Research Article

Load and stress testing for SDN's northbound API

Majd Latah[1] · Levent Toker[2]

Received: 21 August 2019 / Accepted: 17 December 2019 / Published online: 20 December 2019

Abstract
In this work, we apply load and stress testing for well-known Software defined networking (SDN) controllers from an SDN application perspective. More precisely, we focus on the communication between the controller and SDN applications via the northbound Application programming interface (API). We apply proper load and stress testing plans, in order to correctly capture the behaviour of the controllers under consideration. Our load testing includes applying gradually increased workloads to find the throughput each controller can handle. Our stress test, on the other hand, builds upon the results of the load test and includes (1) measuring the API's ability to handle extremely high workloads for a prolonged period of time and (2) directly attacking the underlying hosts of SDN network using Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks. We considered POX, Ryu, Floodlight, OpenDayLight (ODL) and Open Network Operating System (ONOS) SDN controllers. The experimental results showed that ONOS and ODL followed by Floodlight achieve the best throughput. Whereas POX and Ryu are characterized by lower throughput accompanied with partial and/or continuous failures during high workloads or DoS/DDoS attacks.

Keywords Software defined networking (SDN) · Load and stress testing

1 Introduction

Software Defined Networking (SDN) represents a new networking architecture that combines central management and network programmability. SDN separates the control layer from the infrastructure layer and transfers the network management to a central point, called the controller, which can be seen as the brain of the network [1]. SDN controllers provide an application programming interface (API), usually northbound API, to communicate between the SDN controller and the services and applications running over the network. A simplified SDN architecture is shown in Fig. 1, where the controller represents the brain of the network.

Unlike previous works [2–13] which focus on testing each controller's southbound APIs, we apply load and stress testing on controller's northbound APIs. We considered well-known open source SDN controllers, which are: POX, Ryu, Floodlight, OpenDayLight (ODL) and Open Network Operating System (ONOS). POX and Ryu are single-threaded controllers whereas Floodlight, ODL and ONOS support multi-threading. Both ODL and ONOS are candidates for enterprise-scale SDNs due to their enhanced architecture.

In this work, we measure the throughput of each controller based on gradually increased loads that reach 1000 requests per second. Then, we investigate each conroller's ability to handle unexpected cases such as DoS/DDoS attacks and prolonged heavy workloads that exceed its average throughput. Accordingly, we examine the throughput of different SDN controllers as well as potential partial and continuous failures that may occur during the experimental test. Partial failure indicates non-consecutive failed API requests whereas continuous failure indicates consecutive failures of API requests. Moreover, we

✉ Majd Latah, majd.latah@ozu.edu.tr | [1]Department of Computer Science, Ozyegin University, Istanbul, Turkey. [2]Department of Computer Engineering, Ege University, Izmir, Turkey.

SN Applied Sciences (2020) 2:122 | https://doi.org/10.1007/s42452-019-1917-y

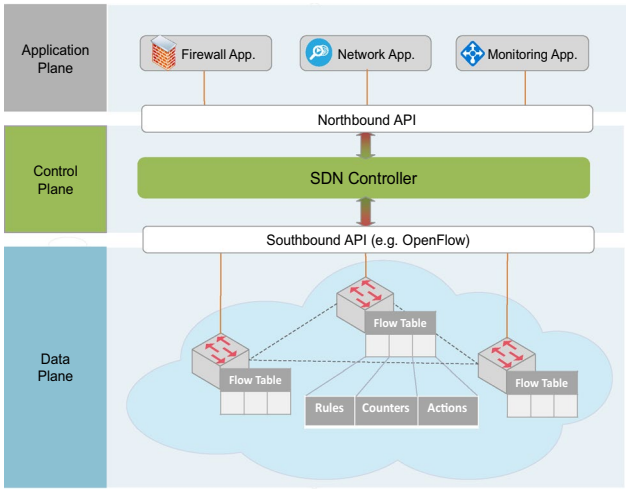


Fig. 1 A basic SDN architecture [1]

considered on-premises and off-premises (cloud-based) deployment of SDN network.

2 Related works

Laissaoui et al. [2] evaluated the performance of Beacon, Floodlight, Ryu and POX SDN controllers in terms of latency and throughput based on Cbench benchmarking tool. They found that Floodlight achieves the best results in terms of latency. Whereas Beacon achieves slightly better results than Floodlight in terms of throughput. They also observed that Ryu achieves better results than POX in terms of throughput however it achieves the worst results in terms of latency when the size of the network increases. This study is emulated on a virtual machine rather than a real SDN experiment.

Khattak et al. [3] compared the performance of OpenDayLight (ODL) with Floodlight SDN controller under different scenarios in terms of latency and throughput using Cbench. They found that ODL has lower throughput when compared to Floodlight. They also suggested that ODL may suffer from memory leakage problem. This study was conducted as a real SDN experiment. Additionally, they modified Cbench tool in order to add the ability to generate probabilistic traffic models.

Fancy and Pushpalatha [4] also compared POX and Floodlight in terms of delay and throughput. They considered star, linear and tree topologies. They found that Floodlight achieves better performance than POX. This study was conducted using Mininet emulator.

Rastogi and Bais [5] compared POX and RYU in terms traffic handling capabilities. The experiments were conducted using Mininet with a simplified data center topology. As a result, they found that POX outperforms RYU in terms of layer 1 switching, whereas RYU shows far betters results in terms of layer 2 switching.

Darianian et al. [6] evaluated ONOS and ODL in terms of throughput, latency and thread scalability in physical and virtualized (OpenStack) environment based on Cbench tool. The experimental results showed that ONOS can achieve higher throughput and lower latency than ODL. They also observed that each controller's throughput decreases when they increase the number of threads due to the communication overhead between different threads on different virtual cores. In addition, this work refuted Khattak et al [3]'s observation regarding ODL's memory leakage.

Mamushiane et al. [7] evaluated Ryu, Floodlight, ODL and ONOS in terms of latency and throughput, again based on Cbench tool. Apart from the previous works [2-6], they studied the effect of network load on each controller. The experimental results showed that ONOS achieves the best throughput. Ryu and ODL showed the best latency. Whereas ODL, Ryu and Floodlight were significantly affected by the network workload. Overall, ONOS showed the best results.

Stancu et al. [8] compared the performance of POX, Ryu, ODL and ONOS. They considered tree topology and found that ONOS achieves the best performance in terms of delay and throughput. Rowshanrad et al. [9] compared Floodlight and ODL in terms of delay and loss in different topologies and network loads. The experimental results showed that ODL outperforms Floodlight in low-loaded networks and also for tree topologies in mid-loaded networks in terms of latency. Whereas Floodlight outperformed ODL in heavily loaded networks for tree topologies in terms of packet loss, and in linear topologies in terms of latency.

Shalimov et al. [10] conducted a comprehensive analysis of the following open source SDN controllers: NOX, POX, Beacon, Floodlight, MuL, Maestro, and Ryu. Beacon showed the best results in terms of latency and throughput whereas POX and Ryu showed the worst results due to lack of multi-threading support.

Tootoonchian et al. [11] compared the performance using four open-source SDN controllers: NOX, NOX-MT, Beacon, and Maestro. The experimental results showed that NOX-MT has the highest throughput, and the least response time compared to other SDN controllers used in this study. NOX-MT is a multi-thread extension of the single-threaded NOX controller and uses optimization techniques such as Input/Output (I/O) batching and Boost Asynchronous I/O (ASIO) library in order to simplify multi-thread operations.

Zhao et al. [12] conducted two separate comparisons. In the first comparison, they compared the performance of

centralized controllers including Ryu, Pox, Nox, Floodlight and Beacon controllers. In the second comparison, on the other hand, they compared the performance of distributed controllers including ODL and ONOS. The experimental results for centralized controllers showed that Beacon achieves the lowest latency and the highest throughput. Whereas the experimental results for distributed controllers showed that ONOS achieves the best results for the same metrics. Bholebawa and Dalal [13] conducted a performance analysis between POX and Floodlight over different network topologies (single, linear, tree and custom) in terms of throughput and latency. The experimental results showed that Floodlight outperforms POX.

Overall, one can observe that the previous works [2–13] focused only on evaluating the communication between the controller and the underlying OpenVSwitches (i.e. testing the southbound API). Our work, on the other hand, takes a step forward by applying load and stress testing on the communication between the controller and the application programming interface (i.e. testing the northbound API). To the best of our knowledge, this is the first work that applies load and stress testing for SDN controllers from an SDN application perspective.

3 Background

3.1 Methodology

For load testing we aim to answer the question of *"How much throughput can each controller provide ?"* Therefore, we measure the throughput of SDN controllers using different workloads. We observe the throughput of each controller while gradually increasing the workload. Increasing the workload also increases the requests that must be handled per second and raises the chance of bottlenecks. If the controller is not able handle the workload, it will end up with partial or continuous failure.

In the next stage, as we already measured the average throughput of each controller from our previous load test, we can build upon this performance metric in order to apply a proper stress testing that examines the behaviour when we exceed the expected number of requests each controller can handle. Therefore, for stress testing we aim to answer the following question: *"How each controller acts under unexpected workloads or attacks ?"*. This also includes measuring the controller's ability to handle extreme workloads for a prolonged period of time. We considered two test cases (1) when the application layer is under attack (i.e. attacking the northbound API), and (2) when the infrastructure layer is under attack (i.e. attacking various hosts in the network). Therefore, we considered (i) exceeding the average throughput of the northbound API provided by each controller and (ii) directly attacking the hosts of SDN network using Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks.

3.2 Tools and libraries

Our experiments considered POX ,[1] Ryu,[2] Floodlight,[3] OpenDayLight (ODL)[4] and Open Network Operating System (ONOS)[5] SDN controllers. We used JMeter[6] for load testing. In addition, we used JMeter-Plugins[7] to simulate users with specific RPS (Requests Per Second). Apart from that we utilized Mininet,[8] an open-source SDN emulator, to emulate an SDN network and scapy[9] library to generate appropriate (D)DoS traffic on the underlying network. Moreover, we employed Grafana[10] and Influxdb[11] to implement a real-time performance monitoring dashboard.

4 Experimental setup

In order to conduct our test, we used Mininet with a simple linear topology that consists of 3 hosts. Our experiments were conducted in a virtual environment using Ubuntu 14.04 LTS which works on Intel i5 processor running at 2.3GHz with 12 GB of RAM where we allocated 8 GB of RAM and 4 CPU cores to our VM. We used Apache JMeter, a well-known open source load and stress testing tool, which runs on the host OS. For our on-premises setup, the average round-trip time (RTT) is < 1 ms. In JMeter a test plan determines a series of steps that need to be executed. A test plan mainly consists of one or more thread groups,

[1] POX (2019) SDN Controller [online]. Website https://noxrepo.github.io/pox-doc/html/ Accessed 12 April 2019.

[2] Ryu (2019) SDN Controller [online]. Website https://osrg.github.io/ryu/ Accessed 12 April 2019.

[3] Floodlight (2019) SDN Controller [online]. Website http://www.projectfloodlight.org/flood-light/ Accessed 12 April 2019.

[4] ODL (2019) SDN Controller [online]. Website https://www.opendaylight.org/ Accessed 12 April 2019.

[5] ONOS (2019) SDN Controller [online]. Website https://wiki.onosproject.org/ Accessed 25 Sept 2019.

[6] Apache JMeter (2019) Load Testing Tool [online]. Website https://jmeter.apache.org Accessed 05 March 2019.

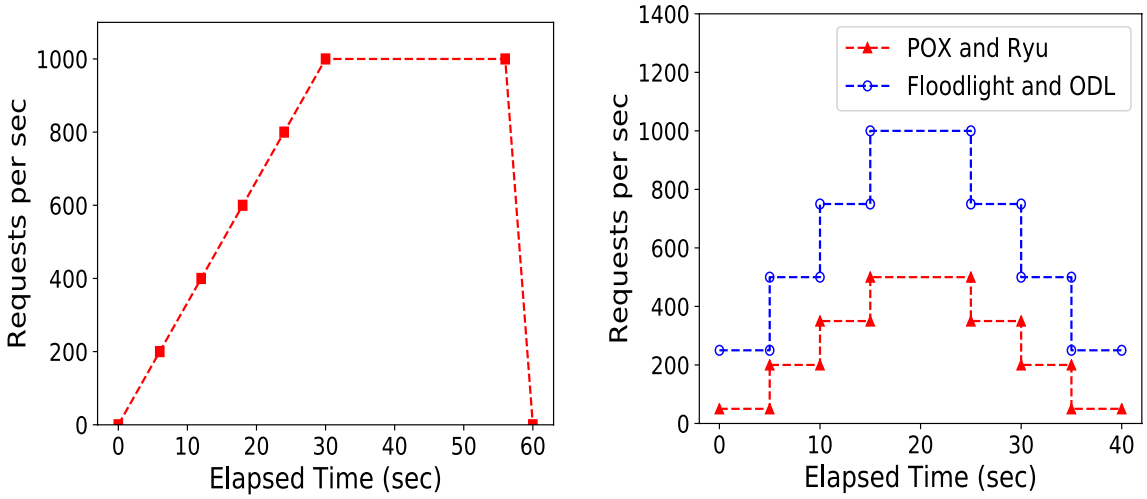
[7] Apache JMeter-Plugins (2019) [online]. Website https://jmeter-plugins.org/ Accessed 12 March 2019.

[8] Mininet (2019) SDN Emulator [online]. Website http://mininet.org/ Accessed 01 May 2019.

[9] Scapy (2019) Python Packet Manipulation Library [online]. Website https://scapy.net/ Accessed 03 May 2019.

[10] Grafana (2019) Real time performance monitoring tool [online]. Website https://grafana.com/ Accessed 12 May 2019.

[11] Influxdb (2019) Time series database [online]. Website https://www.influxdata.com/ Accessed 12 May 2019.



(a) Load test workload **(b)** Stress test workload

Fig. 2 Workload profile for our load and stress testing

sample generating and logic controllers, listeners, timers, assertions and configuration elements. JMeter-Plugins provide a custom set of plugins for Apache JMeter that can be utilized to boost JMeter's capabilities. One of these plugins is Throughput Shaping Timer, which can be used to simulate users with specific RPS (Requests Per Second) throughput without the need for tuning the number of threads and timers. We set the maximum number of concurrent threads to 200, which are used to satisfy the necessary RPS load schedule. We shared our test plan and related setup files on a public repository.[12]

As shown in Fig. 2a, for load testing the workload gradually increases to finally reach 1000 requests per second. This allows us to answer our first question we mentioned previously. At this point, partial or continuous failures may occur when we exceed the number of requests per second the controller can provide. In other words, once we reach a bottleneck, the throughput will decrease as the controller will be not able to handle such high workload.

On the other hand, we conduct a stress test for each controller based on its average throughput. In our experiment, DoS/DDoS attacks are launched from the same SDN network, which takes place in our VM machine. We did not implement any mitigation strategy in both guest and host operation systems or the interfaces hosting the corresponding SDN controllers, which also do not include any mitigation/detection module. We aim to exceed the average throughput of the northbound API provided by each controller by applying extremely high workloads for a prolonged period of time. As shown above in Fig. 2b we have two workload profiles. The first one is for single-threaded controllers (POX and Ryu) whereas the other one is for multi-threaded controllers (Floodlight, ODL and ONOS).

5 Experimental results

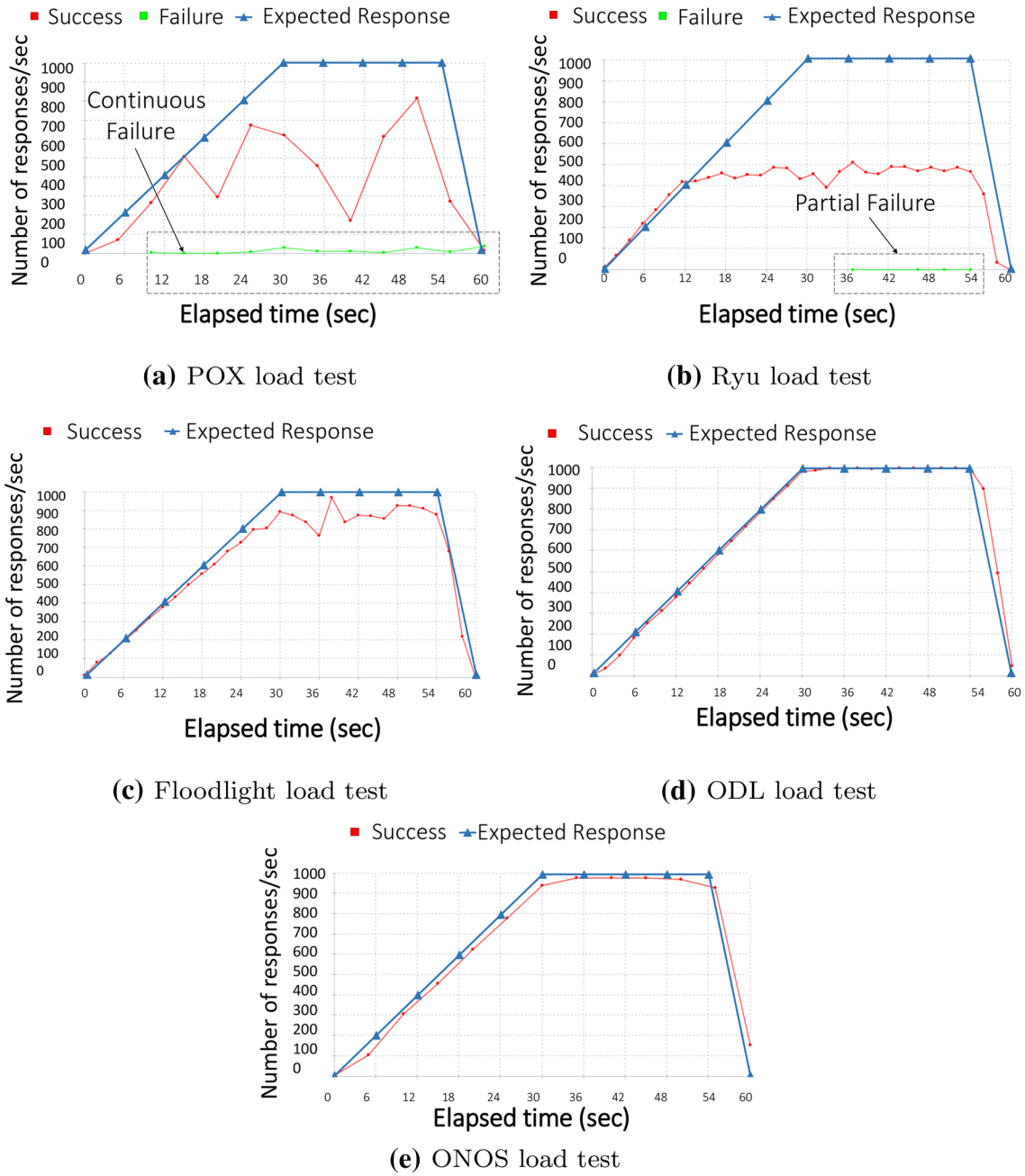
5.1 Results of our load testing

As we mentioned previously, partial failure indicates non-consecutive failed API requests whereas continuous failure indicates consecutive failures of API requests. From Fig. 3a, we observe that POX continuously fails when we exceed its average throughput (243.4 RPS). The obtained response is an order of magnitude lower of the expected one. Whereas Fig. 3b reveals that Ryu partially fails under extremely high workload (1000 requests per second). As a results, we conclude that Ryu is more stable than POX. Figure 3c shows that Floodlight was not able to achieve 1000 RPS due to its high response time in compared with ODL and ONOS. Figure 3d, e, on the other hand, show that both ODL and ONOS achieve higher throughput, however ODL achieves slightly better results in compared with ONOS.

As shown below in Table 1, POX and Ryu achieved the worst results in terms of average throughput and error rate. The best throughput was achieved by ODL followed ONOS and Floodlight with zero error.

Moreover, we used Google Cloud to implement a simple off-premise SDN network. To this end, we setup a separate VM for each controller. Mininet is used to simulate a simple linear topology that consists of 3 hosts. We launched

[12] SDNLoadTest [online]. https://github.com/majdlatah/SDNLoadTest.



(a) POX load test

(b) Ryu load test

(c) Floodlight load test

(d) ODL load test

(e) ONOS load test

Fig. 3 Results of our load testing (On-premises experiment)

Table 1 Results obtained from our load test

SDN controller	Average throughput (requests per sec)	Error rate (%)
POX	243.4	1.44
Ryu	272.8	0.08
Floodlight	595.67	0
ONOS	725.5	0
ODL	758.39	0

our JMeter test plan from our local machine, which works on Intel i5 processor with 12 GB of RAM. We used one-way delay (OWD) to estimate potential asymmetric delay between the SDN network and our local machine that runs JMeter test plan (i.e. we measured the time taken for a packet transmitting from the sender to the receiver). We used Network Time Protocol (NTP) for clock synchronization. Accordingly, we found that the delay from the SDN network to our local machine is 31.85ms whereas the delay from our local machine to the SDN network is 34.5ms. As

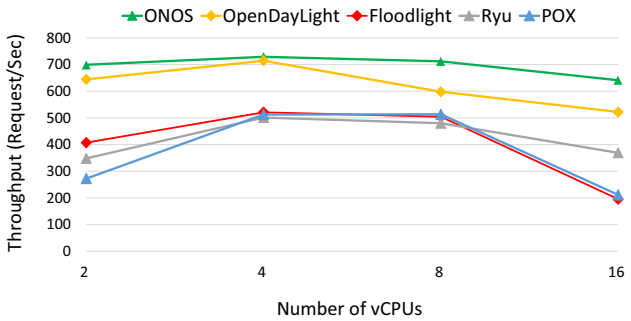
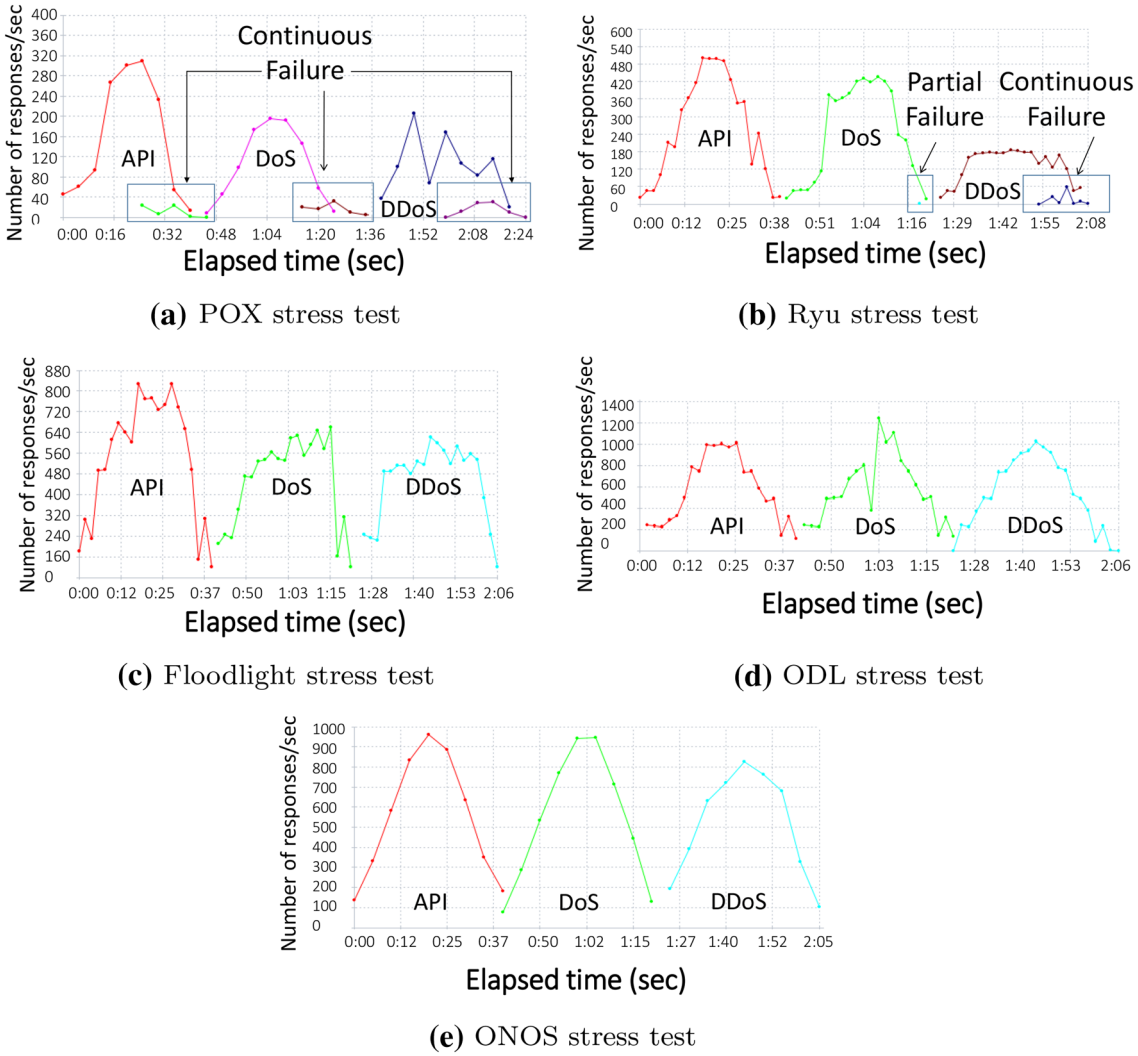


Fig. 4 Throughput versus number of vCPUs (Off-premises experiment)

a result, the reply packets sent from the SDN network are received faster than request packets sent by the JMeter due to less OWD, which may affect the throughput and the number of failed requests. In our experiment, we took into consideration that the number of virtual CPUs (vCPUs) needs to be tuned to obtain the best performance. As shown in Fig. 4, allocating a low number of vCPUs can lead to a low throughput. Also a high number of vCPUs can lead to a low throughput due to the communication overhead between threads on different vCores. The best performance was achieved by allocating 4 vCPUs for each controller.

5.2 Results of our stress testing

Our stress test takes into consideration the average throughput of each controller. Therefore, to answer our second question we gradually increase the workload on each controller's API for a prolonged period of time to see whether the controller can handle this case or not. We also considered launching DoS and DDoS attacks on the underlying network. As shown in Fig. 5a the performance of POX is accompanied



(a) POX stress test

(b) Ryu stress test

(c) Floodlight stress test

(d) ODL stress test

(e) ONOS stress test

Fig. 5 Results of our stress testing

Table 2 Results obtained from our stress test

Stress test	POX	Ryu	Floodlight	ODL	ONOS
API	CF	PF	–	–	–
DoS	CF+SDT	PF+DT	DT+HRT	DT	–
DDoS	CF+SDT	CF+SDT	DT+HRT	DT+HRT	DT

CF Continuous failure, *PF* partial failure, *SDT* significantly decreased throughput, *DT* decreased throughput, *HRT* high response time

with continuous failures in all cases of the stress test along with significantly decreased throughput during DoS and DDoS attacks. This is due to the fact that the controller could not handle extreme loads that exceed its average throughput. From Fig. 5b, on the other hand, one can observe that under DoS attacks Ryu shows partial failures along with significantly decreased throughput. Furthermore, under DDoS attacks Ryu demonstrates continuous failures along with significantly decreased throughput. Moving to Fig. 5c one can observe that Floodlight shows decreased throughput along with high throughput under DoS/DDoS attacks. Fig. 5d also shows that under DoS/DDoS attacks ODL can still achieve a high throughput. Fig. 5e, on the other hand, reveals that ONOS has a smoother decline in throughput in compared with Floodlight and ODL. We can also observe a slightly decreased throughput under DDoS attacks. The results of our stress test are summarized in Table 2.

5.3 Realtime monitoring of SDN's northbound API

We make use of Grafana and Influxdb to implement a real-time performance monitoring dashboard. Grafana allows us to query, visualize, alert on and understand the performance metrics. Whereas Influxdb is a time series database optimized for time-stamped or time series data. Fig. 6 shows our GUI, which updates the performance metrics periodically. Monitored metrics are throughput, total number of requests, number of successful requests and minimum response time. Accordingly, Jmeter sends these metrics to Influxdb using HTTP protocol allowing monitoring the performance of each controller in a real-time manner.

6 Discussion

This work emphasizes the importance of testing SDN's northbound API. It also shows how single-threaded SDN controllers can be significantly affected under unexpected workloads or when attacking the underlying SDN network. Interestingly, the results obtained from our off-premises experiment confirm our on-premises experiment with slight differences in terms of throughput. This can be due to the influence of asymmetric delay between the SDN network and our local machine that runs JMeter test plan.

The best results were achieved by ONOS and ODL potentially due to their enhanced architecture. Moreover, ONOS shows high performance with different number of allocated vCPUs and even under DoS/DDoS attacks. This observation is in line with [6–8, 12] in the sense that ONOS can achieve better performance than ODL. Apart from that, we observed that a high number of vCPUs can lead to a decreased throughput due to the communication overhead between threads on different vCores. The same observation was also confirmed by Darianian et al. [6].

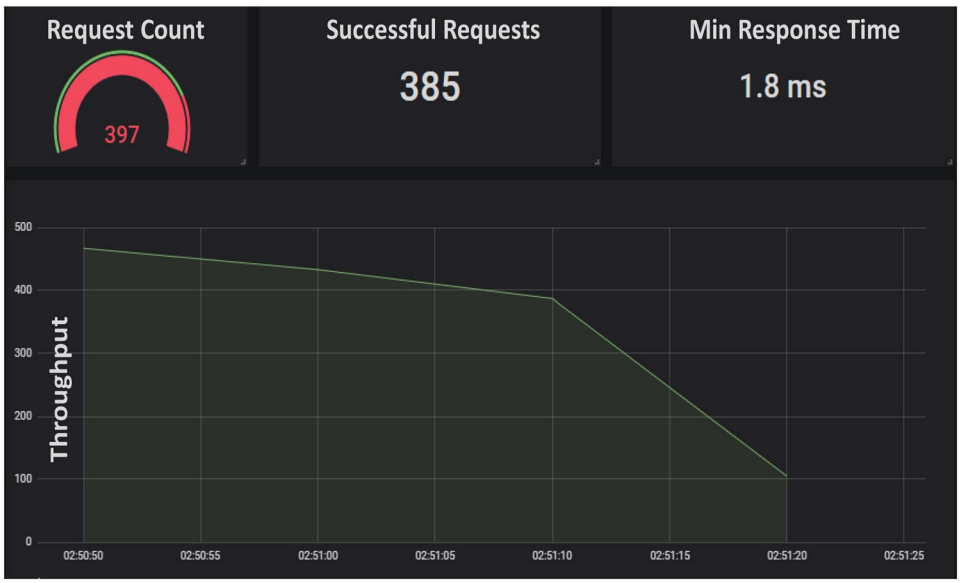


Fig. 6 Realtime monitoring of SDN's northbound API

Our overall results are also in consistence with the those of [2, 4–9, 13]. Additionally, our results are in line with [10–12] in the sense that multi-threaded controllers outperform single-threaded controllers. On the other hand, both our obtained results and those of [6] refute [3]'s conclusion regarding ODL's memory leakage. In other words, our results confirm that ODL outperforms Floodlight in terms of the throughput of its northbound API.

Overall, one can conclude that single-threaded SDN controllers (POX and Ryu) are a good option for fast prototyping rather than for enterprise deployment. Whereas multi-threaded SDN controllers (Floodlight, ODL and ONOS) can be used in the deployment of large-scale networks that include a wide-range of SDN applications. However, the best results are achieved by controllers with enhanced architecture (i.e. ODL and ONOS).

7 Conclusion

In this paper, we applied load and stress testing on well-known SDN controllers. Our work focused on testing the communication between the controller and SDN applications via the northbound API. Our experimental results showed that ONOS and ODL followed by Floodlight achieve the best results in terms of throughput and stability against unexpected cases such as DoS/DDoS attacks and prolonged heavy workloads. We believe that this work can pave the way towards more comprehensive testing of SDN controllers, as well as developing testing tools with appropriate performance metrics for SDNs.

Acknowledgements We would like to thank the anonymous reviewers for their insightful comments and constructive suggestions, which significantly helped us to improve the quality of this work. Additionally, we would like to thank Prof. Hasan Sözer for his valuable comments, which help us improve the quality of this work.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Latah M, Toker L (2019) Artificial intelligence enabled software-defined networking: a comprehensive overview. IET Netw 8(2):79–99
2. Laissaoui C, Idboufker N, Elassali R, El Baamrani K (2015) A measurement of the response times of various open flow/sdn controllers with cbench. In: IEEE/ACS 12th international conference of computer systems and applications (AICCSA), IEEE, pp 1–2
3. Khattak ZK, Awais M, Iqbal A (2014) Performance evaluation of opendaylight sdn controller. In: 2014 20th IEEE international conference on parallel and distributed systems (ICPADS), IEEE, pp 671–676
4. Fancy C, Pushpalatha M (2017) Performance evaluation of sdn controllers pox and Foodlight in mininet emulation environment. In: International conference on intelligent sustainable systems (ICISS), IEEE, pp 695–699
5. Rastogi A, Bais A (2016) Comparative analysis of software defined networking (sdn) controllers in terms of traffic handling capabilities. In: 19th international multi-topic conference (INMIC), IEEE, pp 1–6
6. Darianian M, Williamson C, Haque I (2017) Experimental evaluation of two open flow controllers. In: IEEE 25th international conference on network protocols (ICNP), IEEE, pp 1–6
7. Mamushiane L, Lysko A, Dlamini S (2018) A comparative evaluation of the performance of popular sdn controllers. In: 2018 wireless days (WD), IEEE, pp 54–59
8. Stancu AL, Halunga S, Vulpe A, Suciu G, Fratu O, Popovici EC (2015) A comparison between several software defined networking controllers. In: 12th international conference on telecommunication in modern satellite. Cable and broadcasting services (TELSIKS), IEEE, pp 223–226
9. Rowshanrad S, Abdi V, Keshtgari M (2016) Performance evaluation of SDN controllers: Floodlight and OpenDaylight. IIUM Eng J 17(2):47–57
10. Shalimov A, Zuikov D, Zimarina D, Pashkov V, Smeliansky R, (2013) Advanced study of SDN/OpenFlow controllers. In: Proceedings of the 9th central & eastern european software engineering conference in Russia, ACM, pp 1–6
11. Tootoonchian A, Gorbunov S, Ganjali Y, Casado M, Sherwood R (2012) On controller performance in software-defined networks. In: 2nd USENIX workshop on hot topics in management of internet, cloud, and enterprise networks and services, USENIX Association, pp 1–6
12. Zhao Y, Iannone L, Riguidel M (2015) On the performance of SDN controllers: a reality check. In: IEEE conference on network function virtualization and software defined network (NFV-SDN), IEEE, pp 79–85
13. Bholebawa IZ, Dalal UD (2018) Performance analysis of SDN/OpenFlow controllers: Pox versus floodlight. Wirel Pers Commun 98(2):1679–1699