VOL. 12, NO. 16, AUGUST 2017 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

SDN ENABLED PACKET BASED LOAD-BALANCING (PLB) TECHNIQUE IN DATA CENTER NETWORKS

J. Saisagar, Prashant Kothari D., Ruturaj U. Kadikar and V. Deeban Chakravarthy
Department of Computer Science Engineering, SRM University, India
E-Mail: saisagar8@gmail.com

ABSTRACT

The traffic in data centre networks has been increasing constantly in the past few years. It is almost impossible for one server to handle all the requests coming from the client because of this huge traffic. Hence the solution is to balance the load by transferring the traffic to the underutilized core switches. Traditional load balancers use very expensive and inflexible hardware. Since these load balancers are locked in by the vendors, only few fixed algorithms can be used which neither can be modified in the future nor innovative algorithms be created by the network administrators. An alternative of these hardware based load balancers is to use SDN Load balancers. These SDN load balancers do not require costly hardware and can be programmed, which it makes it easier to implement user-defined algorithms and load balancing strategies. In this paper, we have implemented packet based load balancing technique using OpenFlow vs switches connected to ONOS controller.

Keywords: software defined networking, OpenFlow, mininet, load balancing, ONOS.

INTRODUCTION

The transmission of data has been increasing exponentially in data center networks. It leads to increase in traffic among the switches. This causes the congestion problem which results in lower throughput and long delay [1]. Traditional networks have the control plane and data plane merged together which reduces flexibility and stops the growth of networking infrastructure. Under Software defined networking the data plane and control are made separate which makes it easier to control the network and future network evolutions. The routers and switches basically form the data plane. The control plane consists of the control logic of the network and is handled through a controller as shown in Figure-1[2].

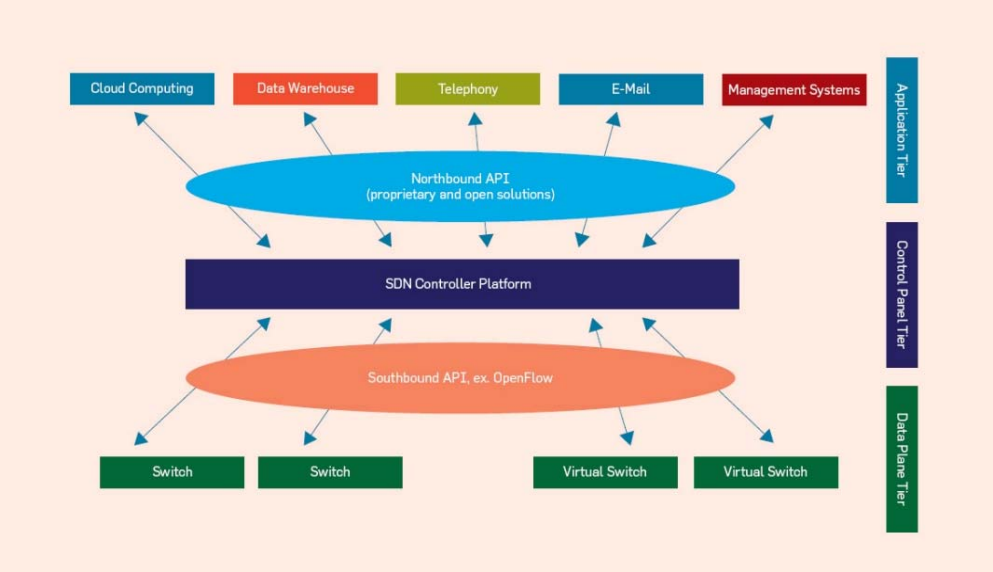

Figure-1. SDN Architecture [Courtesy: www.cacm.acm].

The SDN northbound Application Programming Interface forms the bridge between the SDN controller and the applications which run over the network. It is the most critical part of the SDN network because it forms the base for all the innovative applications which are created. On the other hand the SDN southbound API facilitates interaction between the SDN controller and the underlying switches or routers of the network. OpenFlow is the industry standard SDN southbound API which allows the SDN controller to make real time changes dynamically. OpenFlow computes the Flow-table of switches/routers by adding/deleting the flow entries [3].

Load balancing is a technique that re-routes the traffic to balance the load of servers. Hence it forms a very critical part in every network to run quickly and efficiently. Traditional load balancers use very expensive

hardware which could not be changed according to the need of the network. SDN load balancers on the other hand are not hardware based load balancers and they can be programmed according to the need of the network. SDN load balancer can be integrated in an SDN network in the form of applications.

In this paper, we have developed an application of Open Network Operating System (ONOS) to control the loads by redirecting traffic at the ports of switches. For this we used packet switching technique to allow the traffic if it satisfies the given threshold otherwise the packets will be redirected to another core switch with less traffic.

The upcoming sections of this paper are as follows. Section II highlights the related work in load balancing. Section III describes the Packet-based Load balancing module. Section IV displays the results of the experimentation. Section V consists of conclusion and future work.

RELATED WORK

Load balancing as the name suggests balances the load of a network device and makes the process faster and efficient. It balances the excess load of the node by computing and routing the traffic to a node with lesser load. From past two years there have been various methods incorporating Load balancing in Data Centre Networks based on different algorithms and architectures.

In 2015 Sukhveer *et al*. [4] discusses about Load balancing in SDN using the Round-Robin Algorithm for routing the traffic to other nodes. The results obtained from the Round-Robin Algorithm were compared to the already existing Random based load balancer. The results were compared on the basis of total transactions per second and average response time. Both of these parameters were better for the Round-Robin Strategy than the Random Distribution Strategy. In the same year, Widhi *et al*. [5] proposed Load balancing based on the Extended Dijkstra's algorithm for SDN. The Extended Dijkstra's algorithm calculates and finds the shortest path using edge weights and the node weights. This algorithm also takes account of the link loads so as to avoid congestion. When comparing with Round-robin algorithm, dynamic load-balanced routing in OpenFlow-enabled networks (LABERIO) and Randomized algorithm it produced the best results among them for the following parameters: End-to-End Latency, Throughput and Standard Deviation Load of servers.

In 2016, Gang *et al*. [6] proposed Dynamic sub-topology Load Balancing (DCLB) algorithm for Data centre networks under Software defined Networking approach. The DCLB algorithm requires very little storage space to store the link information in the controller. It also chooses the path where the link cost is the lowest. The DCLB algorithm produces better results than the Dynamic Load Balancing (DLB) algorithm. In the above mentioned papers the controller used is POX controller. In this paper the controller in action is ONOS.

Packet based load-balancing

Tools for implementation

ONOS Stands for Open Network Operating System. It is a controller which provides control plane for SDN including switches and links with communication services to end hosts. ONOS implements various types of functionality, including APIs, resource allocation, permissions, as well as user-facing software such as a CLI and a GUI for system applications as shown in Figure-2. The controller manages the entire network rather than a single device [7].ONOS can run as a distributed system across multiple servers, to use the CPU and memory resources of multiple servers which enables fault tolerance in case of server failure and upgrades of hardware and software without interrupting network traffic flow. The kernel, core services and applications of ONOS are written in Java as bundles that are loaded into the KarafOSGi container. OSGi helps to install and run Java modules dynamically in a single Java Virtual Machine (JVM). ONOS can run on various underlying OS platforms as it run in JVM [7].

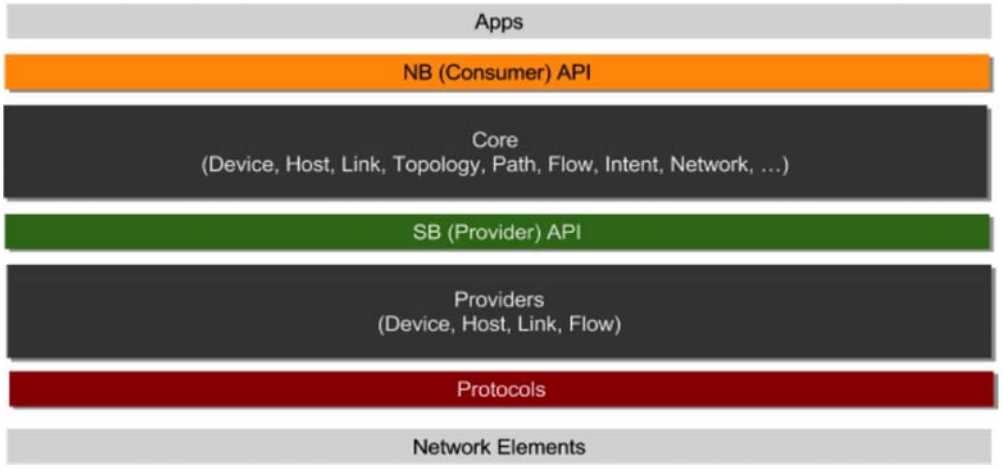

Figure-2. ONOS stack [Courtesy: www.wiki.onosproject.org].

In this work, ONOS is used as the controller to balance the loads and carry out the various events with the help of the ONOS application. The application is useful to build the network for the real time project. In order to create a virtual network with switches to balance the loads we have used Mininet. Mininet is an emulator for SDN which helps to create virtual hosts, switches and links which runs on linux operating systems. It helps to create complex topologies using the Python API extension. In this work we have used Fat-Tree topology to test the load balancing between the switches. The switches enabled in the Mininet are Open Flow switch which provides open source distributed virtual multilayer switching environment [8].

Topology

In Data Center Networks the most commonly used network topology is the Fat-tree topology. Fat-tree topology is used in this work since the branches nearer the top of the hierarchy are thicker than branches further down the hierarchy. Since the branches are data links the varied thickness of the data links allows for more efficient and technology-specific use.

The Topology contains Core Switches and Edge Switches to balance the traffic flowing through the network as shown in Figure-3. The controller gathers the information such as links, flows rules, ports statistics and device id from the topology that is created on Mininet. From the information that is gathered, controller follows the Dijkstra's algorithm to find the shortest path. In case when congestion occurs, the controller balances the load through Intents [9, 10].

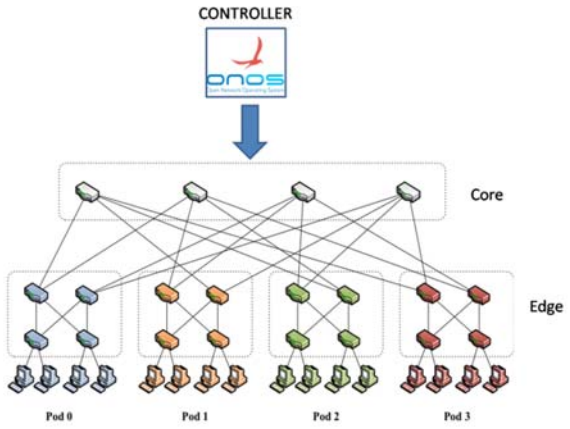

Figure-3. Fat-tree topology.

Packet-based load balancing algorithm

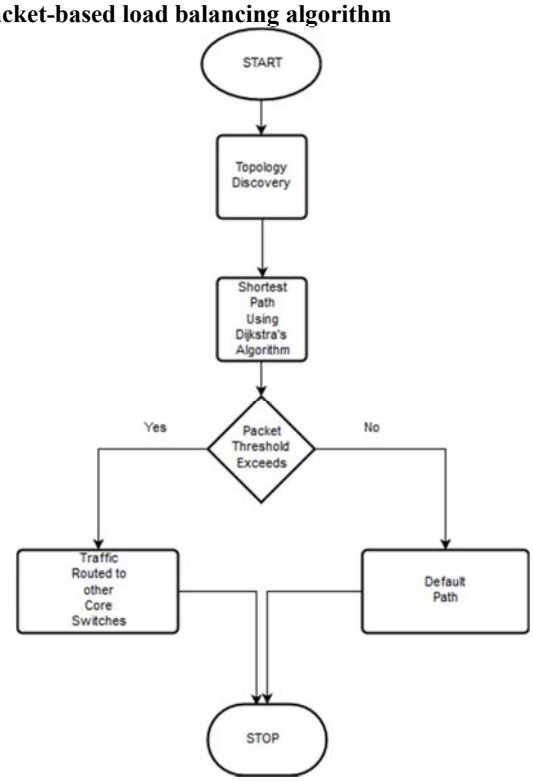

Figure-4. Packet-based load balancing algorithm.

Figure-4 shows the network topology that is created with Mininet Emulator and which provides network flows, switches and hosts. ONOS uses Link Layer Discovery Protocol (LLDP) to discover the network topology created in Mininet. OpenFlow switches created in Mininet forms the data plane of the network [11]. ONOS uses Dijkstra's Algorithm to find the shortest path between a source and a destination. An ONOS Application is created to redirect the traffic flowing through the OpenFlow switches. Threshold value is assigned based on the processing capacity, port capacity and average traffic. When the number of packets received at ports of the core switches is less than the threshold value then the packets are transferred through the default path that is computed by the Dijkstra's Algorithm as shown in the Figure-5(a). If the number of packets received is higher than the threshold value then the packets are transferred to the other core switches in order to avoid congestion as shown in Figure-5(b). The forwarding of the traffic contains a flow rule which allows ONOS to redirect its control of links. In this work we redirect the traffic from one port to another port of the edge switch, so that the traffic finally goes from one core switch to the other core switch.

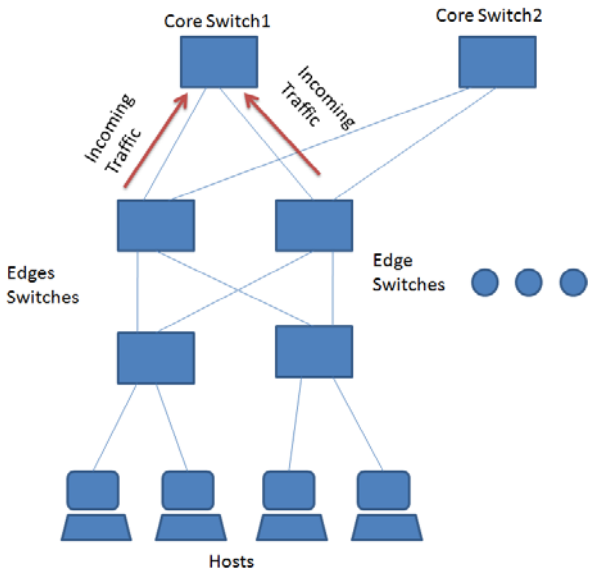

Figure-5(a). Incoming traffic flow.

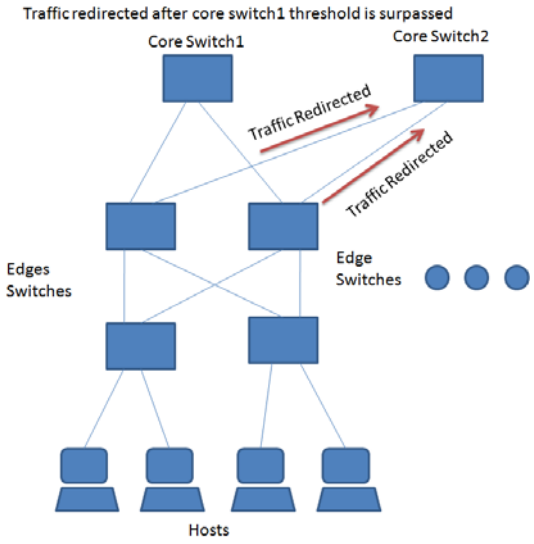

Figure-5(b). Redirecting traffic flow.

RESULTS

The focus is kept on the core switches because; it is the main element or the neck of the network. The saturation of the core switches will result in max degradation of traffic. In this work, we have adopted a packet-based load balancing technique where the traffic is routed to a different core switch if the packet threshold is exceeded.

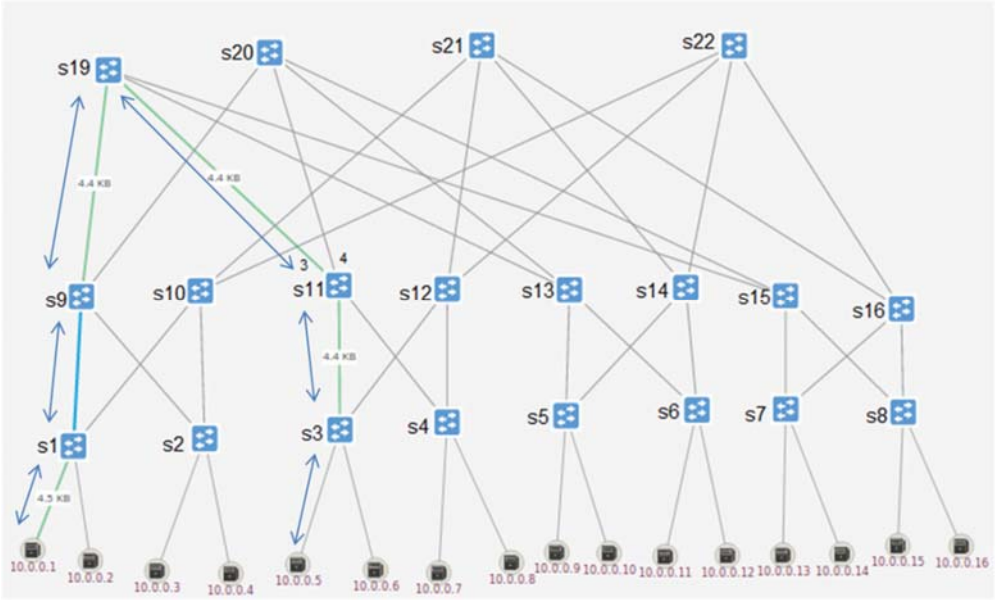

Figure-6. Traffic flow under threshold.

The traffic flow follows a default path created by the Dijkstra's Algorithm of the controller as shown in Figure-6. In this flow the data is transferred from the host 1 to host 5 (10.0.0.1 to 10.0.0.5). The switches used for this path are is s1, s9, s19, s11 and s3. We have taken the results of the number of packets being transferred from the Switch s11 which contains 4 ports. Each port directs the traffic to a different switch. Also the Direction of the flow is Bidirectional since the packets sends an acknowledgement back to the source.

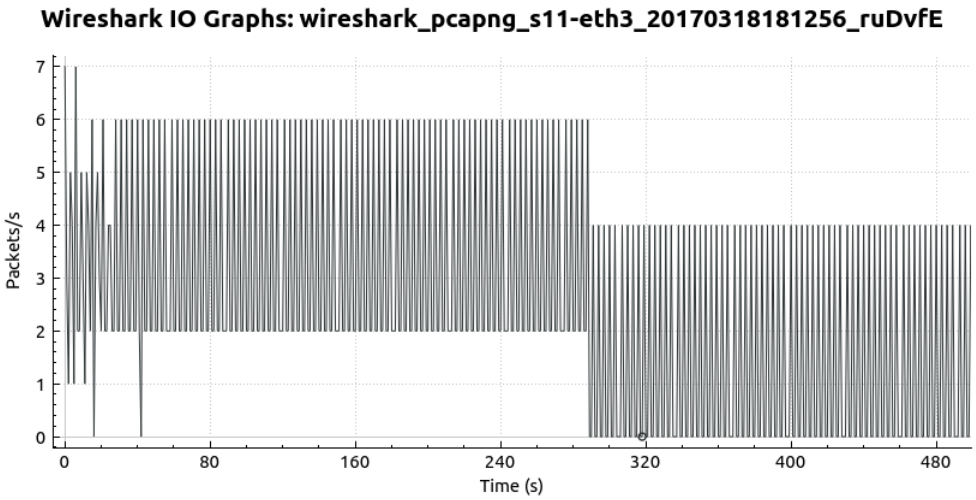

Figure-7. Packets transferred from switch 11 Port 3.

Figure-7 shows the number of packets in the y-axis and time interval in the x-axis. It helps in estimating the number of packets being transferred from switch s11 port 3. For our work we have used Internet Control Message Protocol (ICMP) packets for the transfer. The ICMP packets are transferred until the threshold is satisfied. The threshold limit was given as 300 packets. Until the time interval of 260 seconds the threshold is satisfied and packets are transferred from port 3. Once the threshold is exceeded the packets are redirected to an alternate path.

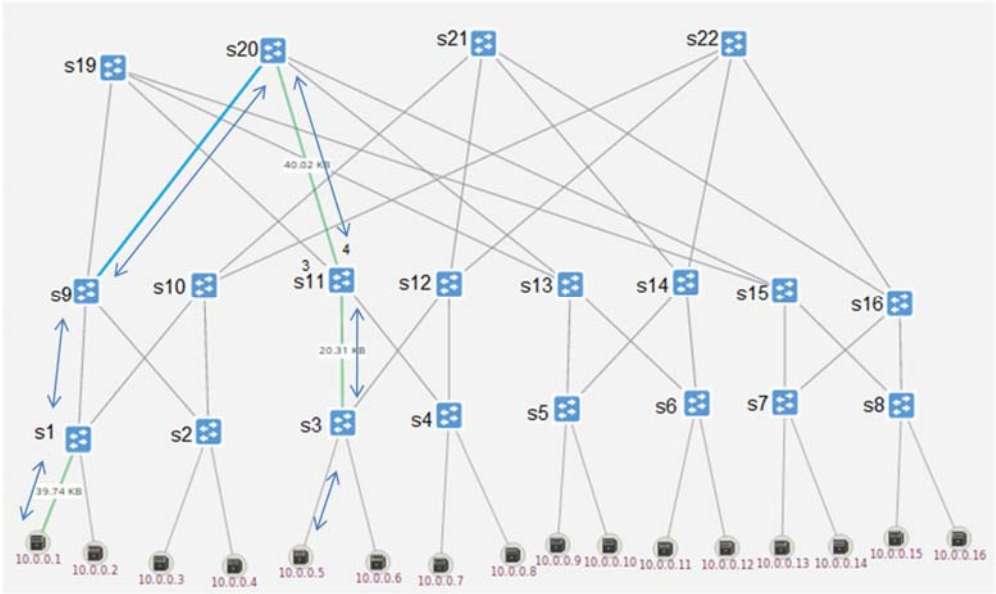

Figure-8. Traffic flow over threshold.

Figure-8 shows the alternate path triggered when the threshold limit is exceeded. The threshold limit was given as 300 packets. The data flow takes place between host 1 and host 5. In this alternate path the switchess1, s9, s20, s11 and s3 are used for packet transfer. From the time interval of 260 second the port 4 of the switch s11 becomes active for the ICMP packet transfer as shown in the Figure-9.

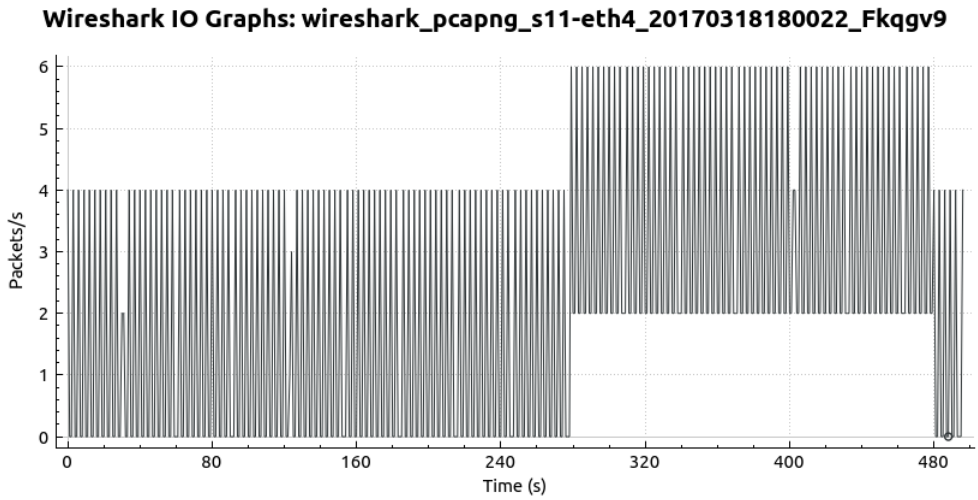

Figure-9. Packets transferred from switch 11 port 4.

```
--- 10.0.0.5 ping statistics ---
500 packets transmitted, 500 received, 0% packet loss, time 499014ms
rtt min/avg/max/mdev = 0.050/0.151/1.554/0.171 ms
```

Figure-10. Ping statistics of packet transfer.

In this work, we have sent 500 packets with a threshold limit of 300 packets. As discussed earlier the first 300 packets where transferred through the port 3 of switch 11 and when the threshold limit was exceeded the packets were transferred from port 4 of same switch. All the 500 ICMP packets were transmitted with 0% packet loss as shown in Figure-10.

Conclusion

In this paper, we present a strategy to solve the load balancing issue in SDN based Data Center networks. We created the fat-tree topology in a virtual network with the help of Mininet. We implemented packet based load balancing technique to redirect the traffic to different switches by setting a threshold. Results show that the packets transferred from one switch to another switch without any packet loss, thereby balancing the load of one core switch with the other.

REFERENCES

[1] Nunes B., Marc Mendonca X. Nguyen, Katia Obraczka and Thierry Turletti. 2014. A survey of software defined networking: Past, present, and future of programmable networks. 1-18.

[2] Kreutz D., Ramos F.M., Verissimo P.E., Rothenberg C.E., Azodolmolky S. and Uhlig S. 2015. Software-defined networking: A comprehensive survey. Proceedings of the IEEE. 103(1): 14-76.

[3] Zhou W., Li L., Luo M. and Chou W. 2014, May. REST API design patterns for SDN northbound API. In Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference on (pp. 358-365). IEEE.

[4] Kaur S., Kumar K., Singh J. and Ghumman N.S. 2015, March. Round-robin based load balancing in Software Defined Networking. In Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on (pp. 2136-2139). IEEE.

[5] Yahya W., Basuki A. and Jiang J.R. 2015. The Extended Dijkstra's-based Load Balancing for OpenFlow Network. International Journal of Electrical and Computer Engineering. 5(2): 289.

[6] Wang L. and Lu G. 2016, January. The dynamic sub-topology load balancing algorithm for data center networks. In Information Networking (ICOIN), 2016 International Conference on (pp. 268-273). IEEE.

[7] ONOS: https://wiki.onosproject.org/display/ONOS/Wiki+Home.

[8] Mininet: http://mininet.org/.

[9] Yang Liu, Jogesh K. Muppala, Malathi Veeraraghavan. 2014. A Survey of Data Center Network Architectures.

[10] Lan Y.L., Wang K. and Hsu Y.H. 2016. July. Dynamic load-balanced path optimization in SDN-based data center networks. In Communication Systems, Networks and Digital Signal Processing (CSNDSP), 2016 10th International Symposium on (pp. 1-6). IEEE.

[11] OpenFlow: http://archive.openflow.org.