Security and Performance Comparison of ONOS and ODL Controllers

Stefano Secci, Alessio Diamanti, José Manuel Sanchez Vilchez, Mamadou Tahirou Bah, Pedra Vizzarreta, Carmen Mas Machuca, Sandra Scott-Hayward, David Smith

Security and Performance Comparison of ONOS and ODL controllers

Stefano Secci [1], Alessio Diamanti [1,2], José Manuel Vilchez Sanchez [2], Mamadou Tahirou Bah [6], Petra Vizarreta [3], Carmen Mas Machuca [3], Sandra Scott-Hayward [4], Dylan Smith [5]

[1] Cnam, France. [2] Orange, France. [3] TUM, Germany. [4]QUB, UK. [5]CIT, Ireland. [6]LIP6, France

Corresponding author: Stefano Secci (stefano.secci@cnam.fr)

Date: Sept. 10, 2019

TABLE OF CONTENTS

Introduction

This report is the result of an effort of the ONF security and performance analysis brigade to compare the two most widely used SDN controllers, ONOS (Open Network Operating System) and ODL (OpenDayLight).

It is worth remarking that the comparison is not meant to be extensive nor exhaustive. The reported comparison methodology and results focus on some specific aspects of the controller system, while not covering the many other aspects that may be interesting to analyze. This report may be updated in the coming years, to compare novel versions as well as other controllers that may be developed. If you are willing to perform additional performance and security analysis tests for inclusion in future reports, please contact us.

Editorial note: the report is not self-contained as is a scientific publication, i.e., a prior technical knowledge of the various technologies are required to fully understand the content of the report.

Citation: *S. Secci, A. Diamanti, JM. Sanchez, MT. Bah, P. Vizarreta, C. Mas Machuca, S. Scott-Hayward, D. Smith, "Security and Performance Comparison of ONOS and ODL Controllers", Informational Report, Open Networking Foundation, Sept. 2019.*

1. Performance Analysis

The performance analysis focuses on path restoration performance and software reliability.

1.1. Path restoration behavior

Contributors: Alessio Diamanti (Orange/Cnam), José Manuel Sanchez Vllchez (Orange), Mamadou Tahirou Bah (LIP6)

In the following, we report on control-plane reactivity to topology changes and discovery events, comparing ONOS and ODL behaviors.

Topology update reactivity

SDN controllers are expected to maintain an updated view of the network in a semi-real time fashion in order to let applications work with a consistent view. Generally speaking, the topology update is implemented following an event driven pattern logic. When some specific packets sent by SDN switches are received or some expected packets are not received by the controller, an event is raised in the controller's core. This event is received and held by subscribed listeners, which will in turn solicit topological representation changes in a database, eventually distributed. We focus in particular on what happens in ONOS and ODL controllers when a OFPT_PORT_STATUS packet is received from a switch to notify a change in a port's status after a link disruption event or after a link is established/re-established.

Supposing that a path computation application is activated in the controller, the controller has to react to this change in the topology, eventually installing new flows to circumvent the disruption and finally ensure communications.

In this section, we evaluate how fast and promptly ONOS and ODL controllers perform these update actions when reacting to a topological change. To perform this comparison, we analyze a very basic test case (as in Figure 1): two hosts, H1 and H2, connected by a single path composed of 6 links and 5 SDN switches (OVS switches), and exchanging UDP packets through an Iperf [16] session. We use Quali version for ONOS and Oxygen version for ODL.

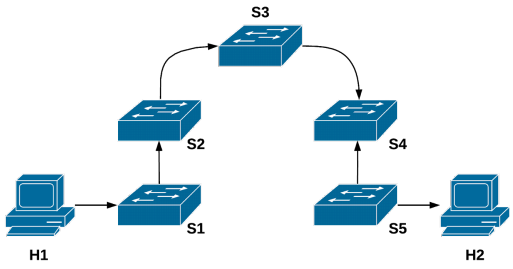


Figure 1: Test case network topology

During tests, the configuration shown in Figure 1 was deployed through a developed python module that simulates complex network topologies with redundant links and alternative paths among hosts. The simulator also provides a fault injection module that is capable of injecting faults and degradations on each of the simulated network elements. To ensure connectivity between the hosts, “org.onosproject.fwd” and “org.onosproject.openflow” apps were activated in ONOS and “odl-l2switch-all” was activated in ODL. In order to gather sufficient data, we iterate the test case 1400 times, cleaning both topology and controller state between iterations.

Supposing the i-th iteration starts at time t_0 when the Iperf session is started, at time $t_0 + \tau = T_{start}$ a failure in the link between switches S2 and S3 is introduced and finally after ΔT seconds the link is restored. To monitor the traffic flowing along the path, at t_0 a tshark [17] capture is started on the link between S3 and S4. In particular, from this capture, it is possible to extract the time at which the first UDP packet appears on link S3-S4, defined as T_{first}. Knowing that the link is restored at time $t_0 + \tau + \Delta T = T_{stop}$, the controller’s reaction time can be computed as $T_{react} = T_{first} - T_{stop}$. Let us note that Iperf is configured in UDP mode in order to remove all synchronization overhead specific to TCP that would have biased the reaction time. Furthermore, in the topology, no alternative path from H1 to H2 was created so that the controller will not perform actions other than those described.

Figure 2 reports the empirical probability distribution function (PDF) of the reaction time (T_{react}) for the 1400 tests and for both ONOS and ODL.

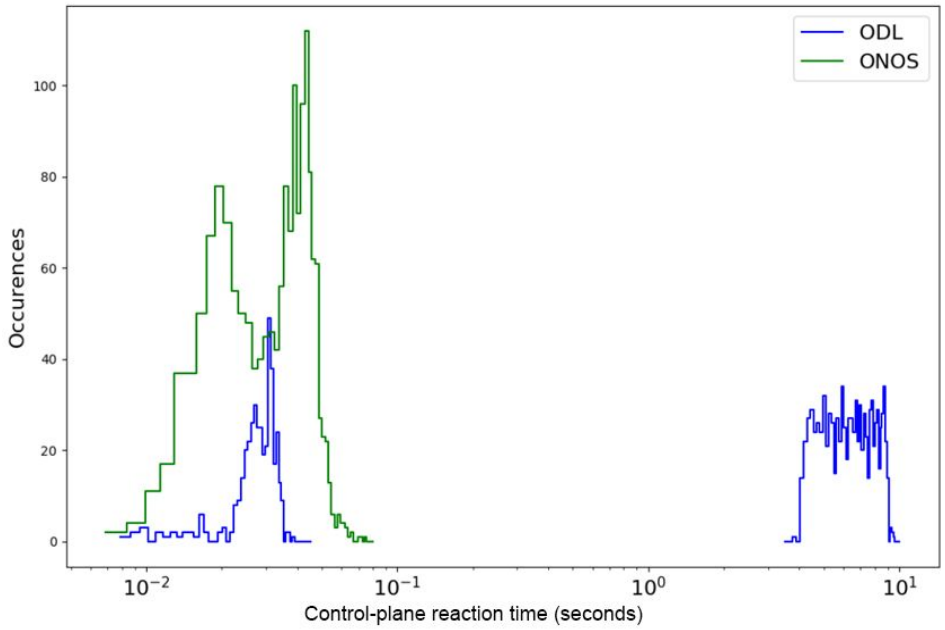
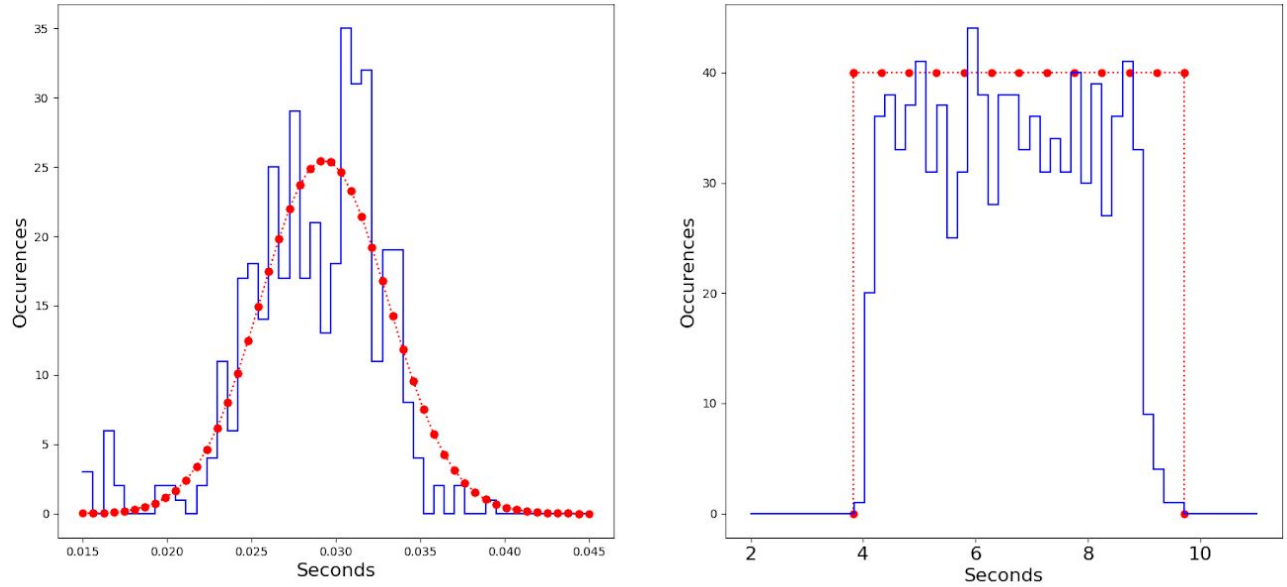


Figure 2: Distribution of reaction time (T_{react}) for ONOS and ODL controllers.

As depicted in Figure 2, for a number of tests both controllers react in a similar way. However, ODL shows two different modes: in 30.1% of the tests, the reaction times fall into the interval [0, 0.04], while in the remaining 69.9% of cases values are in [3, 10]. Thus, in Figure 3, we represent the dynamics of the reaction times in two split PDF plots for the two modes. Equivalent results for ONOS are in Figure 4, in a single plot, for the sake of clarity.



(a) [0,0.04] seconds (b) [3,10] seconds

Figure 3: Distribution of the reaction times for ODL at two distinct intervals.

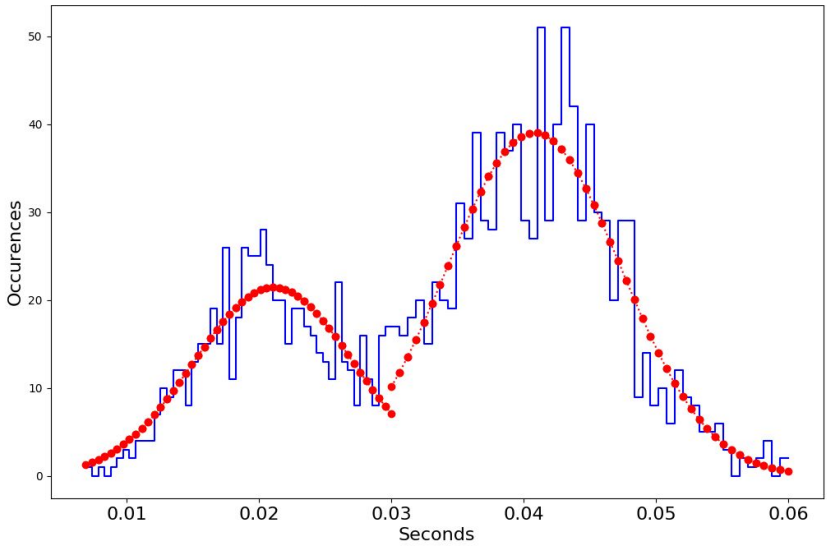


Figure 4: Distribution of the reaction times for ONOS

ONOS appears to be significantly more stable than ODL. Note that ONOS also shows two separate modes, similar to ODL, yet they are much closer than with ODL; while the

distance between the modes is approximately 6 seconds for ODL, it is approximately 20 ms for ONOS with no occurrences gap.

ONOS is also much faster in reacting to topology event updates, with a median reaction time of 36 ms; that is two orders of magnitude less than ODL that has a median of 5.45 seconds, as shown in the boxplot statistics in Figure 5.

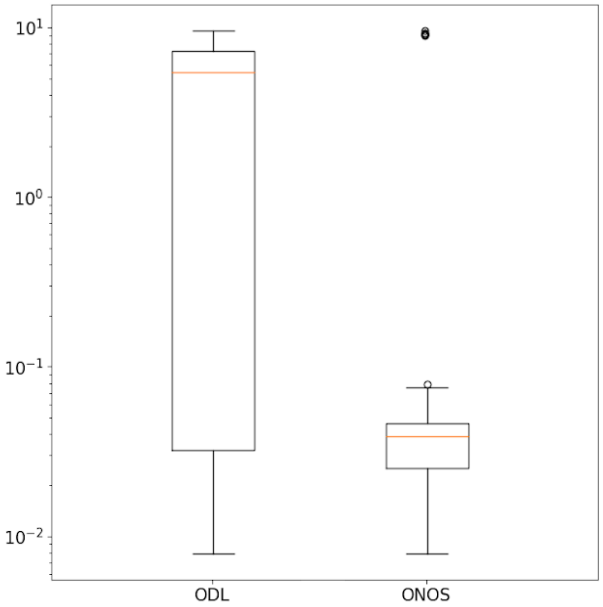


Figure 5: Boxplot statistics of the reaction times. A boxplot shows the minimum, first quartile, median in red, third quartile and maximum values.

To better understand the reasons for the detected unstable ODL behavior, we tried to capture the variability across tests characterizing how frequently the reaction time switches from the first mode (Figure 3a) to the second one (Figure 3b) in subsequent tests. To do so, we use a metric that is incremented by one each time a switch from the first mode to the second one is detected, when considering i-th and i+1-th tests. The result is shown in Figure 6. A controller whose reaction times in subsequent tests would flip among the modes, i.e., fall in the other mode each time, would have produced the first quadrant bisector line in such a plot. However, it is quite close to the bisector, which means that ODL is very unstable as it is reacting in very different ways across subsequent tests. In order to search for possible correlations, we also computed the empirical probability that at the i-th test the reaction time is in the first [second] mode while in the subsequent i+1-th test the reaction time falls in the second [first] mode. We found no difference, with an empirical probability to switch from the first to the second of 0.2075, while the reverse is 0.2083 (very close to the former). Summing these probabilities, we obtain a probability of 0.4158 to switch from one class to another, which confirms the unpredictability of the ODL controller.

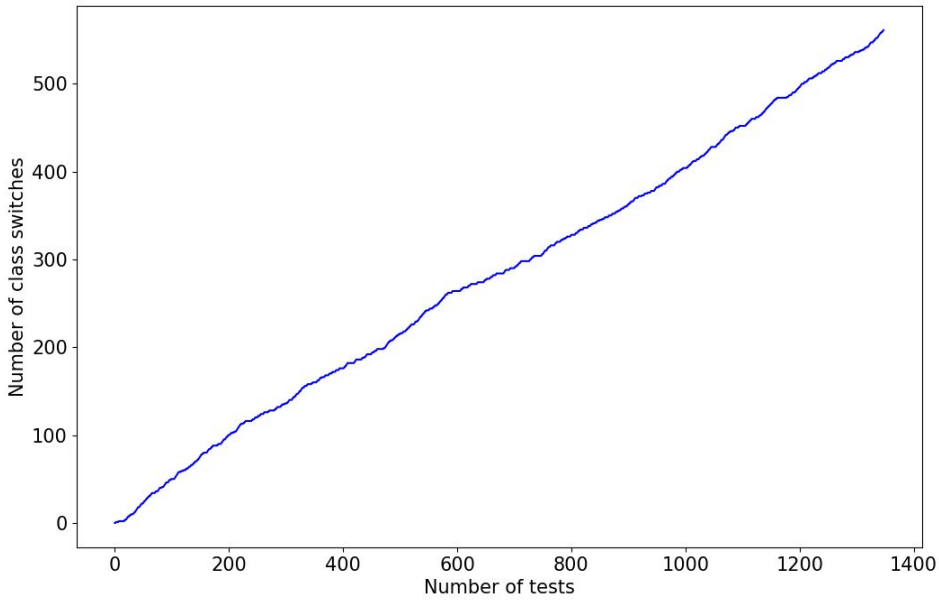


Figure 6: Number of mode switches as a function of the number of tests -- ODL

An aspect that remains unclear from the test is the origin of the large gap between the two working modes in ODL. Further work might inspect openflow messages exchanged between the controller and the switches to identify the ODL core mechanism triggered by those messages.

Topology discovery

We analyze the amount of control traffic required for ONOS and ODL to discover and update the topology over time. Both controllers use Link Layer Discovery Protocol (LLDP) to infer links connecting switches. Basically, the controller sends a PACKET_OUT message to each switch containing as many LLDP frames as active ports in the corresponding switch. Each switch then sends out LLDP frames in designated ports, forwarding to the controller, through a PACKET_IN message, each LLDP frame it receives. Consequently, the controller can infer links binding the port where the packet was sent (inside LLDP frame) and the port where the packet was received (a field in the PACKET_IN packet). This procedure is repeated periodically in order to maintain an up-to-date topology; every 3 seconds in ONOS and every 5 seconds in ODL, by default.

To test the implementation of LLDP in ONOS and ODL, we simply record for a given period all PACKET_IN and PACKET_OUT messages exchanged since the first topology discovery using a topology such as that in Figure 1. In order to fairly estimate the amount of traffic, we manually set the LLDP cycle in both controllers to 5 seconds.

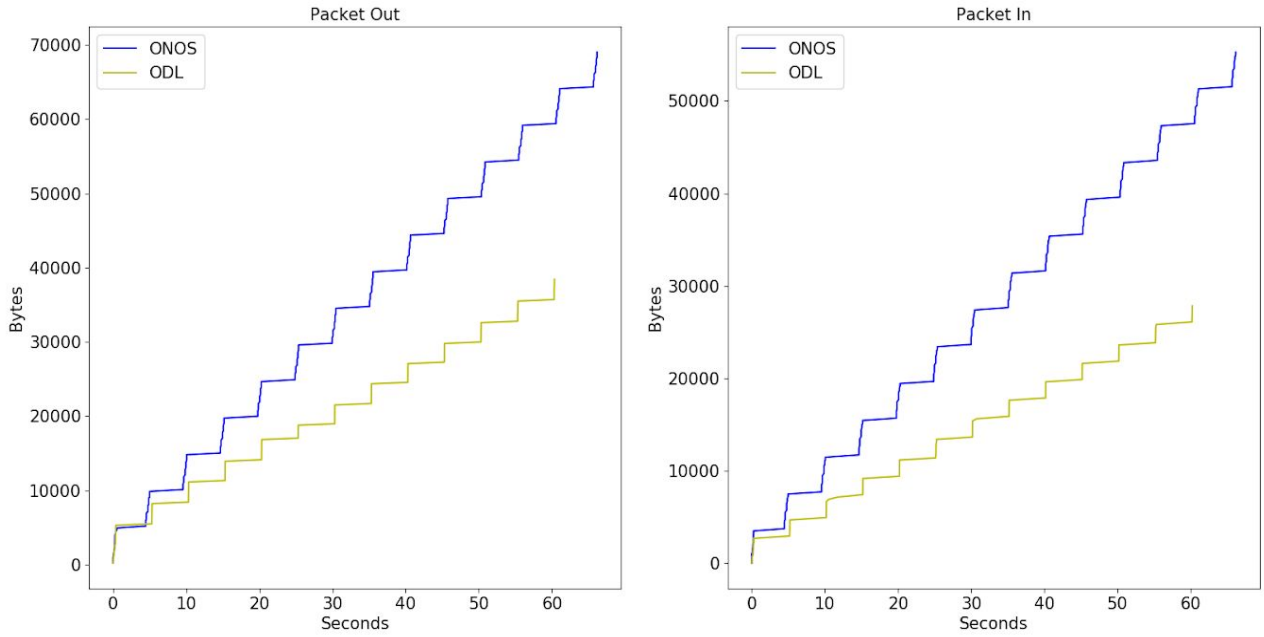


Figure 7: Topology discovery volume for both PACKET_IN and PACKET_OUT messages

Figure 7 shows the obtained results. ONOS produces a larger amount of control traffic in terms of PACKET_OUT and thus PACKET_IN with respect to ODL, as each PACKET_OUT will result in a PACKET_IN if there is an active link on the related port. We attribute this difference in the amount of exchanged packets to an optimized implementation of LLDP in ODL, e.g. OFDPV2 [17].

1.2. Reliability growth performance

Contributors: Petra Vizarreta and Carmen Mas Machuca (TUM)

There is strong empirical evidence that network control plane elements contribute to a significant share of outages that impact on customers. An example from Google B4 SDN-based network shows that software bugs caused more than 30% of customer impacting incidents [1]. SDN controllers are complex, as they not only take over the control plane functions of forwarding devices, but also implement many deployment related functions, e.g., security, virtualization interfaces. Defects in such complex software systems are inevitable.

Software reliability growth models allow an estimation of the risk of critical failures. Their application within a quantitative framework allows network operators and controller software developers to determine when a controller release is ready to be deployed in an operational environment. The outline of the framework is presented in Figure 8:

- First, data is collected from software repositories that contain useful data on code change dynamics, as well as the reported issues and patches (Git, Gerrit, Jira). The trends are then analyzed to assess whether the reliability growth is present and which model would be the most suitable.
- A second step is to find the best reliability model to describe stochastic behaviour of bug manifestation and bug removal processes. After the best model is found and parameterized, it can be used to estimate the software reliability Key Performance Indicators (KPIs), such as expected residual bug content, expected failure rate and interval reliability (i.e., probability of outages in the support period).

Finally, these KPIs can be used to guide the management decisions, such as postponing an official software release or adoption rate. Next, we compare the reliability growth of the latest stable releases of ONOS and ODL.

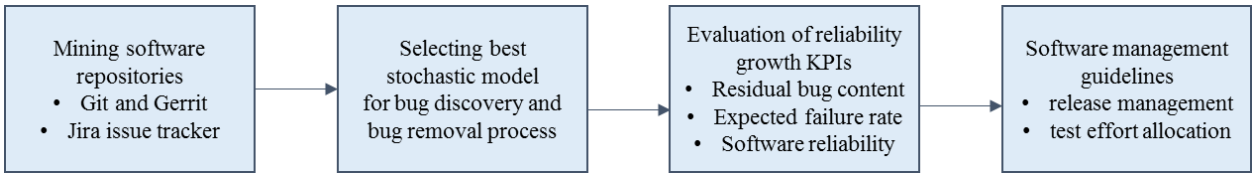


Figure 8: Workflow for evaluation of reliability growth and software maturity (adapted from [3])

Data collection from software repositories

Software repositories contain valuable data on code changes and issues that have been documented during the development and deployment phases of the project. The data considered in this study has been retrieved on 03.09.2018 from ONOS code version control system and ODL issue tracker, which are publicly available. The high level

summary of the repositories and static software reliability metric that can be derived from such data are presented in Table 1.

Table 1: Summary of code version control (Git and Gerrit) and bug (Jira issue tracker) repositories

Project	*Bugs*	*Commits*	*LOC*	*Density [bugs/kLOC]*
ONOS	*2.072*	*13.254*	*852.570*	*2,43*
ODL	*9.060*	*98.084*	*3.920.556*	*2,31*

Comparing the failure dynamics over time, as presented in Figures 9 and 10, short term and long term trends can be observed. Both controllers show peaks in the number of reported bugs shortly before the formal release dates, which are indicated with dashed lines in the figures. Note that the Service Releases (SR) are not shown in the figures.

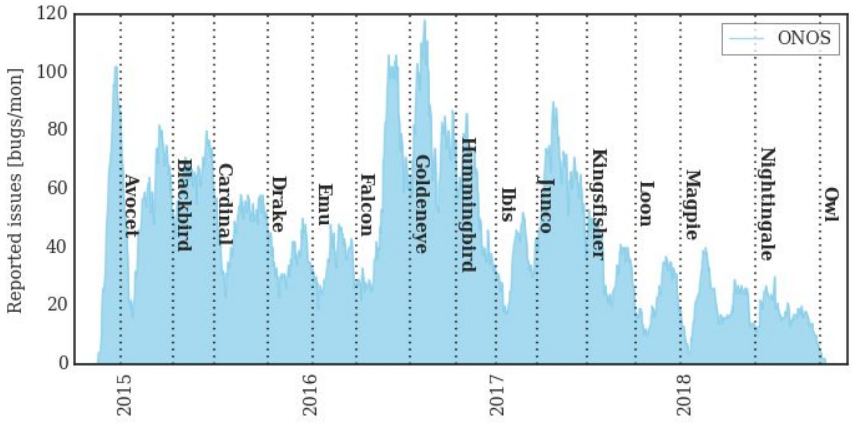


Figure 9: Number of reported issues over time (ONOS)

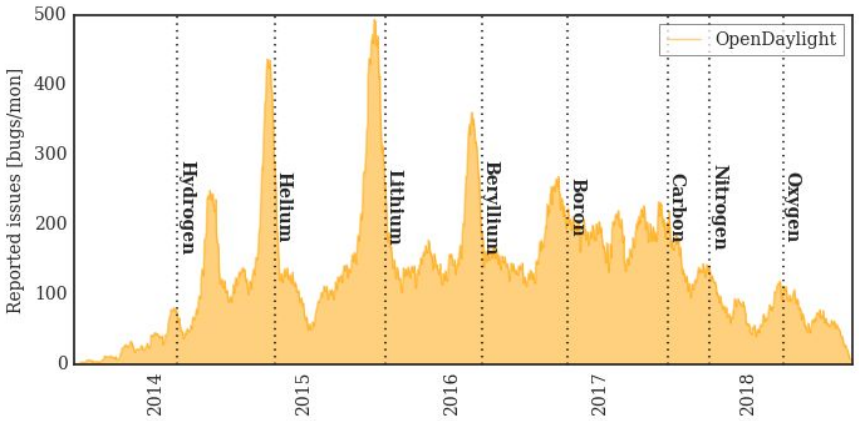


Figure 10: Number of reported issues over time (ODL)

Model selection: Selecting best Software Reliability Growth Model (SRGM)

Software reliability growth models (SRGM) can model the trend of changes in bug manifestation rate depending on the proximity to the official release date.

The bug manifestation rate of individual bugs follows an arbitrary distribution, depending on the reliability model. The overall bug manifestation rate depends on the total bug content in the code. During the testing and early deployment phase of a release, bugs are removed from the code, and hence, the bug manifestation rate eventually decreases, leading to a reliability growth [2]. The most suitable models for SDN controllers are presented in [3,4]. Focusing on the expected cumulative number of the bugs over time, three SRGMs are commonly considered: (i) Generalized Goel Okumoto (GGO), (ii) Gompertz Model (GOMP), and (iii) the Ohba's Inflection S-Shape (ISS) model. Avoiding mathematical details, the three models have different fitting shapes. Their behavior is observable when applying them against the two latest releases of ONOS and ODL is presented in Figures 11-14 (the Mean Square Error, MSE, is given within parenthesis). The relative time is calculated w.r.t. the start of integration testing, i.e., time when the only changes in the code are due to bug fixes. We estimate this time as an offset time based on the release cycles.

Note that we use "affected release" field to separate the bugs per release, which is not reported for all issues. Hence, the numbers provided in this report are for illustration purposes only. An alternative would be to use time based separation, but this would not be entirely accurate since for some bugs there is a time overlap between releases and some bugs are only related to the new features.

Two latest stable releases of ONOS: Magpie and Nightingale

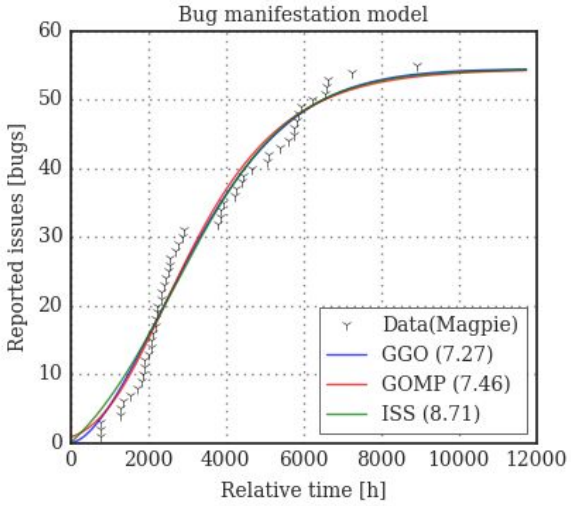


Figure 11: Model selection for Magpie (ONOS v1.12)

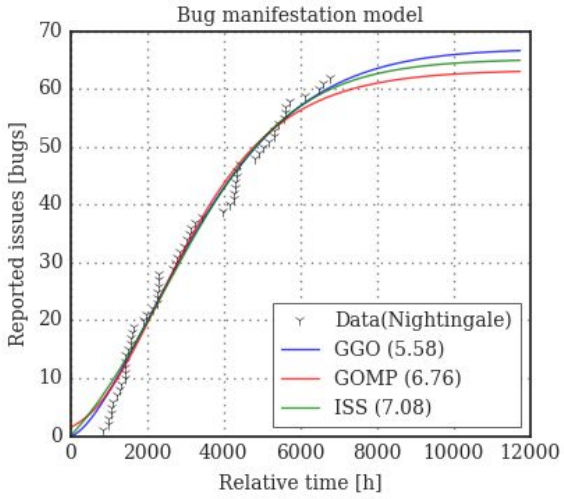


Figure 12: Model selection for Nightingale (ONOS v1.13)

Two latest stable releases of OpenDaylight: Nitrogen and Oxygen

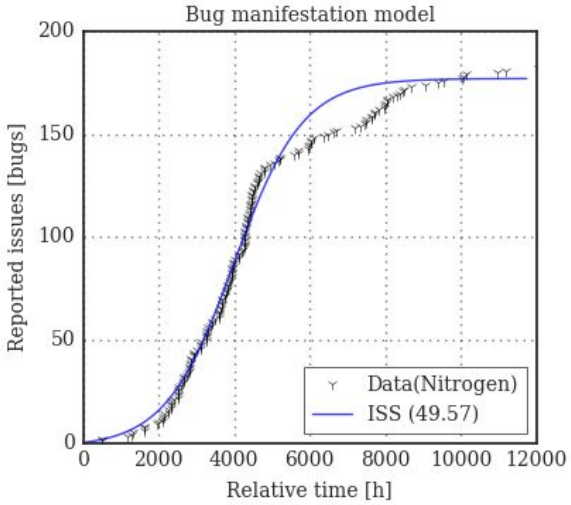


Figure 13: Model selection for Nitrogen release

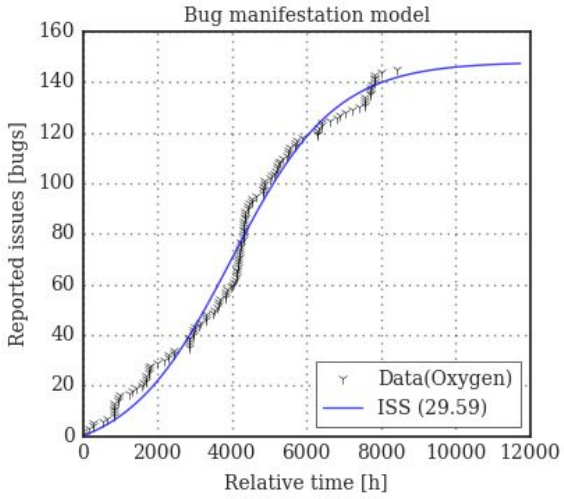


Figure 14: Model selection for Oxygen release

It can be observed in Figures 11-14 that all three models show good fit for the two latest stable ONOS releases, with Generalized Goel Okumoto (GGO) showing slightly lower MSE than the other two. The best fitting model for the last two ODL releases is ISS. It is interesting to observe that all of these latest releases of ONOS and ODL have very similar shape parameters (related to the time dependence, not shown in the figures).

Predicting the controller software reliability with SRGM

Once the best model is found and parameterized, it can be used to quantify and forecast different software reliability KPIs. Some KPIs such as residual bug content, software failure rate and interval reliability (probability of failures in a given time period) can be estimated directly from the model [3].

As an example for ONOS, in Figure 15 we show the expected residual bug content and the expected software failure rate, for the GGO model for Magpie and Nightingale releases. We observe that the number of the remaining bugs in the code on the day of the official release (indicated with the vertical line in the figure) was high: for Magpie and for Nightingale. The expected failure rates are similar for both releases, which is approximately days between the two successive bug manifestations.

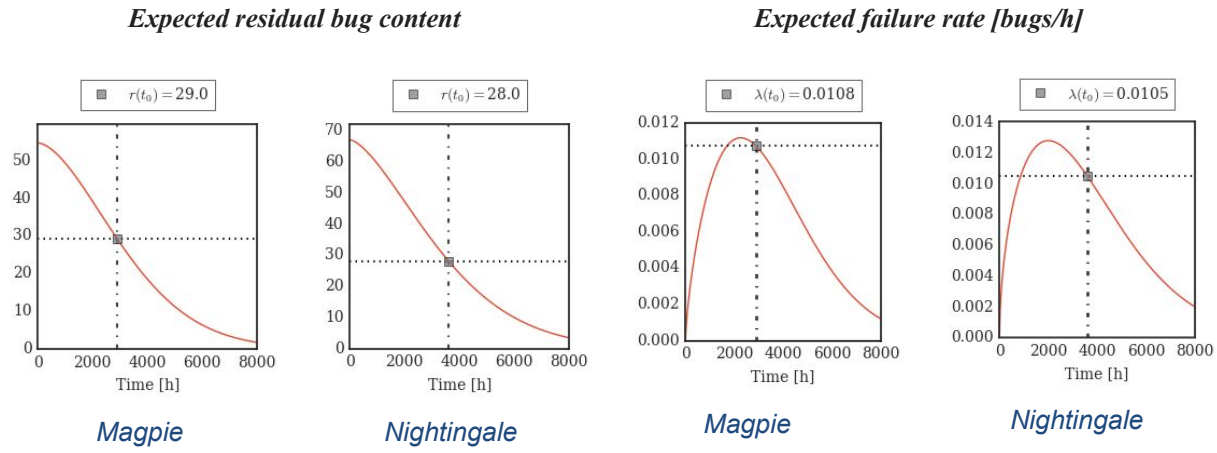


Figure 15: Predicting software reliability KPIs for ONOS Magpie and Nottingale ONOS releases.

The presented reliability growth models and forecasting of reliability growth can be used to guide management decisions, such as postponing an official software release or adoption rate. It has to be noted, however, that the results presented rely on the data reported in software repositories, mainly Jira issue tracker. The accuracy and completeness of the repository is not guaranteed, hence, the presented numbers might under- or over-estimate the risk of software failures.

2. Security Analysis

2.1. Comparison of ODL and ONOS Security

Contributor: Sandra Scott-Hayward (QUB)

In this section, we present a summary of the security design, development and support provided by ODL and ONOS. A full comparative analysis considering the evolution of secure SDN Controller design has been presented in [5].

2.1.1. Security Support

Both ONOS and ODL provide information on their web pages regarding their security support.

In the case of ONOS, this information can be accessed by searching the wiki for “security” or by navigating through the page tree. Details are provided regarding the Security response team, the procedure to report a security issue, and a link to the security advisories page. These pages are updated infrequently.

In the case of ODL, similar information is provided but it is more directly accessible via a drop-down menu on the ODL home page. A link to the vulnerability management process is also provided [6].

Since 2018, management of the security advisories for both ONOS and ODL has transferred to MITRE for generation of CVEs [7,8]. ODL provides this information in [6]. However, no information regarding this process is provided for ONOS.

The number of CVEs generated to date is shown in Table 2. Further information on the types of CVEs reported for each controller is provided in Section 2.2.

Table 2: ONOS and ODL CVEs per annum

	ONOS	ODL
2014	0	2
2015	1	9
2016	0	2
2017	6	7
2018	5	2

2.1.2. Security-specific Projects/Applications

There have been a number of security-related projects and applications developed by contributors to ONOS and ODL.

In the case of ONOS, contributions in 2015/2016 include the *Security-Mode ONOS (SM-ONOS)* [9] and *Access Control based on DHCP* [10] projects, and *Access Control Lists (ACL)* and *AAA* applications. As described in [5], these offer some functionality but are either basic or not part of core ONOS.

There are a number of more recent security-related projects/applications. *ARTEMIS* is an Automated System against BGP Prefix Hijacking [11], which leverages ONOS. *VPLS* is a virtual private LAN service [12] and *Policy Framework for ONOS* provides a network policy framework for ONOS [13]. Of these, the policy framework for ONOS provides a key contribution to addressing one of the core security concerns of SDN controllers; that of policy conflict resolution [14]. When multiple applications/modules submit policies to direct the control of the network, it is critical to ensure that there are no conflicts between these policies being implemented in the network.

Security-related projects in ODL include *Defense4All, Secure Network Bootstrapping Interface, Authentication, Authorization and Accounting (AAA), Unified Secure Channel, Controller Shield,* and *Cardinal - ODL Monitoring as a Service*. Similar to ONOS, these projects vary in maturity with some with limited functionality or no longer maintained while others such as *AAA* [11] have become managed projects, emphasising their importance to the fundamental operation of ODL. Full details of how application developers can configure and use *AAA* are provided.

2.1.3. Security-focused design

The original objective of the ONOS controller was to offer high availability. Availability is, of course, an element in the CIA security triad, along with Confidentiality and Integrity. To support high availability, ONOS offers a clustered configuration in which multiple controller instances coordinate to provide a fault-tolerant and resilient, distributed SDN operating system. In early versions of ONOS, the clustered configuration used the Raft consensus protocol. This implementation led to various issues linked to memory consumption and state information management. Specifically, the cluster could not maintain control of the network when reduced to two instances. Since ONOS v1.14 (Owl), a new cluster configuration architecture has been introduced. This physically decouples cluster management, service discovery, and persistent data storage from the

ONOS nodes into a separate Atomix cluster [16]. With this new implementation, resilient high availability is provided; the ONOS cluster can tolerate the failure of all but one node.

The emphasis on security for design and deployment is clear from ODL. In addition to the embedding of fundamental security functionality such as AAA and the clear guidance on vulnerability reporting, there is a further aspect that encourages a security-focused design by ODL project contributors. This is the requirement for each project within ODL to report on security considerations linked to functionality or feature updates for a given release. This highlights, for example, if the project uses external interfaces and how they will be secured. In general, the developers, identify the requirements to secure the interface or provide a link to the relevant security configuration information. This is further supported by the security considerations page on the ODL website, which describes the security issues that might affect ODL and lists specific recommendations to mitigate security risks. This page also features a "Report Issue" button, which links directly to the jira.

2.2. Comparison of ODL and ONOS Vulnerabilities

Contributor: Dylan Smyth (CIT)

Both ODL and ONOS have been subject to vulnerability research, with ODL receiving its first documented security vulnerability in 2014. Since then, several other vulnerabilities have been reported in both controllers. This section will provide an overview of the recorded vulnerabilities for both ODL and ONOS, and assess whether these vulnerabilities can provide an indication of the overall security level of the controllers.

OpenDaylight (ODL)

As detailed in Table 2, ODL has a total of 22 vulnerabilities recorded, 16 of which relate to core ODL code and applications [8][17]. Six security advisories have also been issued regarding bugs and vulnerabilities in components used by ODL but maintained by third parties[1].

Denial of Service (DoS) is a common trend among vulnerabilities reported for ODL. A number of these are related to excessive resource consumption. *CVE-2017-1000359* documents a flaw, which allows an attacker to cause significant resource consumption due to access to certain ports used by the XSQL service not being restricted.

[1] CVE-2015-3414, CVE-2015-3416, CVE-2015-4000, CVE-2015-7501, CVE-2016-2183, and CVE-2016-4970

CVE-2017-1000361 also results in resource consumption and can be triggered via a port-status message from an SDN switch. A bug, *CVE-2017-1000411*, causes expired flows to remain in the Config Datastore resulting in memory consumption and eventually controller shutdown. Users can trigger a bug, which prevents flows from being added to a switch by repeatedly adding the same flow via the REST API, documented as *CVE-2017-1000358*. The aforementioned XSQL service contained another vulnerability, *CVE-2017-1000360*, where an attacker can cause a Null Pointer Exception by sending data to that service via unrestricted ports. *CVE-2018-1078* causes flows to remain in switches when they should be removed, due to a bug in node reconciliation, which could potentially be used in a TCAM Exhaustion attack. A bug in the layer 2 switch application, *CVE-2017-1000357*, causes the switch to reject packets from the controller which prevents correct operation of the network.

Access control and authentication components of ODL have also been subject to security bugs. One such bug, *CVE-2017-1000406*, allowed old passwords to be used after a password change due to the old password being cached. Another similar bug was found where any username and password combination could be used to access the controller, documented as *CVE-2015-1778*. The odl-mdsal-apidocs feature was found to contain a bug, *CVE-2015-1857*, which allowed sensitive information to be read due to lack of proper security restrictions. The Netconf TCP service contained a bug, which allowed arbitrary files to be read. This was caused by an XML External Entity (XXE) vulnerability and is documented as *CVE-2014-5035*.

Vulnerabilities related to the network topology have also been documented, which refer to known attacks against SDN and affect a large percentage of controllers. *CVE-2015-1612* and *CVE-2015-1611* refer to the Link Fabrication Attack (LFA) where a Link Layer Discovery Protocol (LLDP) message can be crafted or captured and replayed in order to cause the controller to believe a link exists between two switches when that link does not physically exist. *CVE-2015-1610* refers to the Host Location Hijacking Attack, where an attacker can send a packet into the network with a spoofed source address causing the controller to believe that a host has changed location in the network.

The ODL defense4all application contained a serious vulnerability, *CVE-2014-8149*, which allowed data to be written to arbitrary files. The SDN Interface application, now deprecated, contained an SQL injection vulnerability, documented as *CVE-2017-1000411*.

ONOS

As detailed in Table 2, ONOS has a total of 12 public vulnerabilities recorded [7][18]. Similar to ODL, the bugs affect a variety of components.

DoS attacks make up just less than half of the known vulnerabilities. *CVE-2017-13763* describes an issue with a clustering component, where forged packets can be sent to a controller causing an increase in memory usage. The IP forwarding service could be crashed by sending jumbo Ethernet frames. This bug is recorded as *CVE-2015-7516*. Similarly, *CVE-2018-1000615* documents a bug where the OVSDB service can be crashed. The vulnerability recorded as *CVE-2017-1000079* causes certain core features of ONOS to stop working correctly.

ONOS has 2 XXE vulnerabilities associated with it. *CVE-2018-1000616* describes an XXE vulnerability in the XML Configuration Parser, while *CVE-2018-1000614* describes a similar issue in the Netconf Alarm Translator service.

Correct authentication has also been a problem for ONOS. *CVE-2017-1000080* describes a vulnerability where the websocket used by the web user interface (UI) can be accessed without any authentication. This then gives an attacker access to the information normally seen on the ONOS web UI (e.g. topology information). A serious bug, documented as *CVE-2017-1000081*, allows the unauthenticated upload of applications, allowing anyone to execute arbitrary code on the controller. Further issues have been found in the web interface. *CVE-2017-13762* describes a bug whereby an attacker could exploit an XSS vulnerability in the web UI by passing HTML and JavaScript from a switch connected to the controller. The same XSS vulnerability can be exploited via the REST API, as documented in *CVE-2017-1000078*.

A race condition was found and documented as *CVE-2018-12691,* which enables ACL rules to be bypassed. Topology issues such as the Host Location Hijacking attack and the LFA have also been an issue for ONOS. However, these are not documented with CVE IDs. Recent versions of ONOS contain protections against the generation and replay-type LFA.

Discussion

Both ODL and ONOS have a number of vulnerabilities associated with them. In both cases, the vulnerabilities are diverse and affect a wide variety of components.

A number of ODL vulnerabilities are caused by lack of authentication and restrictions, and allowing untrusted data to reach controller components. Ensuring that users must authenticate before controller interaction, and sanitizing and checking all user supplied data is key to preventing vulnerabilities such as those presented in this Section.

ONOS has had similar issues with lack of restriction and authentication, specifically in components related to the web UI. ODL has no documented vulnerabilities related to the web UI. However, the web UI provided by ONOS is more feature-rich and complex, so that more bugs would be expected.

To summarize this section, considering the identified features of a secure, robust, and resilient SDN controller [14], there has been an increased focus on security within both ONOS and ODL controller communities over the past six years. In the case of ONOS, the improvements in clustering functionality and the introduction of a policy framework are encouraging developments. However, the consistent consideration of security across project development in ODL, the provision of core AAA functionality, and the emphasis on security considerations for deployment and vulnerability management highlight the commitment of ODL to security. As discussed in Section 2.2, a range of vulnerabilities have been identified in both ONOS and ODL controllers. There are some similarities with respect to exposure to DoS and authentication vulnerabilities, which have now been addressed. Of course, from a software development perspective, as the deployment of both systems increases, the emergence of further vulnerabilities will be a strong indicator of the system security.

Summary

In this report, we have compared both quantitatively and qualitatively, though non-exhaustively, key characteristics of the ONOS and ODL controllers. We focused the performance analysis in particular on control-plane reactivity to network topology events and in software reliability growth models that could help towards prudent open source project management. In terms of security aspects, we provided a qualitative comparison of the two controllers, and analysed their recorded vulnerabilities to date.

This report is the result of unfunded activities by the involved parties - results were also presented during the third ONF sec&perf brigade workshop held in Paris, June 17, 2019, at TMA 2019. Companion slides to this report are available at https://wiki.onosproject.org/pages/viewpage.action?pageId=12422167.

Citation: *S. Secci, A. Diamanti, JM. Sanchez, MT. Bah, P. Vizzarreta, C. Mas Machuca, S. Scott-Hayward, D. Smith, "Security and Performance Comparison of ONOS and ODL Controllers", Informational Report, Open Networking Foundation, Sept. 2019.*

References

[1] R. Govindan, I. Minei, M. Kallahalla, B. Koley, and A. Vahdat, “Evolve or die: High-availability design principles drawn from Google’s network infrastructure,” in Proceedings of ACM SIGCOMM Conference. ACM, 2016, pp. 58–72.
[2] M. R. Lyu et al., Handbook of software reliability engineering. IEEE computer society press CA, 1996.
[3] Vizarreta, Petra; Trivedi, Kishor; Helvik, Bjarne; Heegaard, Poul; Blenk, Andreas; Kellerer, Wolfgang; Mas Machuca, Carmen: Assessing the Software Maturity of SDN Controllers Using Software Reliability Growth Models. Transactions on Network and Service Management, 2018
[4] Vizarreta, Petra; Ermin Sakic; Kellerer, Wolfgang; Mas Machuca, Carmen: Mining Software Repositories for Predictive Modelling of Defects in SDN Controller. Submitted to: IFIP/IEEE International Symposium on Integrated Network Management (IFIP IM), 2019
[5] Scott-Hayward, Sandra. "Trailing the Snail: SDN Controller Security Evolution." *arXiv preprint arXiv:1711.08406* (2017).
[6] OpenDaylight Vulnerability Management Process [Online] Available: https://wiki.opendaylight.org/view/Security:Vulnerability_Management#Risk_Assessment
[7] ONOS CVE list [Online] Available: https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=ONOS
[8] ODL CVE list [Online] Available: https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=OpenDaylight
[9] “Security-Mode ONOS.” [Online]. Available: https://wiki.onosproject.org/display/ONOS/Security-Mode+ONOS
[10] “Access Control Based on DHCP.” [Online]. Available: https://wiki.onosproject.org/display/ONOS/Access+Control+Based+on+DHCP
[11] “ARTEMIS: an Automated System against BGP Prefix Hijacking.” [Online]. Available: https://wiki.onosproject.org/display/ONOS/ARTEMIS%3A+an+Automated+System+against+BGP+Prefix+Hijacking
[12] “Virtual Private LAN Service - VPLS” [Online] Available: https://wiki.onosproject.org/display/ONOS/Virtual+Private+LAN+Service+-+VPLS
[13] “Policy framework for ONOS” [Online] Available: https://wiki.onosproject.org/display/ONOS/POLICY+FRAMEWORK+FOR+ONOS
[14] Scott-Hayward, Sandra. "Design and deployment of secure, robust, and resilient SDN Controllers." In *Proceedings of the 2015 1st IEEE conference on network Softwarization (NetSoft)*, pp. 1-5. IEEE, 2015.
[15] “Authentication, Authorization, and Accounting (AAA) Services” [Online] Available: https://docs.opendaylight.org/projects/aaa/en/latest/dev-guide.html
[16] “Cluster Configuration in Owl (1.14)” [Online] Available: https://wiki.onosproject.org/pages/viewpage.action?pageId=28836788
[17] “OpenDaylight Security Advisories” [Online] Available: https://wiki.opendaylight.org/view/Security:Advisories
[18] “ONOS Security Advisories” [Online] Available: https://wiki.onosproject.org/display/ONOS/Security+advisories
[19] IPERF tool documentation [Online]. Available: https://iperf.fr/iperf-doc.php
[20] TSHARK tool documentation [Online]. Available: https://www.wireshark.org/docs/man-pages/tshark.html
[21] Pakzad, Farzaneh, et al. ”Efficient topology discovery in software defined networks.” Signal Processing and Communication Systems (ICSPCS), 2014 8th International Conference on. IEEE, 2014.

