component of the network, from the data plane which handles the actual data movement in the network. This separation of data plane from control plane making it possible to control or monitor a network from a centralized controller view. In traditional networksrouter and switches have individual decision making capabilities provided by vendor specific software running on them. Since each vendor follows different software specifications functionalities ofall these devices differ from each other and can't be changed uniformly across the network as different vendor components requires different changes making the reconfiguration process very complex.In SDN, data planes are cheap commodity forwarding devices without any decision making capabilities. SDN allows to have a centralized control plane where all the decisions are taken for the date plane. This centralized control planeuses theOpen Flow protocol to communicating with the data plane. Network applications are implemented on top of the control plane by using a combination of flow table entries. Controller takes the flow based routing decisions and each controller decision are made as a new rule entry or overwrite an existing rule in the flow table.

Flow rules stores in the open flow table in the Open flow switches. Controller independently sets up every flow in per flow based routing. If an incoming flow matches an already existing flow entry then the same flow rule is followed for that incoming flow as well.

II. RELATED WORKS

In computer networks Software-deûned network (SDN) has become one of the most important architectures for the management of larger complex networks, which require repolicing or reconûgurations from time to time. SDN achieves easy repolicing by decoupling the control plane from data plane. Thus, the network

[*] (M.Tech), SRM University

[**] (Assistant Professor S.G), SRM University

routers/switches become simple forwarding devices pushing packets by following the ûow table rules given by the control plane [1].

POX is a Python based open source Open Flow/Software Defined Networking (SDN) Controller. POX controller is used mainly for academic and research works as it allows faster development and prototyping of new network applications. POX controller comes pre-installed with the mininet virtual machine. Using POX controller you can turn dumb Open Flow devices into hub, switch, load balancer, firewall devices. The POX controller allows easy way to run Open Flow/SDN experiments. POX can be passed different parameters according to real or experimental topologies, thus allowing you to run experiments on real hardware, test beds or in mininet emulator [2].

The idea of programmable networks has recently gained huge momentum due to the emergence of the Software-Deûned Networking (SDN). SDN is often referred to as a "radical new idea in networking", promises to dramatically simplify network management and enable innovation through network programmability [3].

SDN is an emerging paradigm in networks which have immense potential to replace the existing network systems with an intelligent and more robust mechanism for routing, scheduling and load balancing. In this paper various existing legacy system scheduling strategies are implemented in the SDN based environment for identifying an optimal strategy.

III. SDN COMPONENTS

(A) Controller

The control plane is also called controller. This controller has a global view of the network and controls the flow through the network. The switch contacts the controller with the packet and the controller make a decision based on the scheduling strategy implemented. Switch will act as a simple forwarding device based on the flow table entry.

Since most intelligence is transferred to the controller and the switch only performs the actions specified buy the controller. This makes the switches very simple and inexpensive. As shown in fig.1 switch and controller communicates with the switch using Open Flow protocol. Fig. 1 shows the interface between the controller and the switch using Open Flow protocol.

POX is an open source controller for developing SDN applications. POX controller provides an easy way to implement the Open Flow protocol to establish communication between the controllers and the switches in the SDN network. POX controller can be used run different applications like hub, learning switch, load balancer or firewalls. Each ûow-entry includes at least a simple action specifying the forwarding condition for that flow.

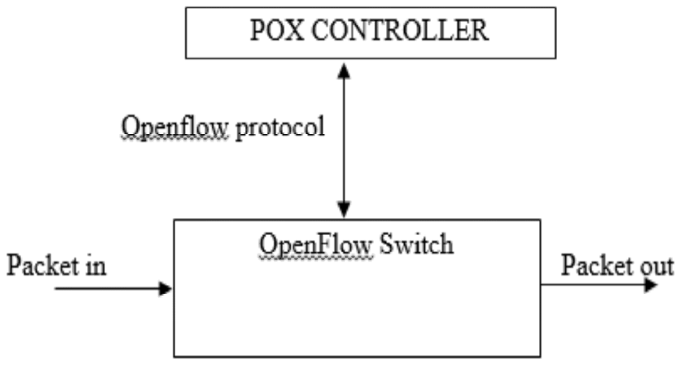

Figure 1: Controller switch Interface

(B) Open Flow Switch

An Open Flow switch can be a software program or a hardware device that forwards packets in the software-defined networking (SDN) environment. Open Flow switches are based on Open Flow protocol or compatible with the Open Flow protocol. Open Flow switch maintains an open flow table which contains all the rules given by the controller for forwarding the packets. As shown in fig. 2 Open Flow switch contains the Open Flow table.

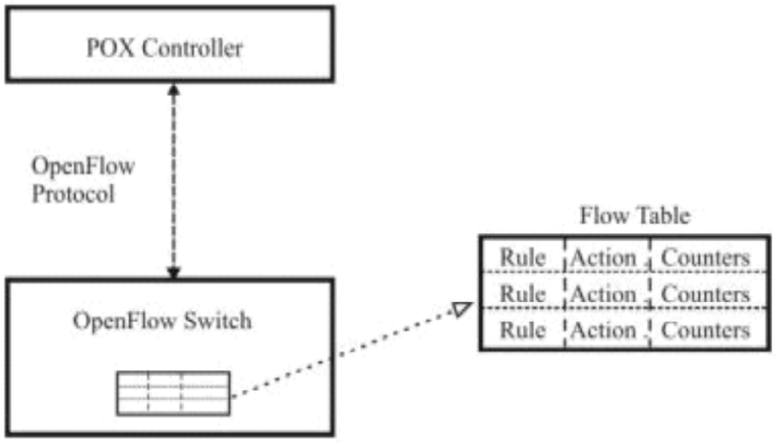

Figure 2: Open Flow Switch

(C) Open Flow Table

Open Flow table has forwarding rules given by the controller. Each rule in controller has a priority field which helps to resolve conflicts between 2 or more rules in the Open Flow table. Each rule in the table will expiry time after some time automatically or controller explicitly orders to remove a particular rule. Rules can be based on the source address or destination address or port number or protocol based. Fig. 3 shows the various fields in the Open Flow table in which Classifier field contains the IP address and port information to classify the incoming packets and Action field contains the forwarding rule specified by the controller for each specific classifier.

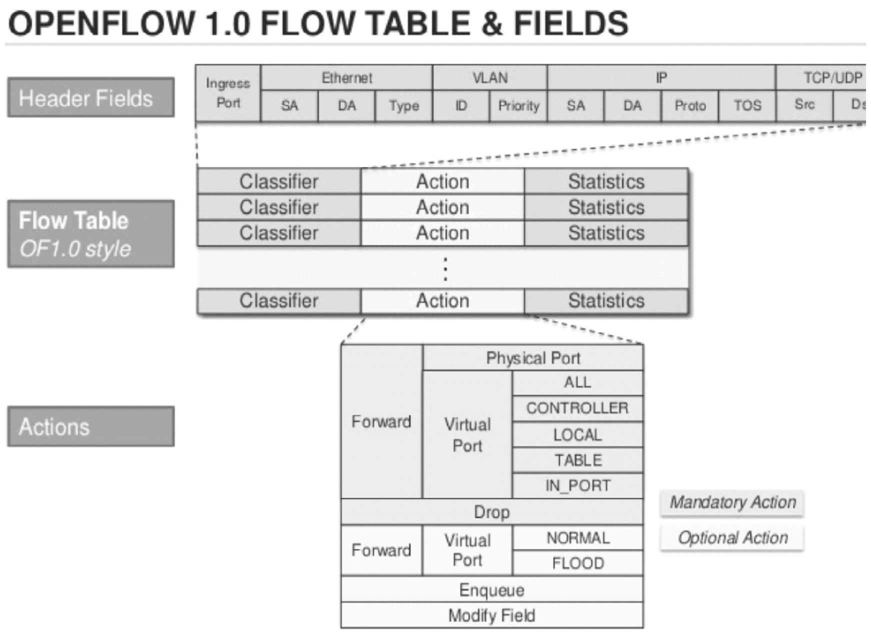

Figure 3: Open Flow Table

(D) Mininet

Mininet is an emulation tool which can be used to virtual network topologies with switches, controller, hosts and links. It uses virtualization concept to make a single system act as a virtual network running a specified topology. It is a simple, robust tool to develop and test Open Flow based applications. Using network does not require any physical configuration which is helpful in creating a complex topology without worrying about the configuration overhead. Mininet is designed for Linux environment it uses command line interface in the terminal window.

IV. TOPOLOGY

As shown in Fig. 4 the topology consists of a POX controller, a single Open Flow virtual switch (ovsk), 4 servers and several clients. All the servers and client have static IP's configured to them. Severs will be SimpleHTTPsevers running on the default port 80. The switch will the Open Flow table. When a client send a request the destination IP address is checked with the Open Flow table , if its new destination the switch forwards the request to the controller which then reply's with the destination port to forward the request. The switch will forwards the request to the port specified by the controller and updates the flow table. When next time a request with the same destination address comes the switch directly forwards the request based on the flow table entry.

Each flow table entries will have a timeout period after which the entry is removed from the flow table. The controller takes all the decisions based on the programmable control logic and pushes them as rules to the flow table, switches simply follow these rules from the flow table.

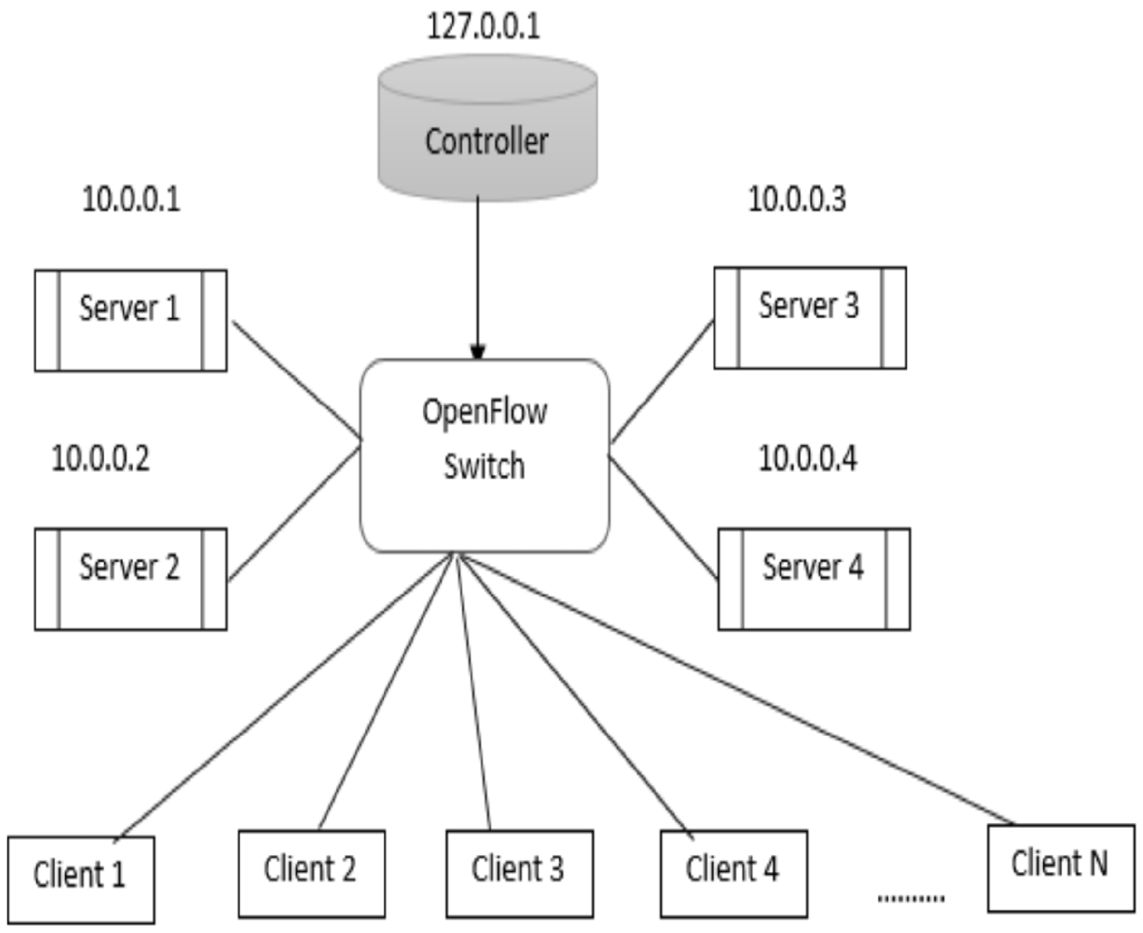

Figure 4: Topology

V. SCHEDULING ALGORITHM

In computing, scheduling distributes workloads across multiple computing resources, such as processors, a computer cluster, network links or servers. Load balancing aims to optimize resource use, maximize throughput, avoiding overload of any single resource and to get minimum response time. In this project load balancing in a SDN network is done using different scheduling algorithms to identify an optimal scheduling strategy for SDN environment.

(A) Random Scheduling

In random scheduling the controller selects a server from among the list of servers available based on a random selection process without any other consideration of that server state or load.

(B) Round Robin (RR)

In Round Robin generally time slices are assigned to each server in equal portions in a circular order, handling all servers without any priority. Roundrobin scheduling is simple, easy to implement a starvation free load scheduling. Roundrobin scheduling can also be applied to other scheduling problems, such as data packet scheduling in computer networks.

(C) Weighted Round Robin (WRR)

Weighted Round robin is similar to round robin but in addition to a fixed time-slice for each server a multiple of this time-slice is pre-defined and a same sever is picked subsequent number of times as the number of times the multiple to the time-slice is defined.

VI. IMPLEMENTATION

The topology as shown in Fig. 4 is created in mininet. An Open Flow switch connects all the clients and the servers together. Controller will be running separately as a remote controller in the network. Only the switch can communicates with the controller directly.

All the servers and client will have a unique IP. The four servers run in the IP's 10.0.0.1 to 10.0.0.4 respectively. Each of the servers runs a service reachable at 10.0.1.10. The service has a fixed size load as response. Whenever a client requests to that service IP one of the servers pick up the request and respond with HTTP 200 OK message in header and the load.

Scheduling algorithms are implemented with the pox controller which supports python programing. Fixed size pockets are used as load uniformly across all the server. When a client send a request to reach this service IP the switch receives this request and forwards that to the controller. The controller schedules that request to one of the servers based on the scheduling strategy specified at that time. Once the controller schedules the request to a particular server then the switch forwards that request to that specific server. The server which receives the request will respond with a fixed size load.

One successful completion of the request and response pair is considered as a single transaction. To compare the effective strategy Transactions per second, Average response time and Load per second are measured. The scheduling strategies implemented are Random Scheduling, Round Robin and Weighted Round Robin.

Fig. 5 show the response time comparison between the three strategies Weighted Round robin (WRR) has the minimum response time of the three strategies implemented.

Fig. 6 show the throughput comparison in which also the weighted round robin (WRR) has the overall better throughput. This study clear shows that in a fixed load environment Weighted Round Robin (WRR) based scheduling strategy is ideal for network performance in the SDN environment.

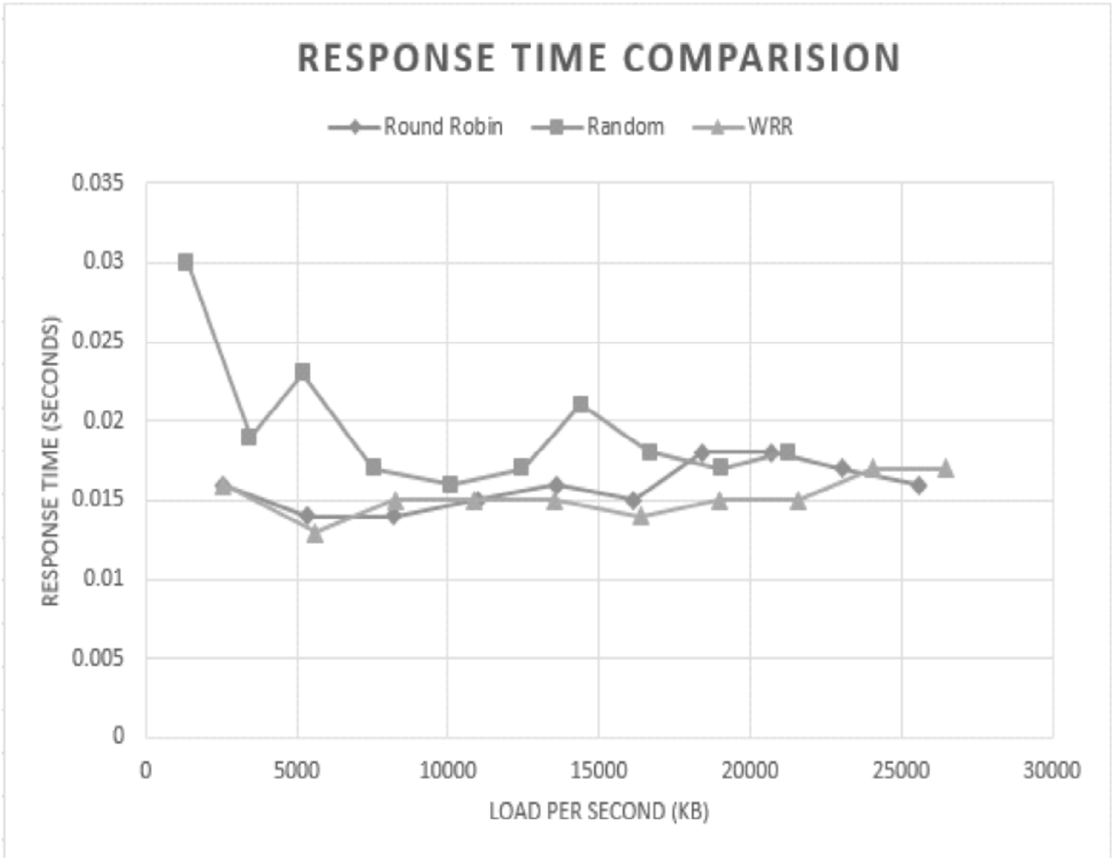

Figure 5

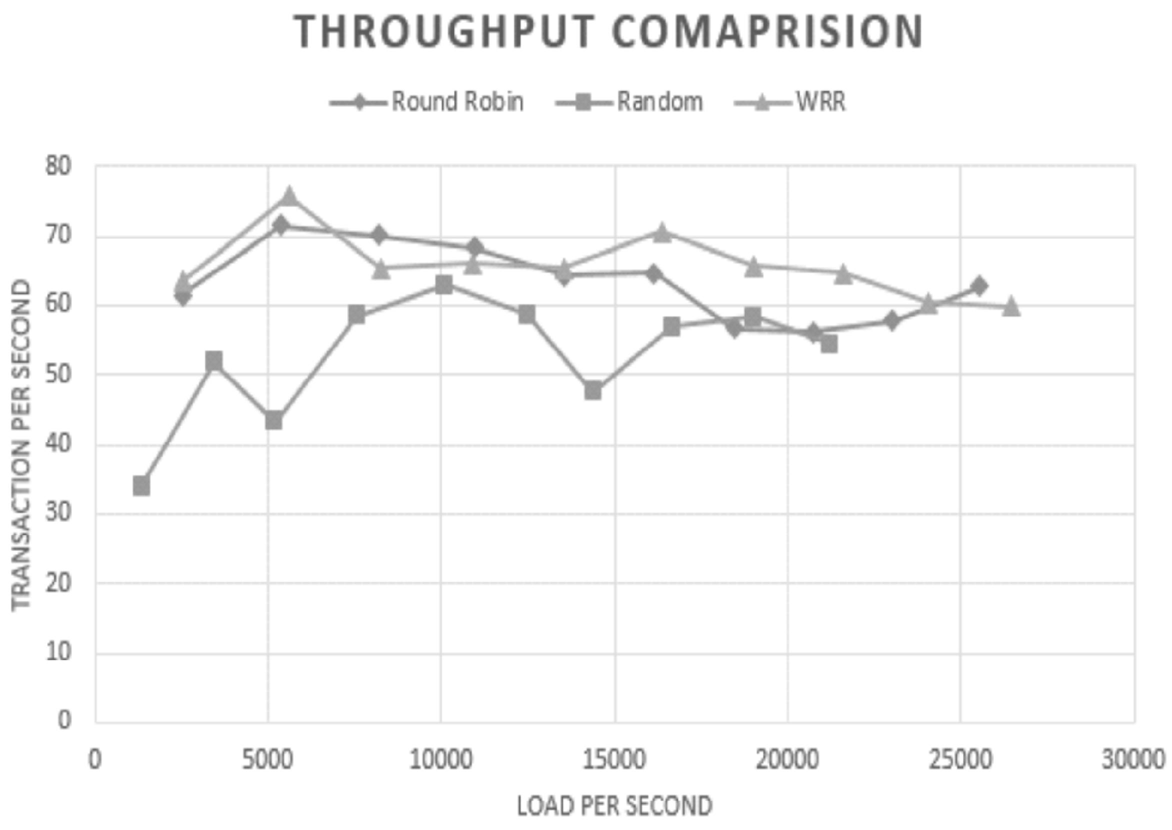

Figure 6

VII. FUTURE WORK

To further utilize the SDN paradigm of programmable switches it is design an intelligent controller to access the network performance continuously and change the scheduling strategies automatically based on the network performance. The idea to identify the performance of different scheduling strategies under different load scenarios and formulate a rule table which states the ideal strategies for the changing network performance based on the load. The controller utilizes this rule table to changes the scheduling strategy according to maintain optimal performance under different loads sceneries.

REFERENCES

[1] Fei Hu, Qi Hao, and Ke Bao "A Survey on Software-Deûned Network and Open Flow: From Concept to Implementation" in IEEE Communication, fourth quarter, VOL. 16, pp. 4, 2014.

[2] Japinder Singh, Navtej Singh Ghumman, Sukhveer Kaur "Network Programmability Using POX Controller", in ICCS, pp. 134-138, 2014.

[3] Katia Obraczka, Marc Mendonca, Nunes, B., Thierry Turletti, X. Nguyen. "A survey of software defined networking: Past, present, and future of programmable networks." in IEEE Communication, third quarter, VOL. 16, pp. 3, 2014.

[4] Hiroaki Hata, "A Study of Requirements for SDN Switch Platform", in IEEE, pp.79-84, 2013.

[5] Ellen Zegura, Feamster, Nick, Jennifer Rexford, "The road to SDN: an intellectual history of programmable networks." ACM SIGCOMM Computer Communication Review 44, no. 2 pp 87-98 2014.

[6] Lara, Adrian, Anisha Kolasani, and Byrav Ramamurthy. "Network innovation using openflow: A survey." IEEE pp.1-20 2013.

[7] CHEN Shuqiao, LAN Julong, WANG Peng "Open Flow Based Flow Slice Load Balancing", in IEEE,pp.72-82 2014.

[8] Koerner, Marc, and Odej Kao, "Multiple service loadbalancing with Open Flow." In High Performance Switching and Routing (HPSR)", IEEE 13th International Conference, pp. 210-214, 2012.

[9] Limin Xiao, Li Ruan, Mingfa Zhu,Mingming Zhu, Yuanhao Zhou "A Load Balancing Strategy for SDN Controller based on Distributed Decision", IEEE 12th International Conference, pp.851-856, 2014.

[10] Li-Chun Wang, Yu-Jia Chen, Yi-Hsin Shen "Trafûc-aware Load Balancing for M2M Networks Using SDN", IEEE 6th International Conference, pp. 668-671, 2014.