Performance analysis of Software Defined Network (SDN) in link failure scenario

IOP Conf. Series: Materials Science and Engineering **557** (2019) 012028 doi:10.1088/1757-899X/557/1/012028

Performance analysis of Software Defined Network (SDN) in link failure scenario

M A I M Sakari[1], N Yaakob[1], A Amir[1], R B Ahmad[1,2], M N M Warip[1] and Z Ibrahim[3]

[1]School of Computer and Communication Engineering (SCCE), Universiti Malaysia Perlis (UniMAP), 01000 Arau Perlis
[2]Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin (UniSZA), Besut Campus, 22200 Besut Terengganu
[3]Technopreneur at UniMAP Sdn Bhd, Blok D, Bangunan PPPIT UniMAP, Jalan 3 Pengkalan Asam, 01000 Kangar Perlis

Abstract. The proliferation and advancement in network technologies has brought unprecedented opportunities towards the development of Software Defined Network (SDN) which presents an alternative network architecture that separates data forwarding and control functions to simplify global network management. As SDN is developed to handle large-scale network, addressing network dynamics such as link failure remains challenging. Link failure in SDN could still cause major performance disruption which can lead to severe performance degradation for underlying applications. Therefore, this paper analyses several link failure circumstances in SDN and show how it reacts to such situations. As SDN decouples data and control planes, routing algorithm is determined by its controller. In this paper, routing algorithm is used to control the behaviour of the network traffic when one or more links are disconnected. Ryu controller which is one of the open controllers in SDN, acts as a brain to adapt how traffic is controlled for the whole network. Performance of SDN in different network topologies is compared in terms of throughput and Round-Trip-Time (RTT) which shows desirable SDN performance.

1. Introduction

Ensuring uninterrupted Internet services and providing high Quality-of-Services (QoS) is becoming burgeoning demand in fulfilling industry-specific regulations. This alternative is towards improving future Internet and service mobility to enhance overall user experience. As such, networking protocols have evolved significantly over the last few decades. However, traditional network configuration is time-consuming and error-prone due to manual configuration by network administrator. Moreover, the traditional network configurations also complicates network segmentation where all devices are placed in the same 'zone'. Such configuration faces high risk of route diversion during link failure scenarios. Figure 1 shows a traditional network that is made up of packets switches and several clients connected through Ethernet connection. This design complicates administrative task for consistent policies deployment. As a result, organizations are more likely to encounter some failures in network links. To resolve this issue, a new paradigm on network management which is called Software Defined Networking (SDN) has been introduced as shown in Figure 2 [1].

SDN is a new networking architecture which is proposed to overcome the limitations in the traditional network. The aim of SDN is to ease network management and allow network administrators to quickly react on dynamic topology change such as link failure [2]. In addition, SDN introduces new networking architecture which decouples network controller and data transmission functions which make it directly programmable. Due to its promising features, SDN offers a cost-effective, dynamic, manageable and adaptable networking, making it well suited to handle dynamic nature of today's applications [3]. Moreover, the adaptable nature in SDN architecture makes it very convenient for network administrators to handle large and unprecedented program, automation and fully control of the network.

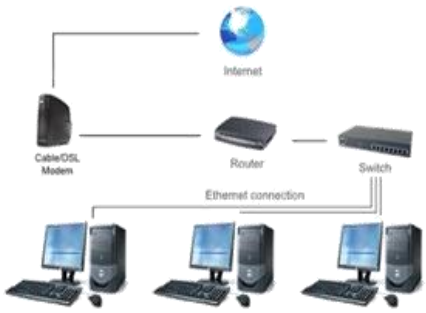

Figure 1. Traditional Network

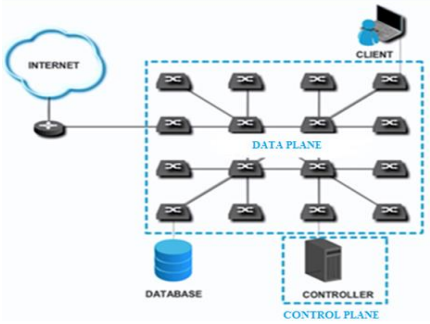

Figure 2. Architecture of SDN

This kind of network will also allow better network management and optimization of network resources. Figure 2 shows the connection between client and database server in SDN which consists of control plane (controller) and data plane. Unlike the traditional networking, SDN can simulate link failure scenarios without affecting the network architecture.

Link failure is one of the common problems in both the traditional network as well as SDN [4]. This problem is sometimes inevitable especially in network with large data centre. In SDN, a link failure in network architecture happens when switches are disconnected with controller or during switch replacement. Link failure scenario could generate large amount of unexpected traffic in the network due to the rerouting and retransmission of packets [5]. In link failure scenario, a network administrator in SDN handles traffics from a centralized controller without having to look at real network switches. However, link failure in SDN will still cause major performance disruption which can severely affect the underlying applications. Nevertheless, the performance of traditional network in link failure in terms of packet loss, throughput, end-to-end delay, and packet delivery ratio will be much affected compared to SDN. This is due to difficulties in identifying and analysing the problem in link failure scenarios due to large number of routers over the Internet and variations in network topology. This paper investigates SDN's performance and reaction in several link failure scenarios and topologies. Performance analysis of several link failure scenarios in SDN have been simulated and analysed using Mininet environment and Ryu controller.

2. Simulation Design and Testing Environment

To simulate SDN performance in link failure scenarios, simulation environment have been designed in several topologies; basic and custom. The basic topology includes the single and linear topology, while custom topology consists of triangle and fat-tree design. Each topology is simulated with OpenFlow-enabled network architecture using Mininet simulator. The default topology is minimal topology which is predefined with one OpenFlow kernel switch connected to two hosts and OpenFlow reference controller, whereas number of switches and hosts can be changed for other topologies using the command-line interface (CLI).

2.1. Single Topology

Figure 4 shows a single topology which consists of a single OpenFlow-enabled switch connected with multiple hosts. The switch is connected to the OpenFlow controller via a secure channel. A linear topology with 8 hosts connected with its own switch in linear fashion is clearly shown in Figure 3. All switches are interconnected with each other and are in turn connected with OpenFlow controller.

2.2. Linear Topology

Linear topology requires n number of switches for n number of hosts. In other words, each host will be connected with their respective switch as shown in Figure 4. For example, host H1 will connect with switch S1, host H2 with switch S2. All the switches are connected with one another which in turn connected with a common controller. A linear topology in this paper is designed to have 8 hosts and thus 8 switches.

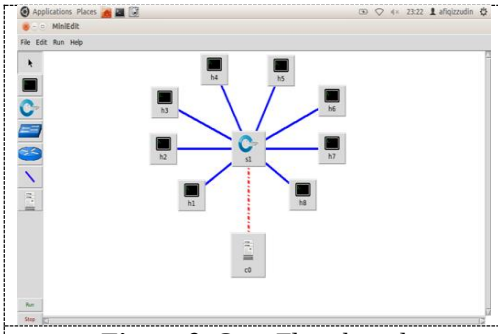

Figure 3. OpenFlow-based Single Topology having 8 Hosts

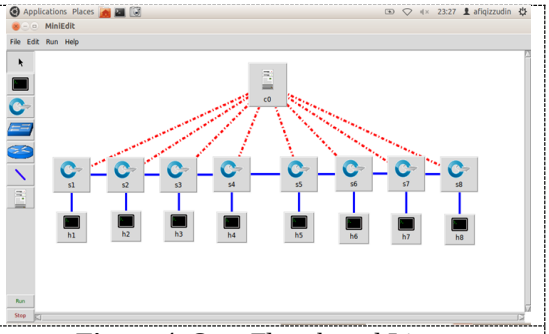

Figure 4. OpenFlow-based Linear Topology having 8 Hosts

2.3. Triangle Topology

Triangle topology consists of three switches which are interconnected with each other. As shown in the Figure 5, only switch S1 has two hosts, the other 2 switches (*S2* and *S3*) has 3 hosts. All IP addresses is changed from the default set by Mininet in order to resemble real-world networks. Each switch will have its own default and static route.

2.4. Fat Tree Topology

Fat tree topology is one of data center topology that provides efficient communication between hosts. This custom topology is using multipath routing with 4-pod fat tree as shown in Figure 6.

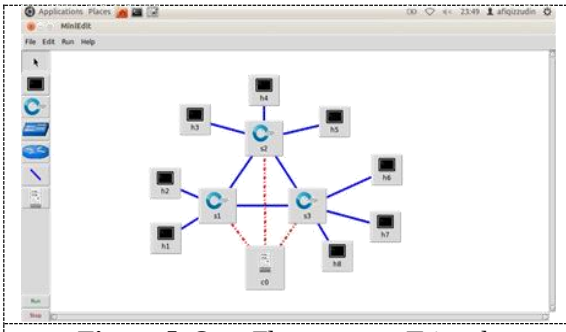

Figure 5. OpenFlow-custom Triangle Topology having 8 Hosts

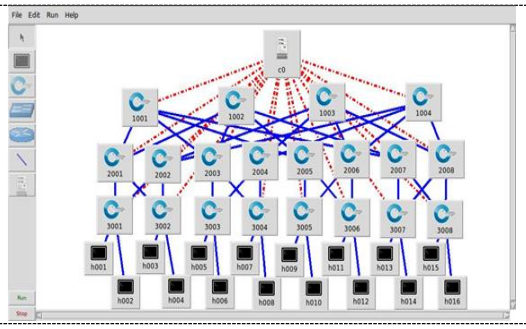

Figure 6. OpenFlow-custom Fat Tree Topology with 4 pods having 16 Hosts

The hierarchical layers of the network includes the core, aggregation, edge switch and hosts. Each switch has been renamed to differentiate switch for each layer. Assuming that the controller is at the top of the layer, then the 4 core switches are numbered from 1001~1004. Both second layer (aggregation switch) and third layer (edge switch) has 8 switches starting from 2001~2008 and 3001~3008 respectively. Hosts are numbered from h001~h016 and IP addresses are ranging from 10.1.0.1~10.8.0.2.

3. Link Failure Scenario Testing

Two methods are used to simulate link failures in this paper. The first method is by disabling a specific link interconnected between switches or hosts and the second method is to completely shut down all switches which disable communication between hosts in the network. *Ping* tests were run inside Mininet CLI to test the connection in each scenario. The connection was tested between the first host and last host in each of topology. The performance are measured in terms of Round Trip Time (RTT), number of packet transmit and receive, percentage of packet loss and transmission time. Figure 7 and Figure 8 show examples of link failure scenario in triangle and combination scenarios respectively.

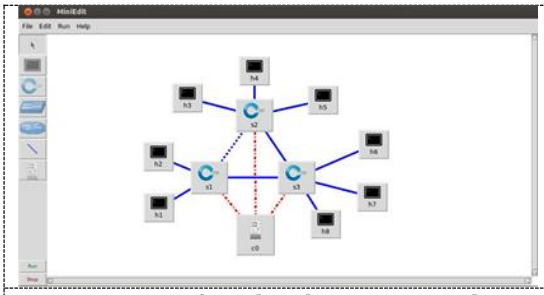

Figure 7. Link Failure between s1 and s2

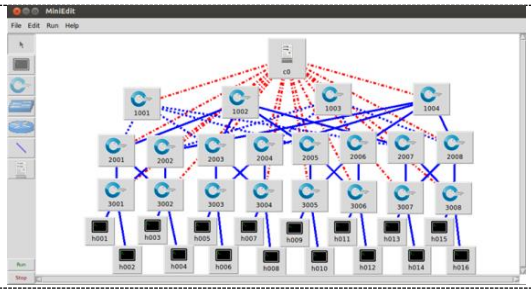

Figure 8. Example of Switch Failure in fat-tree

4. Performance Analysis

A comparative performance analysis of a single, linear, triangle and fat-tree topology were conducted in Mininet simulator. The basic topologies are simulated with fully working link since a link failure in such network will cause a major breakdown. Hence, performance analysis for customs networks is done by comparing all network topologies on the basis of packet transmission rate, Round-Trip-Time (time required to transmit packet from source node to destination node) and throughput.

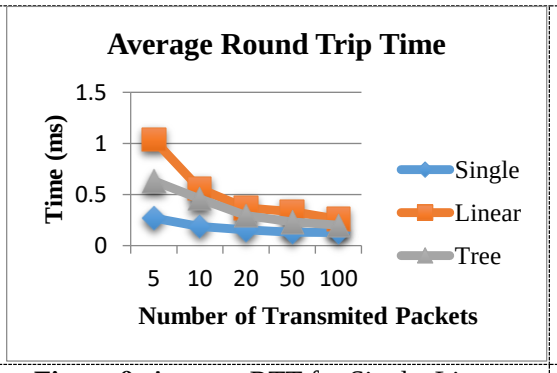

Figure 9. Average RTT for Single, Linear and Tree Topology

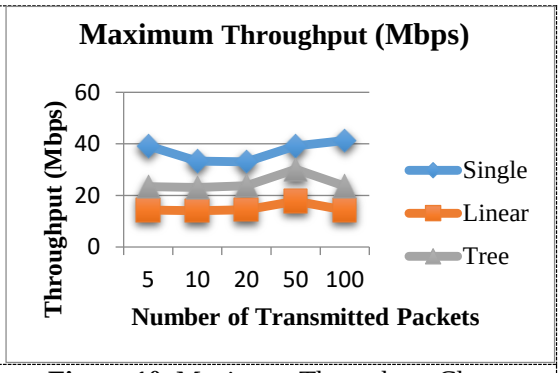

Figure 10. Maximum Throughput Chart for Single, Linear and Tree Topology

Figure 9 shows the average RTT for basic network topologies of single, linear and tree topologies while Figure 10 shows the resulted throughput for the same topologies. From the figure, it is clearly shown that a linear topology is taking more time for transmission of packet as compared to single and tree topology. This is due to the increase in the number of hops between end nodes thus resulting in longer propagation time for intermediate nodes to deliver packet to destination.

However, single topology is taking minimum amount of time to deliver packets to its destination since all nodes are connected with a single OpenFlow-enabled switch. As such, packet transmission is faster in single topology. Figure 10 shows the maximum throughput with respect to packet transmission rate. The results shows that linear topology obtained the lowest throughput compared to the other two topologies because the bandwidth utilization is less and the overall round-trip propagation delay between nodes is higher. Whereas, single topology network is having maximum throughput since the single topology only have one switch and between any two nodes in a network consists of only two hops. Thus, single topology incur less delay and higher throughput whereas linear topology incur more delay and less throughput.

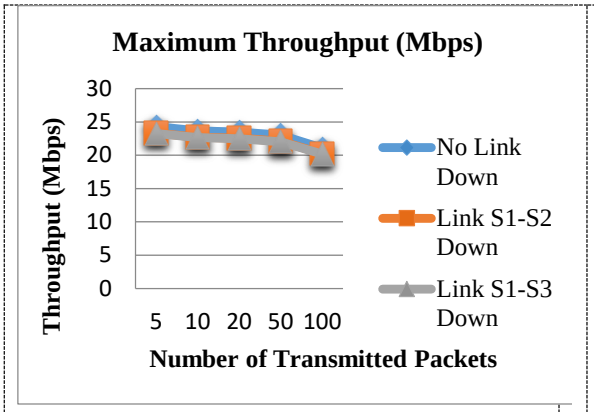

Figure 11. Maximum Throughput for Triangle Topology

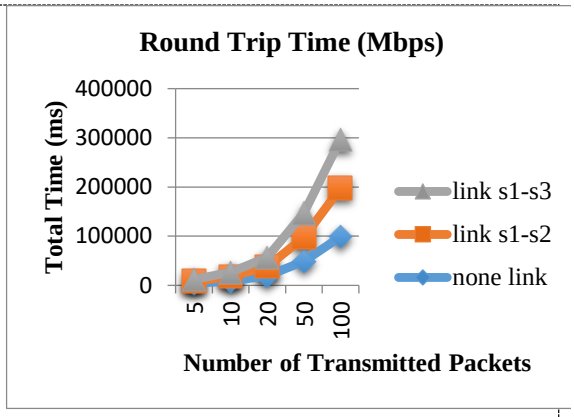

Figure 12. RTT for Triangle Topology

Maximum throughput and packet transmission rate is as shown in Figure 11 and Figure 12 respectively. Figure 11 shows the throughput comparison in 3 scenarios. As shown, the throughput when *s1-s3* link is down in triangle topology is very less as compared to other two topologies. This is because the bandwidth utilization is less and thus the overall round-trip propagation delay between nodes is higher. Obviously, when no link is down, maximum throughput is achieved since the network has no failure so all packets can be transmitted. Referring to Figure 12, total time taken for *h1* to transmit packets to *h8* when link *s1-s3* is down is more than when there is no link failure. When link *s1-s3* fail, the controller will provide a new routing table for *h1* to reach *h8*. As the rerouting will incur some delay, it is proven a link down will affect the total time taken for transmitting packets. In link failure scenarios, switch in triangle topology will follow routing table provide from Ryu Controller.

5. Conclusion

This paper investigates performance of SDN in different link failure scenarios and topologies. Performance comparison in term of throughput, and Round-Trip-Time (RTT) have been presented and discussed. Based on the results obtained, it is observed that SDN is capable of handling link failure scenarios by rerouting traffics to alternative links. In such cases, packets can still be transmitted to their destinations with the increase in RTT and lower throughput.

Acknowledgments

This paper is financially funded by Technopreneur at UniMAP Sdn. Bhd. (TUSB) [Grant number: 2017/08/0006].

References

[1] “Traditional VS Software Defined Networking.” [Online]. Available: http://www.mavenspire.com/blog/traditional-vs.-software-defined-whats-the-differences.
[2] Huang Liaoruo, Shen Qingguo and Shao Wenjuan, "A source routing based link protection method for link failure in SDN," *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, Chengdu, 2016, pp. 2588-2594.
[3] T. Theodorou and L. Mamatas, "CORAL-SDN: A software-defined networking solution for the Internet of Things," *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Berlin, 2017, pp. 1-2.
[4] Huang Liaoruo, Shen Qingguo and Shao Wenjuan, "A source routing based link protection method for link failure in SDN," *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, Chengdu, 2016, pp. 2588-2594.
[5] S. Waleed, M. Faizan, M. Iqbal and M. I. Anis, "Demonstration of single link failure recovery using Bellman Ford and Dijikstra algorithm in SDN," *2017 International Conference on Innovations in Electrical Engineering and Computational Technologies (ICIEECT)*, Karachi, 2017, pp. 1-4.