Evaluation of the Detection Capabilities of the ONOS SDN Controller

Christopher Metter*, Valentin Burger*, Zheng Hu†, Ke Pei† and Florian Wamser*

*University of Würzburg, Institute of Computer Science, Würzburg, Germany
{christopher.metter|valentin.burger|florian.wamser}@informatik.uni-wuerzburg.de

†Huawei, Corporate Reliability Departement, Shenzhen, China
{hu.zheng|peike}@huawei.com

***Abstract*—The process of monitoring the network, especially for larger ones, is very complex and contains many pitfalls. For instance the balance between granularity of information and their performance impact on the network. With the help of SDN this challenge can become easier, as it offers new methods, mechanisms and opportunities. One of the current most important Open Source controllers is the ONOS SDN controller. According to its developers it is a production ready controller that offers high availability due to its logically centralized and physically distributed architecture. But, as our investigations show, it is unable to cope with hazardous network conditions such as sporadic or recurring packet loss. It either does not detect packet loss or only detects it after long time periods, failing all common network availability targets.**

***Index Terms*—SDN controller; Network monitoring; Detection; ONOS**

I. Introduction

SDN is an increasingly important technology that breaks up the ossified structure in networking: It decouples the control from the data plane of network devices [1]. With this shift in networking, it is possible to centralize the control plane of many devices into one single software, the so called controller. This controller opens up many new possibilities, such as a flexible and programmable management station to steer, control, and monitor the network.

Providing a high level of availability always has been and will be one of the top challenges of network management. For the common user this struggle is only visible by some magic promise of services with ”up to x nines of availability” or is only discussed if a certain service, e.g. Google, is unreachable or users of a certain ISP are unable to access the internet [2]. In the background, providers have to monitor their services 24/7 in order to fulfill contracts and SLAs and not to lose money over them.

The process of monitoring the network, especially for larger ones, is very complex and contains many pitfalls. For instance, the balance between granularity of information and their performance impact on the network. With the help of SDN this challenge can become easier, as it offers new methods, mechanisms and opportunities. One of the current most important Open Source controllers is the ONOS SDN controller [3]. According to its developers it is a production ready controller that offers high availability due to its logically centralized and physically distributed architecture. But, as our investigations show, it is unable to cope with hazardous network conditions such as sporadic or recurring packet loss on network links. It either does not detect packet loss or only detects it after long time periods, failing all common network availability targets.

The contribution of this paper is twofold: At first we present a stochastic analysis of the theoretical detection performance of the ONOS SDN controller towards link impairing effects, namely packet-loss and jitter in the data plane. Second, we support our evaluation by measuring the detection performance of the ONOS controller for the case of packet loss in the data plane.

This paper is structured as follows. Chapter II presents related work and the backgrounds of the probing networks with SDN. Chapter III introduces a theoretical analysis of the detection mechanisms of the ONOS controller. Chapter IV establishes our testbed, the measurement scenario and the evaluation of the detection performance of the controller. Finally, Chapter V summarizes the content of this paper and gives an outlook to further work on this topic.

II. Failure Detection in SDN

The remainder of the chapter is structured as follows: At first, a definition of existing approaches towards failure detection and failure tolerance is discussed. Secondly, background on detection mechanisms in the OpenFlow SDN Protocol is presented. Finally, a general overview on detection mechanisms is given. The findings of this chapter form the basis for the further investigations.

A. Definition of Fast Failure Detection

In order to evaluate a *fast failure detection*, we first define common properties a fast failure detection mechanism should provide. At first, it is a hard requirement to detect failures within defined time constraints. The failures the solution should be able to detect are *node, link, network or controller failure*. In our case failure includes anything that imposes a deviation from normal operation of the network, e.g. device outage, lossy link, or a misbehaving controller. Additionally, failure detection is a fundamental building block for ensuring

fault tolerance in large scale distributed systems. Its main objective is to reduce the time it takes to detect a failure. The challenge is to meet carrier-grade requirements with a detection time of less than 50 ms without impacting the data traffic [4].

B. Background: Detection Mechanisms in the OpenFlow SDN Protocol

Current state-of-the-art SDN controllers mostly rely for their fast failure mechanisms on the limited detection capabilities of the southbound SDN protocol OpenFlow [5, 6]. A fundamental part to detect link failures is the LINE protocol. This protocol detects a direct physical link between two adjacent components, and, therefore, is able to detect a change in this connection. On average, the detection time ranges from 50 to 150 ms, and is, therefore, not suitable for the provider requirements with less than 50 ms of detection time. In SDN, after the line protocol has detected a failure, a notification is sent to the controller. The content of this message contains the affected switch, the affected port of that switch and the new port status (link down, blocked or live). Therefore, in an SDN-only network two messages are sent to the controller: one from each side of the link. Upon receipt, the controller reacts according to its programming. Though the line protocol is already in use for more than ten years, it also has some limitations. First, it is only able to detect a link failure at one local link. Furthermore, it cannot detect packet loss or temporary partial failures within the network. At last, it cannot detect network failures, e.g. the routing between two end hosts in a network is not possible, even if each of the used links is up and running according to the line protocol. Additionally, OpenFlow offers capabilities for flow monitoring. Each OpenFlow-enabled switch offers counters, which store traffic information in different granularity, e.g. per-table, per-flow and per-port (for example: bytes/packet count per flow). The controller queries these statistics at regular intervals. Although offering basic statistics, these options also have several limitations. Frequent polling has to be used in order to monitor the dynamics of a flow. Moreover, in general, polling of statistics induces load on components, for example, a higher CPU load on controller or a higher signaling traffic due to statistic packets. Therefore, these mechanisms are insufficient for carrier-grade OpenFlow/SDN deployments.

C. Probing Mechanisms

In general two types of probing exist: active and passive probing. Passive probing determines the state of the network by relying on in-network traffic monitors. E.g. by comparing the packet count statistics of a flow from two adjacent switches, the packet throughput and the packet loss can be computed. The advantage of this technique is that no additional measurement overhead is generated in the data plane. The disadvantage is that it is often not as accurate and fast as the second probing mechanism: active probing. Furthermore, these statistics have to be polled by a central mechanism, e.g. an application of the controller. This leads to additional load on the network and rises the problem of polling accuracy.

An active probing mechanism, in turn, inject special probing packets into the networks data plane. According to the investigated metric (e.g. packet loss, packet delay, throughput or congestion), the accuracy of the results and the resolution in time is higher than it is for passive probing. A non-neglectable disadvantage of the active probing approach is the induced measurement overload in the data plane by the measurement packets. In critical situations, such as a link overload, each additional packet that has to be transmitted via a link worsens the effect on the data plane traffic.

III. Failure Detection Analysis

This chapter is dedicated to the analysis of the performance of active probing. As our goal is to evaluate the detection times of ONOS, we at first need to consider the theoretical limits of the approach of time-out based probing. Afterwards, we measure the performance of our application in our testbed for the case of packet loss in the data plane.

A. Detection Time Analysis

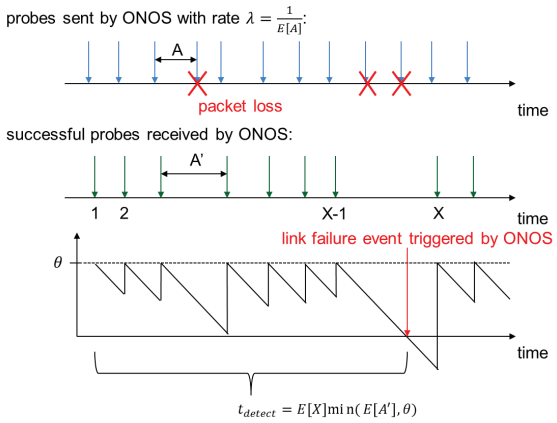

Fig. 1. Probing Process of one Link

1) Stochastic Analysis of Failure Detection Time: In order to identify the best probing rate, the current ONOS implementation is analyzed. To monitor a link ONOS sends probes with a fixed rate λ as shown in Figure 1. After each arrival of a probe a timer is reset to 0. The ONOS implementation detects a link failure and sends a link failure event if the timer exceeds a threshold θ, see Figure 1. To analyze the performance of probing we use A as random variable (RV) for the inter-arrival time of probes. We consider the packet loss probability p as the probability that a probe is lost. Due to the loss of probes and the network dynamics the process of successful probes that arrive at the controller differs from the inter-arrival process of probes. Therefore, we use A' as RV for the inter-arrival time of successful probes, with $E[A'] \geq E[A]$. The probability of a link failure event is then given by

$$p_{fail} = P(A' > \theta) = 1 - P(A' \leq \theta) = 1 - F_{A'}(\theta) \quad , \quad (1)$$

where $F'_A(t)$ is the cumulative distribution function of RV A'. The mean detection time of a failure is calculated by considering the number of successful probes X that arrive at the controller until the first successful probe comes late with probability p_{fail}. Considering that a failure event is triggered if $A' > \theta$, the mean detection time can then be calculated by

$$t_{detect} = E[X]min(E[A'], \theta) = \frac{1}{p_{fail}} min(E[A'], \theta). \quad (2)$$

Since X follows a geometric distribution with parameter p_{fail}, which has an expected value $E[X] = \frac{1}{p_{fail}}$. If the probes are sent with a constant rate λ, as in the ONOS implementation, the inter-arrival time of probes A is $\frac{1}{\lambda}$ with probability 1. In this case the inter-arrival time of successful probes A' can be calculated by considering Y as RV for the number of probes that are sent to the controller until a probe successfully arrives at the controller with probability $(1-p)$. Y follows a geometric distribution with parameter $(1-p)$ and has the CDF $F_Y(k) = 1-p^k$ and expected value $E[Y] = \frac{1}{1-p}$. Since the probes are sent with constant rate λ, we can calculate A' by $A' = Y \cdot A = Y \cdot \frac{1}{\lambda}$, with $E[A'] = E[Y] \cdot \frac{1}{\lambda} = \frac{1}{1-p} \cdot \frac{1}{\lambda}$ and $F_{A'}(t) = 1 - p^{\lfloor t \cdot \lambda \rfloor}$.

Hence, in this case the probability of a link failure is

$$p_{fail} = 1 - F_(A')(\theta) = p^{\lfloor \theta \cdot \lambda \rfloor} \quad (3)$$

and the mean detection time is equal to

$$\begin{aligned} t_{detect}(p, \theta, \lambda) &= \frac{1}{p_{fail}} min(E[A'], \theta) \\ &= \frac{1}{p_{fail}} min(\frac{1}{1-p} \cdot \frac{1}{\lambda}, \theta) \\ &= \frac{1}{p^{\lfloor \theta \cdot \lambda \rfloor}} min(\frac{1}{1-p} \cdot \frac{1}{\lambda}, \theta) \end{aligned} \quad (4)$$

In order to identify the optimal probing rate, the current ONOS implementation is analyzed. Based on the configuration parameters of ONOS, a calculation of multiple metrics is possible. In the following, two metrics will be evaluated: the mean detection times and the detection probability. The mean detection time describes the time interval between the configuration change of a link, e.g. from 0% packet loss to 5% packet loss, and the point ONOS is actually recognizing it. Link detection probability describes the probability ONOS is able to detect packet loss on a data plane link. Variable input parameters are the probing frequency λ, how often a probe is sent through a link per second, and the timeout θ, which determine the number of probing packets that are required to be lost consecutively in order for ONOS to realize a link failure.

Figure 2 shows the probability p_{fail} of the ONOS detection module to recognize a change in the links status. The x-axis depicts the data plane packet loss probability, the y-axis the probability an event is recognized by ONOS. Again, three probing frequencies are shown. From left to right: 1/3 s, 1/2 s and 1/1 s. The timeout has been set to $\theta = 9\,s$. For the default probing frequency of 1/3 s, the results show the highest detection probabilities. If the probing frequency is increased, a higher number of consecutive probes have to be lost so that the timeout is exceeded. Hence, for a fixed timeout, the probability of ONOS to trigger a link failure event decreases with the probing frequency. In order to detect link failure events with high probability for high probing frequencies, the timeout has to be reduced with the probing frequency.

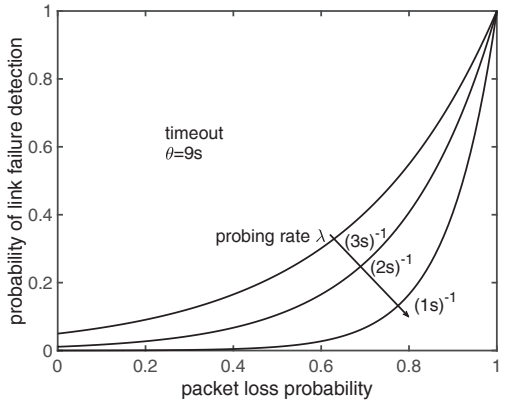

Fig. 2. Calculus - Detection Probability

Figure 3 shows the probability of a link failure event p_{fail} dependent on the packet loss probability for a fixed probing rate 1/3 s and different timeouts 3 s, 6 s and 9 s. The probability of detecting a link failure event increases with the probability that a probe is lost. With increasing timeout the probability of detecting a link failure event decreases, since a higher number of consecutive probing packets have to be lost so that the timeout is exceeded. If the timeout is set lower than the inter-arrival time of probes $\theta = \frac{1}{\lambda}$, a link failure event is triggered after each probe. Hence, to maximize the probability of a link failure event given a probing rate, the timeout is set to $\theta = \frac{1}{\lambda}$.

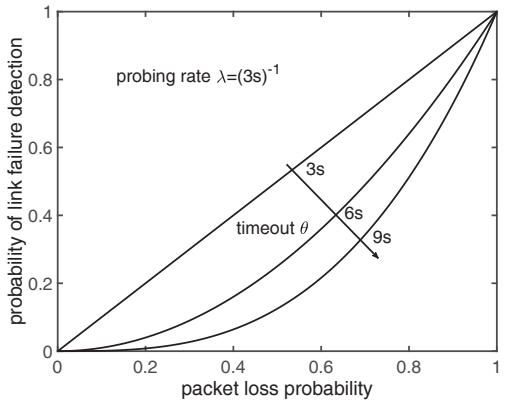

Fig. 3. Calculus - Detection Probability

Figure 4 shows the mean detection time calculation for a variable probing frequency and two timeout values. On the x-axis the data plane packet loss probability is depicted, the

mean detection time by ONOS in seconds is depicted on the y-axis. The solid line always depicts a probing frequency of 1/3 s, the dashed line a frequency of 1/2 s, and the dashed and dotted line a frequency of 1/1 s. Results in black have the timeout set to three times the probing frequency, for results in blue the timeout is set to one time the probing frequency. Apparently, ONOS mechanisms do not perform very well in this scenario. Only for data plane packet loss values greater than 45%, the mean detection time is less or equal to a minute. Increasing the packet loss further also decreases the detection time. But at 100% packet loss the detection time is still at more than 10 s. Increasing the probing rate from 1/3 s to 1/2 s also decreases the detection time. Here, mean detection times of less than 60 s can be found for a packet loss value of circa 35%. For 100% packet loss the mean detection time is around 8 s. Finally, for a probing frequency 1/1 s, the mean detection time decreases even further. Here, packet loss values around 25% lead to a detection time of around 60 s. The final detection time for loss values around 100% is less than 5 seconds. Decreasing the timeout value to the probing frequency also drastically decreases the detection time. Here, already for values below 10% of packet loss, detection times below 60 s can be found. Again, increasing the probing frequency leads to better results, but this time the improvement is smaller in comparison.

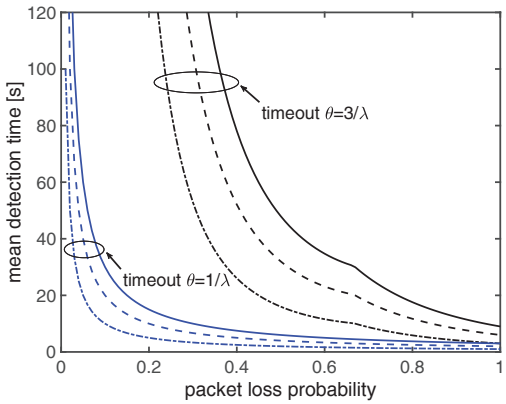

Fig. 4. Calculus - Mean Detection Time

B. False Positive Failure Rate

The results from Section 3.1 show that the mean detection time of a link failure indicated by packet loss depends on the probing rate λ and the timeout θ. As shown in Figure 5, the jitter on the link delays the arrivals of the successful probes at the controller. On a link with 0% packet loss the inter-arrival time of two successful probing packets A' can also exceed the timeout, due to jitter on the link. Hence, in this case a false positive link failure event is triggered even if no packet is lost. These false positive link failure events need to be avoided, as they would interrupt the operation of a healthy system. Especially if the probing frequency is high, a too low timeout θ can lead to false positive link failures due to jitter.

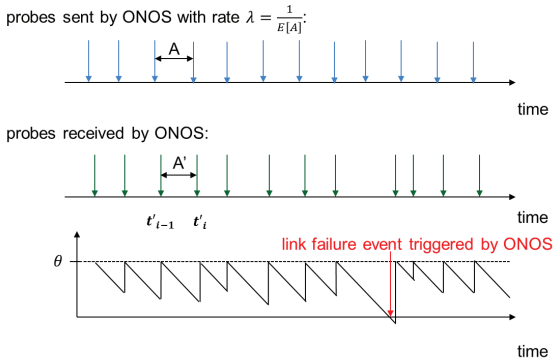

Fig. 5. Probing Process with False Positive caused by Jitter

In order to evaluate the rate of false positive link failures, we implement a Monte-Carlo simulation that simulates the probing process of one link. The simulation time is T, Probes are sent with rate λ and are dropped with probability p. Each probe is delayed by a normal distributed random time with parameters $(0, \sigma)$. The probing timeout is θ.

As performance metrics we consider the rate of link failure events

$$f_{fail} = \frac{1}{T} \sum_{i>0} X_i, \text{where}$$
$$X_i = \begin{cases} 1 & t'_i - t'_{i-1} > \theta \\ 0 & \text{else} \end{cases} \quad (5)$$

In case of $p = 0$ %, f_{fail} is also the rate of false positive link failure events.

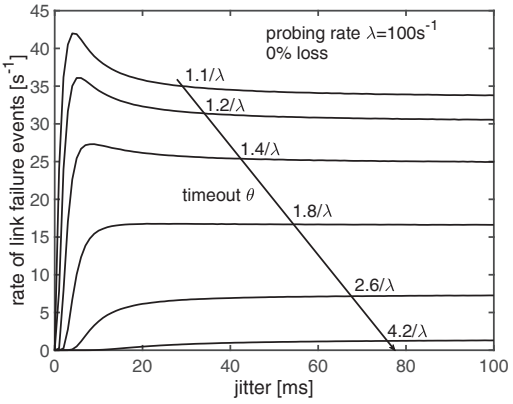

Fig. 6. Rate of false positive Link Failure Events caused by Jiteer

Figure 6 shows the rate of failure events for 0% packet loss, i.e., the rate of false positive link failure events caused by jitter. The probing rate is $1/100\,s^{-1}$, and the timeout is varied relative to the average inter-arrival time of probes $\frac{1}{\lambda}$. The results show that a small timeout θ leads to high false positive rate. This leads to a high probability of flapping of the link status. With increasing jitter a longer timeout $theta$

is necessary to keep the rate of false positive link failures low. Hence, for effective operation with a low rate of false positives, the timeout θ has to be set depending on the jitter on the link. As rule of thumb the timeout θ can be set using a margin of 2 times the jitter:

$$\theta := 2E[A] + 2\sigma = 2/\lambda + 2\sigma \quad (6)$$

Figure 7 shows the rate of false positive link failures for the rule of thumb setting for 100 and for 10 probes per second. The result shows that using the rule of thumb setting the less than one false positive link failure event is triggered every 10 seconds.

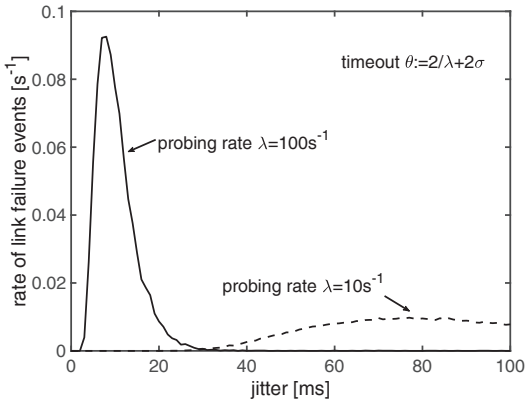

Fig. 7. Rate of false positive Link Failure Events for Equation (6)

The minimum value for the timeout θ can be calculated by evaluating the inverse cumulative distribution function of a normal distribution $N^{-1}_{1/\lambda,\theta}(p)$ with mean $\frac{1}{\lambda}$ and standard deviation σ depending on the tolerated false positive rate $\bar{f}_{fail}$:

$$\theta_{min}(\lambda, \sigma, \bar{f}_{fail}) = N^{-1}_{1/\lambda,\sigma}(1 - \bar{f}_{fail}) \quad (7)$$

IV. Performance Evaluation

This section provides a thorough investigation of the ONOS failure detection performance for the case of packet loss. At first, the testbed, the measurement scenarios, and the evaluated performance indicators are introduced. Afterwards, the results are presented, evaluated, and discussed.

A. Testbed Description

Figure 8 shows the testbed used for the measurements of this section. One physical server is running Mininet to emulate a network topology of four switches. The topology of the testbed is a ring-topology with four switches. To each of the switches one simulated host is connected. Two physical servers form the ONOS controller cluster. The switches are load-balanced between these two nodes, i.e. one controller node controls two switches.

In order to test the detection capabilities of ONOS for the packet delay, packet loss is configured on the data plane link between switch 1 and 2. To be able to determine the reactions

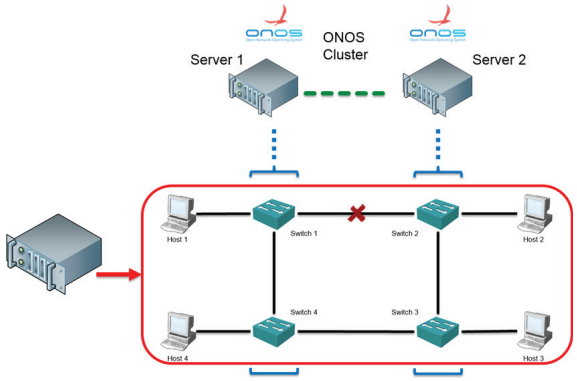

Fig. 8. Testbed Overview

and their delay, the signaling traffic between the controllers and their connected switches is recorded and evaluated after each run. Analyzing these traces allows us to calculate the mean reaction time and the detection probability of each scenario.

B. Measurements

In this section we evaluate the detection capabilities of ONOS for the case of packet loss based on measurements. Finally, we compare these results with the predicted performance presented in Section III.

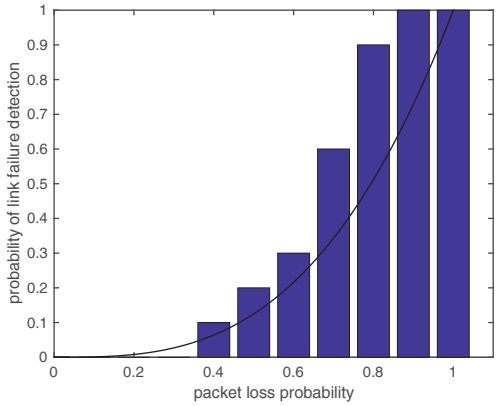

Fig. 9. Number of successful detections for the ONOS controller for the case of packet loss

In Figure 9 the detection probability of the ONOS controller for multiple packet loss values is depicted. The x-axis shows the configured packet loss in the data plane, the y-axis the detection probability after 10 measurement repetitions with a measurement duration of 120 seconds, each. The blue bars visualize the measurement results, the black line the according model results for comparison. These measurement results only begin after a configured packet loss value of 30% as ONOS is unable to detect any change in the inter-connection of the connected switches for lower values. Taking a look at Figure 4 covers these results, as the expected mean detection time for

packet loss below 35% is beyond our measurement duration of 120s. Beginning with 40% packet loss ONOS slowly and unreliably begins to detect the change in the link quality with a detection probability of 10%. Increasing the packet loss further to 50% and 60%, the detection probability rises to 20% and 30%. For packet loss values beyond 70% ONOS is able to reliably detect a change in the network conditions with a detection probability of 90%. After that the detection probability remains at 100%.

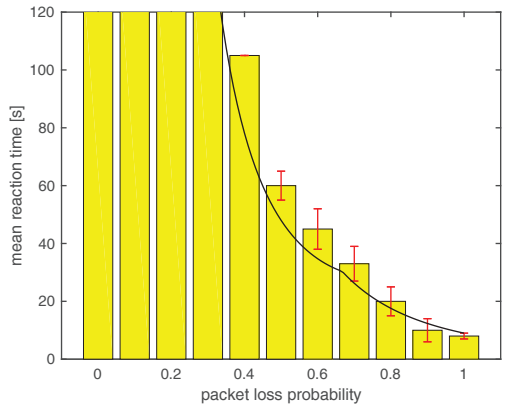

Fig. 10. Mean Reaction Time of the ONOS controller for the case of packet loss

Figure 10 shows the mean reaction time of the ONOS controller in this measurement scenario. The x-axis shows the range of measured packet loss values. The y-axis depicts the mean reaction time in seconds, respectively. Each results is shown as a yellow bar; additionally, 95% confidence intervals are shown in red. For the purpose of comparison, the calculated model values are shown as a black line. As these results have been derived from the same measurements as the ones from Figure 9, there are no results for the mean reaction time for 30%, depicted here as a bar filling the whole height of the figure. With a configured packet loss value of 40%, ONOS takes 105 s to detect a change in the data plane. For 50% the reaction time decreases to around 60 s. A packet loss of 60% leads to reaction time of around 45 s. 33 s are required for a detection of 70% of packet loss in the data plane. Further increasing the packet loss value from 80% to 100% leads to a decrease of the reaction time to 20 s, 10 s, and 8 s. As the confidence intervals show, different variances of the reaction time have been recorded. Throughout the results, all confidence intervals are small, indicating a low variance of the results.

Comparing these measurement results to the model results in general confirms our theoretical analysis. But especially for the detection time an offset between measurements and model becomes evident. This can be attributed to the difference in the two displayed metrics: detection time vs reaction time. With the detection time t_{detect} we express the time the internal detection mechanisms of ONOS require to "realize" that a link is exposed to hazardous conditions. The reaction time t_{react} actually can be expressed as a sum: $t_{react} = t_{detect} + t_{calc}$, where t_{calc} identifies the time ONOS requires to calculate a the reaction to the detected event. Within our testbed we are unable to measure t_{detect} directly, therefore, we only capture t_{react}.

V. Conclusion

Software-Defined Networking is gaining momentum since its introduction at the end of the last decade. Through its decoupling of the control and the data plane of network devices, it is possible to configure the network in a very flexible and central manner. These advantages lead to an interest by service providers. But, before migrating to a new technology, providers have to be ensured that, despite the possible advantages, their reliability and availability requirements are met.

One of the currently most important SDN controllers is ONOS. According to the ON.Lab, the company developing this controller, it is very reliable and production ready. In this paper we theoretically analyzed the capabilities of the detection mechanisms of ONOS and verified this analysis by exemplary measurements. According to the presented results, the implemented mechanisms only provide an unreliable detection and are therefore unable to fulfill the requirements of service providers. For example, it can take more than one minute to detect packet loss values of 50% in the data plane.

Reducing or disabling the threshold of failure mechanisms is a possible approach to increase the fast-failure reaction performance. Usually, most detection and reaction mechanisms have implemented safety margins, so that the controller does not already react to the slightest possible failure in the network. For example, the standard probing application of the ONOS controller only reacts to three consecutively lost probing packets. The reason for this behavior is that there always is a very small packet loss and jitter within a network. Therefore, it is only logical to take this in mind when designing detection algorithms, as so-called false-positive failure events can lead to congestion on the calculated backup links, which, in turn, would trigger a real link failure on that link. However, by decreasing these safety thresholds or even disabling them, a reaction to real failure can also be detected in less time.

References

[1] M. Jarschel, T. Zinner, T. Hossfeld, P. Tran-Gia, and W. Kellerer, "Interfaces, attributes, and use cases: A compass for SDN," *Communications Magazine, IEEE*, vol. 52, no. 6, pp. 210–217, June 2014.

[2] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[3] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "Onos: towards an open, distributed sdn os," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 1–6.

[4] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Openflow: Meeting carrier-grade recovery requirements," *Computer Communications*, vol. 36, no. 6, pp. 656–665, 2013.

[5] B. Heller, "Openflow switch specification, version 1.0.0," http://www.openflow.org/documents/openflow-spec-v1.0.0.pdf, OpenFlow Consortium, Tech. Rep., 2009.

[6] O. S. Specification-Version, "1.4.0," 2013.