Performance of software-defined networking controllers for different network topologies

Conference Paper · August 2017

DOI: 10.1109/PACRIM.2017.8121925

CITATIONS
7

READS
609

3 authors, including:

Kamel Alrashedy
University of Victoria
4 PUBLICATIONS **12** CITATIONS
SEE PROFILE

T. Aaron Gulliver
University of Victoria
848 PUBLICATIONS **6,098** CITATIONS
SEE PROFILE

Performance of Software-Defined Networking Controllers for Different Network Topologies

Kamel Alrashedy[1], Ben Kimmett[1] and T. Aaron Gulliver[2]
[1]Department of Computer Science
[2]Department of Electrical and Computer Engineering
University of Victoria
PO Box 1700, STN CSC, Victoria BC, Canada V8W 2Y2
Kamel@uvic.ca, blk@uvic.ca, agullive@ece.uvic.ca

***Abstract*—With the increasing complexity of networks, Software Defined Networks (SDNs) have been developed to help administrators configure and operate network services with controllers such as Beacon, OpenDaylight and Floodlight. These controllers provide a suitable platform for high bandwidth applications. In this paper, several SDN controllers are evaluated using different network topologies. Some controllers cannot operate on topologies containing loops such as Pox and Beacon. OpenDaylight can be used on topologies with loops, but does not provide any benefit for multiple paths, i.e. when there is more than one path for a packet to reach its destination. Floodlight is the most efficient and adaptable controller with topologies containing loops and multiple paths. The controllers are evaluated on Mininet using ping and iperf, and the initial packet latency and throughput for the topologies are compared.**

***Index Terms*—Software defined network (SDN), Pox, Beacon, OpenDaylight, Floodlight, loop topology**

I. Introduction

In a traditional computer network, network routing is performed by switches with built-in routing logic. In a Software Defined Network (SDN) [1], switches do not have routing logic. Instead, the routing logic is abstracted to a programmable SDN controller which takes the form of a computer algorithm. This algorithm tells the switches how to route network traffic. This aspect of SDNs gives the network flexibility, as the controller algorithm and switches can be added, replaced, or updated depending on the needs of the network. It also provides reliability as if a switch fails, it can be replaced without affecting the routing logic. In [2], the performance of four SDN controllers was tested with the Tree topology. The controllers were tested using the MiniNet virtual networking testbed software [3] running in a virtual machine. ping and iperf were used to measure the quality of the virtual network. This paper extends these results to multiple network topologies using currently available SDN controllers.

II. Experimental Method

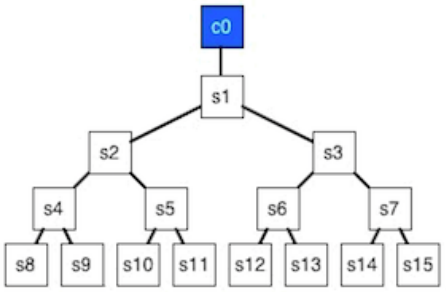
All experiments were run in a MiniNet [3] instance using the VirtualBox virtualization software on a Linux PC. The Pox [4], Beacon [5], OpenDaylight (ODL) [6] and Floodlight [7] controllers were used in the experiments. Floodlight was provided by the Floodlight developers, while Beacon, Pox and OpenDaylight were provided by the MiniNet developers. The controllers are denoted by C_0 in the figures. Five topologies were used in the experiments, and these are described below.

The Tree topology is shown in Fig. 1a and is the same topology used in [2]. This topology consists of 15 switches numbered S_1 to S_{15} which are connected to 16 hosts. The switches are arranged in a fully balanced binary tree layout such that switch S_1 is the root node of the tree, and switches S_8 to S_{15} are leaf nodes. Non leaf node S_n has S_{2n} and S_{2n+1} as child nodes. Two hosts are connected to each leaf node in order so that S_8 is connected to H_1 and H_2, and S_9 is connected to H_3 and H_4. The Tree topology is the only topology considered that does not contain loops. The Ring topology is shown in Fig. 1b. This topology has 16 hosts and 16 switches, and each switch is linked to the ones before and after it in numeric sequence. Thus, S_2 is linked to S_1 and S_3 and S_1 is linked to S_{16}. A host is linked to each switch.

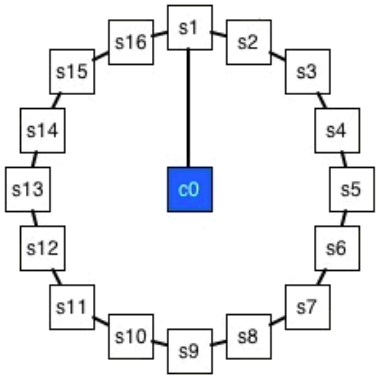
The Mesh topology is a randomized topology so that each time the MiniNet software is run a new topology for the switches is generated. A typical Mesh topology is shown in Fig. 2a. The Mesh topology generation method is as follows. A random link is chosen between two switches that have three or fewer existing connections and are not directly connected. This link is added to the network. This is repeated until all switches have at least two links to other switches. Once the switch connections have been made, the hosts are connected to the switches. Switches S_1 through S_4 are connected to two hosts each, denoted H_1 to H_8, and switches S_{13} through S_{16} are also connected to two hosts each, denoted H_9 to H_{16}. This topology may contain loops.

The Torus topology was inspired by the topologies used in data centers using software defined networking [8]. This topology has 16 hosts and 16 switches, and each switch is linked to four other switches. The switches are grouped into four 4-switch rows with each switch connected to the switches preceding and following it in the row. Each switch is also connected to switches in the preceding and following rows that are in the same column. A switch on the first row is linked to a switch in the preceding row which is the last row, and vice versa. This topology shown in Fig. 2b.

The Hypercube topology has 16 switches arranged in two



(a)



(b)

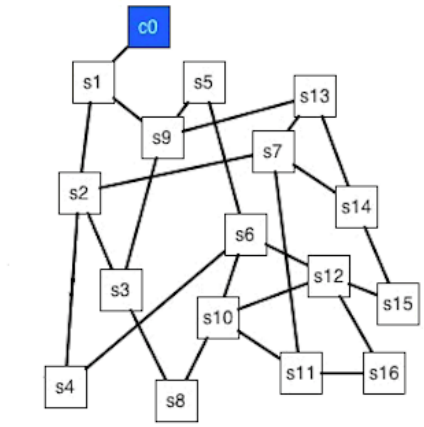
Fig. 1: The Tree (a) and Ring (b) topologies.

cubic shapes, as shown in Fig. 3. Each switch in a cube is connected to four other switches, three in the same cube and one in the same position in the other cube. Host H_n is connected to switch S_n.

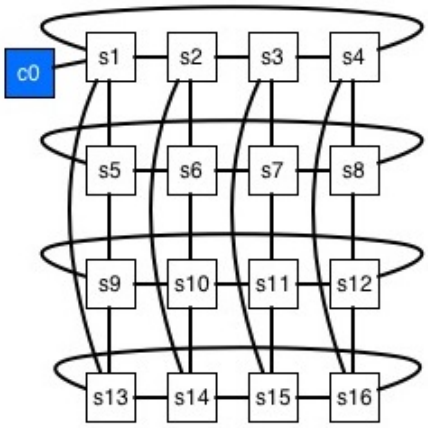
III. Performance Results

An experiment was conduced which consists of sending a ping command on a specific path in the topology. This command sends packets to a remote host, and then records the packet Round-Trip Time (RTT), and is commonly used for network connectivity and throughput testing. The path used was H_1 to H_{16} in the case of the Tree, Hypercube, and Mesh topologies, as it is a longest path in the Tree and Hypercube topologies, and likely to be a long path in the Mesh topology. For the Torus topology, the path H_1 to H_{11} was used as it is the longest path. The path H_1 to H_{16} was used with the Ring topology even though H_1 to H_9 is the longest path. The Tree and Hypercube topologies have the same longest path, but this is different for the other topologies. Ten pings were sent, and the minimum, median, and maximum ping times recorded. The iperf bandwidth testing tool was used to provide an estimate of the network throughput for each topology-controller combination, as well as the maximum achievable bandwidth.

Some controllers were not designed for network topologies with loops or multiple paths. Thus, the experiments were conducted using the most common controllers, OpenDaylight and Floodlight. The results of the ping experiments are



(a)



(b)

Fig. 2: The Mesh (a) and Torus (b) topologies.

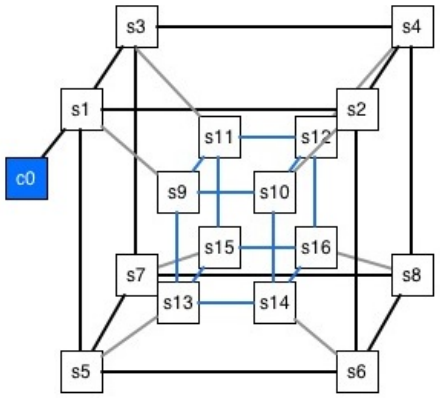


Fig. 3: The Hypercube topology.

given in Tables I and II. The Floodlight controller minimum and median ping times are always lower than those of the OpenDaylight controller. Thus, once Floodlight determines the destination, the subsequent ping times are much smaller. However, the difference between the first and median ping times with OpenDaylight are similar. This implies that the

efficiency of OpenDaylight does not change over time. The minimum and median ping times for all topologies are smaller with Floodlight than with OpenDaylight. However, the maximum ping time with Floodlight is greater than that with OpenDaylight for the Tree, Mesh and Torus topologies. The maximum ping times with the Hypercube topology are almost equal. The drawback with OpenDaylight is packet loss. During the experiment ten pings were sent, and the first three packets failed to reach the destination, which is a 30% packet loss.

TABLE I: OpenDaylight Ping Test Results

RTT (ms)	Tree	Ring	Mesh	Torus	Hypercube
Min	0.32	0.25	0.32	0.29	0.26
Median	0.38	0.35	0.39	0.36	0.37
Max	0.61	0.53	0.45	0.44	0.56

TABLE II: Floodlight Ping Test Results

RTT (ms)	Tree	Ring	Mesh	Torus	Hypercube
Min	0.09	0.06	0.06	0.07	0.06
Median	0.13	0.06	0.08	0.29	0.32
Max	0.76	0.29	0.61	0.56	0.55

The RTT was evaluated for two different paths with the Ring topology. First a ping was sent from H_1 to H_{16}. This is typically the longest path for the topology. It can also be the shortest path when the packet takes the direct link between switches S_1 to S_{16}. Then a ping was sent from H_1 to H_9, which is guaranteed to be a long path in the topology. The results are given in Table III. This shows that with the OpenDaylight controller, the two paths have similar minimum, maximum, and median ping times. This suggests that the packets followed the same route regardless of the path requested. Conversely, the minimum, median, and maximum ping times with the Floodlight controller increase when the requested path changes from H_1 to H_9 to H_1 to H_{16}. These results suggest that OpenDaylight sends packets through the longer path, while Floodlight uses the shorter path. Thus, while the OpenDaylight SDN controller algorithm can tolerate loops in the network topology, it is less adaptable when this situation arises, preferring to find a path and use it rather than searching for a more efficient route.

TABLE III: Ring Topology Ping Test Results

RTT (ms)	ODL h1-h16	ODL h1-h9	Floodlight h1-h16	Floodlight h1-h9
Min	0.33	0.25	0.06	0.09
Median	0.35	0.37	0.06	0.09
Max	0.43	0.41	0.27	0.77

The TCP bandwidth was evaluated for the five topologies using the iperf tool, and the results are given in Table IV. The Tree topology has the highest throughput with OpenDaylight, which is approximately 13 Gigabits per second (Gbps). The other topologies have similar throughput with a maximum of 0.114 Gbps. OpenDaylight provides a high bandwidth with network topologies that do not have a loop or multiple paths. With Floodlight, the Tree topology has the lowest bandwidth of 16.2 Gbps, while the other topologies have similar results with an average of 19.5 Gbps.

TABLE IV: Iperf Bandwidth Test (Gbps)

Controller	Tree	Ring	Mesh	Torus	Hypercube
OpenDaylight	12.9	0.114	0.048	0.113	0.111
Floodlight	16.2	20.0	18.6	19.3	19.8

The initial ping time results are shown in Table V. This is the time taken for the first ping with a given topology-controller combination. This provides an estimate of the time required for a controller algorithm to adapt to the network topology. As with the iperf results, Table V suggests that OpenDaylight takes longer to adapt to the network topology than Floodlight when the topology has loops or multiple paths. While OpenDaylight can run on networks with loops, it is not as efficient as it takes longer to adapt to the Torus and Hypercube topologies.

TABLE V: Initial Packet Time (ms)

Controller	Tree	Ring	Mesh	Torus	Hypercube
OpenDaylight	1.38	1.98	999	1.00	998
Floodlight	15.4	8.25	19.4	61.6	66.7

IV. Conclusion

Software Defined Networks (SDNs) are a new paradigm for the administration of network services. In this paper, SDN controllers were tested using ping and iperf with the Mininet virtual network. Numerous combinations of controllers and network topologies were considered. The results presented show that the Floodlight SDN controller is efficient and can adapt to the topology. In particular, it can take advantage of loops and multiple paths in the network topology. Some controllers perform poorly with topologies containing loops, so they are unsuitable for many modern networks. The OpenDaylight controller algorithm can adapt to topologies with loops, but does not exploit the existence of multiple paths to improve the performance.

V. Acknowledgement

Kamel Alrashedy acknowledges the financial support of the Saudi Ministry of Education through a graduate scholarship.

References

[1] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[2] A. L. Stancu, S. Halunga, A. Vulpe, G. Suciu, O. Fratu, and E. C. Popovici, "A comparison between several software defined networking controllers," in *Proc. Int. Conf. on Telecommun. in Modern Satellite, Cable and Broadcasting Services*, Nis, Serbia, Dec. 2015, pp. 223–226.

[3] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proc. ACM SIGCOMM Workshop on Hot Topics in Networks*, Monterey, CA, USA, Oct. 2010.

[4] A. Al-Shabibi, “POX controller,” Jun. 2017. [Online]. Available: https://openflow.stanford.edu/display/ONL/POX+Wiki.
[5] D. Erickson, “The Beacon OpenFlow controller,” in *Proc. ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, Hong Kong, China, Aug. 2013, pp. 13–18.
[6] OpenDaylight, “OpenDaylight: A Linux foundation collaborative project, 2013. [Online]. Available: http://www.opendaylight.org.
[7] Project Floodlight, “Floodlight is a Java-based OpenFlow controller, 2012. [Online]. Available: http://www.projectfloodlight.org/floodlight/.
[8] B. Andrus, J. J. V. Olmos, V. Mehmeri, I. T. Monroy, S. Spolitis, and V. Bobrovs, “SDN data center performance evaluation of torus and hypercube interconnecting schemes,” in *Proc. Advances in Wireless and Optical Commun.*, Riga, Latvia, Nov. 2015, pp. 110–112.