# Architecture of a Cloud-based Fault-Tolerant Control Platform for improving the QoS of Social Multimedia Applications on SD-WAN

Kashinath Basu[1] , Aws Hamdullah[2], and Frank Ball[3]
[1, 2] School of Engineering, Computing and Mathematics, Oxford Brookes University, Oxford, UK
[3] Frank Ball Consulting, Oxford, UK
Contact author e-mail: kbasu@brookes.ac.uk

*Abstract*— **Social media application are becoming multimedia centric with live and stored video, audio, augmented reality, haptic, etc. emerging as the main categories of traffic. Their QoS requirements are more stringent than their legacy counterparts. At the carrier level, Software Defined – Wide Area Network (SD-WAN) is one of the promising technologies for transporting these multimedia traffic. A SD-WAN will typically have a mesh of centralized controllers managing the networking infrastructure. Reliable operations of these controllers are a key requirement for the successful operation of the WAN. Controller failure will prevent the forwarding switches from communicating with the controller. This will prevent the switches from forwarding any new traffic, as well as flow entries from existing traffic will also time out after a period bringing the network to a standstill. Rebooting a controller or starting a new one will introduce delays degrading the QoS. This research presents an architecture for handling controller failure via transparent migration of the controller load in a semi-meshed controller environment. The architecture includes a real time cloud-based centralized storage of the flow states north of the controllers and a virtualized connection management unit at the south. The results demonstrate that the proposed model can transparently handle controller failure without affecting the QoS.**

***Keywords— SD-WAN; QoS; social media; Software defined network; fault-tolerant; reliability; OpenFlow; cloud; NFV; flow table***

## I. INTRODUCTION

From the communication point, the legacy social media applications have been generally text and image based and had long tolerable delay bounds but required lossless and error free network service to maintain appropriate level of Quality of Experience (QoE). The traditional networking infrastructure and communication models was adequate to provision appropriate network level Quality of Service (QoS) to support the QoE of the users. In this setup, most issues of reliability and fault resulting from the breakdown and fault of the infrastructure devices and links were handled by a slow rerouting process of the traffic and rebooting of the infrastructure. This however had minimal impact on the QoE of the users since the applications were inherently delay tolerant. In contrast, several of the present and emerging social media applications such as YouTube, Facebook, Snapchat, vTime, AltSpaceVR, etc. include multimedia such as video, audio and animation (both in streaming and real time interactive form) as well as multi sensorial media such as haptic, olfactory, gustation, etc. along with text-based media [1]. These wide varieties of media demand a much stringent QoS with seamless continuous and synchronized playback with minimal loss and error. This can be partially handled at the network level with adequate orchestration of network resources to keep delay and jitter at a minimal. However, without a comprehensive fault-tolerant framework, these network level QoS targets can be jeopardized resulting in the degradation of the QoE.

This reliability concern is more serious in the context of the emerging SD-WAN networking model which is an extension of software defined networking (SDN) for enterprise and WAN. The model is based on two key concepts: a) decoupling of network control and transport functions into two separate planes and b) centralization of control and management. To realize this model, the architecture can be broadly classified into four key components (Fig. 1) across these two planes: 1) the forwarding switching fabric in the data plane, 2) the controller platform and the 3) northbound application programming interfaces (APIs) that controls the packet treatment in the control plane and the 4) southbound control/data plane signaling interface between the switches and the controller. In this model, the implication of network failure and fault is significantly different from traditional network. For example, in the context of packet treatment in a traditional network the forwarding decision is made either locally or based on intelligence gathered in co-ordination between distributed forwarding devices. In this traditional set up, most local failure has minimal impact and only limited regional implications.

978-1-7281-5611-8/20/$31.00 ©2020 IEEE

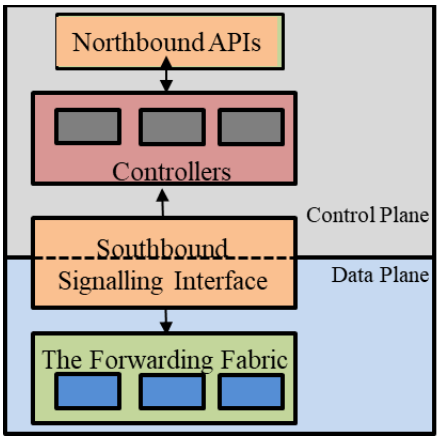


Fig. 1: The four components of the SD-WAN architecture

However, in the SD-WAN model, failure and fault is more critical since control and network management is centralized at the control plane. This model can be categorized into three fault domains (Fig. 2). The forwarding fault domain includes failure in the data forwarding plane and includes switches and links. The interface fault domain involves failure of north bound and southbound interfaces. The northbound interface is generally a logical API based interface with the northbound applications running either locally on the controller platform itself or on a separate server. The southbound interface involves signaling APIs and physical communication interfaces between the forwarding devices and the controller(s). Finally, the controller fault domain involves failure in the control platform, the brain of the SD-WAN network responsible for network state aggregation, liaison with the northbound application and updating of the flow tables of the forwarding devices.

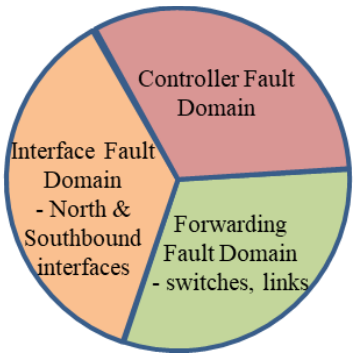


Fig. 2: Categorization of the SD-WAN architecture into failure and fault domains

A fault in the switch or link in the forwarding domain of a SD-WAN network is simpler in some aspect to similar situation in a traditional WAN network. In the SD-WAN, the flow rules of the faulty switch in the centralized controller are unaffected and the controller can temporarily reorganize the topology to bypass the faulty region. Similar action can be taken for faulty links. Once the switch or link is replaced, the topology can be updated without any additional delay in relearning the topology. The other two fault domains, the interface and the control domain are critical and go hand-in-hand as a fault in one has a cascading effect on the other and will result in the failure of the entire network. When there is a fault in either of these two domains, the data plane will not be able to receive any flow table updates and existing entries will also eventually timeout. This will bring the entire network to a standstill. Several papers have referred to the significance of this problem; however, most works have primarily focused on the problems in the data forwarding plane [2, 3].

Reliable fault-tolerant solutions of the control and interface domain are crucial for the wide scale deployment of the SD-WAN by carriers and telecom operators for forwarding social multimedia and other categories of time critical and isochronous traffic. A controller could fail for various reasons such as of server crash, power failure, security breach, etc. Similarly, the link between the controller and the switches can also break. In this paper, the focus is not on failure avoidance but on recovery after failure through transparent load migration with minimal disruption of the services. Here a network is considered to be consisting of a semi meshed set of controllers which is the ideal setup in both enterprise and carrier grade SD-WAN. The rest of the paper is organized as follows: Section 2 gives a brief analysis of the traffic characteristics of social multimedia traffic and the significance of a fault-tolerant reliable networking infrastructure; Section 3 presents related work on fault tolerance in softwarized networks; Section 4 describes the architecture of the proposed fault- tolerant control plane model. For performance comparison, an alternative fully meshed model is also presented. This however is not scalable but suitable as a benchmark for comparison; in section 5 the simulation results of the performance of these models in a mininet environment with POX [4] controllers is discussed; and finally section 6 evaluates the nobility of the work and presents the concluding remarks.

## II. SOCIAL MULTIMEDIA TRAFFIC CHARACTERISTICS

There are wide categories of social media applications ranging from blogging, image sharing, video streaming, live video, interactive animation to augmented reality with multisensory data. These applications generate a wide variety of traffic profiles and have different levels of QoS requirements in order to provide the optimal level of QoE for the end users. For example, text centric blogging sites generate text-based data which require loss and error free service but is relatively tolerable to reasonable end to end delay 800 milliseconds. Hence, these data are packetized in larger packets since longer packetization delay is not a concern. Also, these types of social media sites generally produce bursty traffic. On the other hand, live video/audio and augmented reality traffic have a much smaller end to end delay bound of 200 – 250 milliseconds and hence have a smaller packet size to limit the packetization delay. The traffic is continuous with either isochronous or synchronous characteristics with very limited tolerable jitter. Here, any small disruption in the transmission will immediately result in the impairment of the QoE. Hence, the reliability of the infrastructure is crucial. In the SDN-WAN environment, due to the centralization of the control plane any fault in the controller will have cascading impact on the performance of the forwarding devices resulting in additional latency and delay for the applications [5]. Therefore, a reliable control plane that is robust and fast in managing fault is vital in SD-WAN.

## III. RELATED WORK

The SD-WAN controller manages the flow entries of the forwarding switches by sending the required commands that

either create new or update existing flow entries within the flow tables in the WAN switches [6]. Hence, it is vital to continue the operation of the control plane in the case of controller failure in order to maintain the proper operation of SD-WAN and the related services.

There are a number of open source and commercially available controller products in the market. Some of these have evolved as a community driven initiative such as NOX/POX, OpenDaylight, Floodlight, Open Network Operating System (ONOS) [7,8] whereas other came from established networking industry vendors such as Juniper's Contrail, HP's Virtual Application Network (VAN), Cisco's Application Centric Infrastructure (ACI), etc. In most of these products, the main focus is on the reliability and fault tolerance of the switches and links in the forwarding domain. In the control domain some of the fault tolerance approaches include fast reboot of the failed controller, redundancy across controller cluster [9], using centralized database for fault tolerance and replication [10], optimization of the location of the controller in the subnet [11], etc. There are big questions on the scalability and performance of these solutions in terms of recovery time and loss [12].

There are a number of ongoing researches specifically focused on the fault tolerance issues of the controller. [13] proposed a fault-tolerant procedure based on a distributed control layer. This distributed controller architecture claims to improve the availability, reliability and efficiency of the softwarized network. The work is based on an external Java based open-source toolkit named JGroups [14]. It can be used to discover and cluster networks elements by broadcasting or unicasting control messages among the network resident hosts. It supports the functionality of networks monitoring and failure detection. During controller failure, it performs load balancing among the controllers by dynamically migrating the forwarders from one controller to another.

Kim et al. presented another methodology for SDN controller's fault-tolerance which includes building controller's failure detection and recovery procedure inside a software module named CORONET [15]. Here, the failure detection method is based on a heartbeat mechanism that can be used to monitor a network device's status. It is represented by packets being broadcasted at regular intervals within a network [16]. As for the recovery, it uses a distributed dictionary or hash table hosted on the Onix distributed control platform [17].

[18] presented a mechanism for SDN failure recovery using the OpenFlow protocol. In this mechanism, the lifespan of a flow state entry in the switch is varied to recover from a failure. Here the arrival time of the last packet of each flow is recorded and the expiration interval of the flow entries is managed by using an idle and hard timeout period: The idle-timeout is the idle interval in which if no packet is matched by a flow it will be purged. The hard timeout is the interval after which a flow entry will be removed from the flow table regardless of how many packets are matched [19]. These timeout signaling primitives are part of the standard fields of the OpenFlow protocol and can be managed by the controller without any external module.

On the forwarding domain, there has been substantial volume of work dealing with switch failure and recovery. [20] presented a model for fault recovery in OpenFlow switches using a reliable proactive and reactive mechanism. In the proactive case, the controller calculates an alternative path before the occurrence of a switch fault, whereas in reactive, the controller will calculate the alternative path after it has been notified of a switch failure. The work suggests storing all the policies, rules and flow tables of the SDN resident switches in a single compressed controller unit. In the case of a switch failure, the controller will react by switching to the alternative path in order to be able to connect to the rest of the forwarders that reside on the other side of the failed switch. In addition, it will update the alive switches with the alternative path from a network-wide compressed forwarding state table.

Another approach used by some of the controllers [21] is the use of clustering to diminish the probability of the single point of failure and also to provide load balancing functionality. Clustering has been demonstrated to increase the scalability, performance and reliability of a network with the increase in the number of clustered controllers. However, controller clustering requires adequate synchronization of the flow rules among the controllers. Another related approach is to interconnect the software defined switches to multiple controllers to handle failure [22]. In this case, the first controller is treated as a master and the rest as slaves. This however requires a transparent handover phase for the switch to migrate from a master to a slave during failure.

Our proposed architecture addresses some of the shortcomings of the research work discussed above and builds on some of the good ideas to produce a novel and robust solution.

## IV. Architecture of the Proposed SD-WAN Control Platform

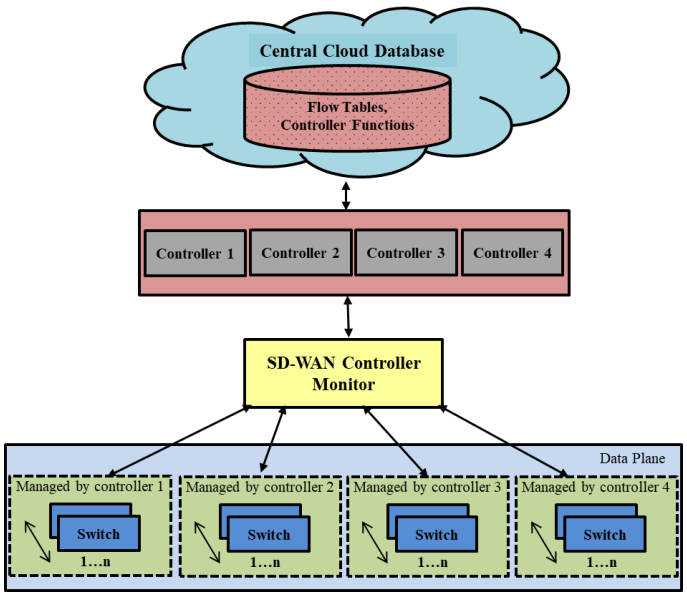


Fig. 3: Architecture of the fault-tolerant SD-WAN control platform

Fig. 3 presents the architecture of the SD-WAN fault-tolerant framework for the deployment of a reliable SD-WAN in a distributed controller-based environment. Having a distributed set of controllers provide the reliability, redundancy and scalability as required in a WAN environment. In the framework, each controller directly manages a set of physical and/or virtual switches and act as the primary master controller for the set. In addition, a controller may also act in parallel as a secondary slave controller for a different set of switches. The slave controller temporarily takes over as the primary controller for a set only when the master controller is unavailable. The switches are connected to the controllers through an intermediate controller monitor called here as the SD-WAN Controller Monitor (SCM). This logical unit can be implemented as a virtualized distributed network function unit like a Network Function Virtualization (NFV) component. This unit is responsible for monitoring the status of the controllers and detection of failures. It checks the status of the connection by regularly broadcasting hello messages to the controllers which are replied back if the controller is alive. If a failure is detected, the SCM is then responsible to transfer the connections of all the switches from the primary master to the secondary slave controller. At the northbound interface of each of the controllers, it is connected to a cloud-based centralized database which stores a copy of the flow rules. It is stored as a hash table for easy retrieval and can be invoked through application programming interface (API). When a controller failure is detected by the SCM, a copy of the flow table of the failed controller is migrated from the cloud to the corresponding secondary slave controller which takes over the control of all the switches from the former and operates as a temporary acting master controller for them. At a later stage, when the failed controller becomes alive again, it updates it current state from the cloud and the SCM transfers back the control of the switches. This whole process of failure, transfer of control and recovery of the controller is transparent to the switches.

## V. The Benchmark Control Platform

This model is used here only as a benchmark to compare the performance of our model above against this ideal scenario. In this setup (Fig. 4), all the switches are connected to all the controllers in the subnet in parallel in a fully meshed configuration. All the controllers receive the same flow information and maintain identical flow tables. However, from the controller to switch communication, the switch receives identical replies from each of the controllers but only processes the first copy and drops the other remaining copies. This setup ensures the performance of the switches are completely unaffected by the any particular controller failure. Although this solution is not scalable in a real implementation, it is suitable here for comparison purpose.

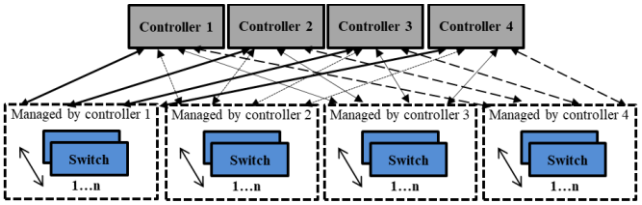


Fig. 4: The Benchmark control platform

## VI. Experiment Setup and Results

### A. Setup

The SD-WAN network was designed using Mininet with POX controllers and Open vSwitch (OVS) switches. OpenFlow was used for southbound signaling and the SCM and the cloud-based hash tables' remote procedures were written in the Python programming language for compatibility with POX's Python core. All the connections between the switches and the SCM as well as between the controllers and the cloud were setup using TCP sockets. The iPerf3 [23] toolkit was used to configure and capture flow statistics. The htop [24] package was used to measure the memory and CPU usage of the various modules.

The SD-WAN network was configured with eight OVS and four controllers (Fig. 3). Each of the controllers directly managed two switches acting as their primary master controller. In addition, each controller also acted as the secondary slave controller for a different set of switches. The secondary slave controller for the switches managed by controller-1 was set to controller-2, those managed by controller-2 was set to controller-3 and so on. The link rates were configured at 100 mbps and the aggregated social multimedia traffic load were kept at 60mbps. A mixture of flows with different start and inter-arrival times with a range of packet length distribution were used to simulate different categories of social multimedia traffic. All experiments were run for 90 seconds; however, the first 30 seconds were only for stabilizing the network and the flow tables. Hence, we only focus on the last 60 seconds duration in our discussion. In this 60 second interval, the controller failure was set at the 4th second and restoration point after the 44th second.

## B. *Results*

Before analyzing the performance of our model, the first experiment shows the impact of hard and idle timeout on the performance of the open vSwitch during controller failure. As mentioned earlier, idle timeout is the maximum idle interval for an existing flow entry to be matched with a new packet arrival; otherwise the entry is deleted from the forwarding table. Hard timeout is the maximum duration of a flow table entry after which it is purged irrespective of the amount of matching traffic. In this first experiment, we compare the performance of the OVS in three scenarios. In the first two cases, there are no fault-tolerant mechanisms in the network. In one case, only the idle timeout set to 30 seconds, and in the other the hard timeout set to zero. This is compared against the performance of the benchmark controller. In all subsequent experiments, only hard timeout is used to clearly distinguish the performance gain of the proposed SD-WAN platform without the assistance of any idle timeout period.

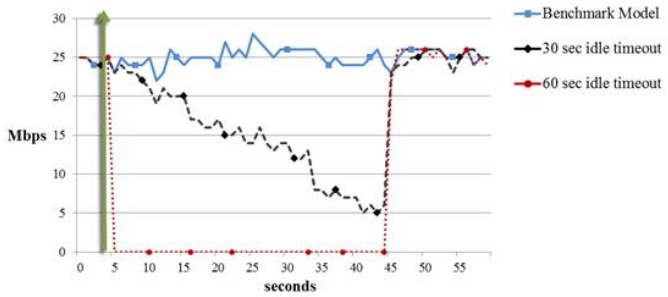


Fig. 5: Impact of timeout period on the throughput of a switch during controller failure

Fig. 5 shows the impact of the failure of the controller on the traffic load of the directly connected OVS. As mentioned, the proposed fault-tolerant platform has not been considered here. Instead the timeout impact in the two cases have been compared against the benchmark model. In the result, the OVS in the benchmark threshold model is completely transparent to the failure since it has direct connection with all the controllers and therefore it continues to receive openflow messages from the other identical controllers after the failure. In the case with 30 second idle timeout, all existing flows continued to be unaffected for 30 seconds depending on the last packet arrival. New flows however were dropped along with packets from existing flows which were idle for more than 30 seconds. In the case with zero second hard timeout, the throughput of the switch immediately becomes zero after the controller failure. The result shows that idle timeout duration can partly compensate the throughput of a switch connected to a failed controller. However, it is challenging to select an optimal timeout period that on one hand during fault will provide transparent continuity of the social multimedia traffic and maintain its QoE, and on the other hand during normal condition will still ensure that the flow table entries are still valid and up-to-date and have not become obsolete due to dynamic route changes.

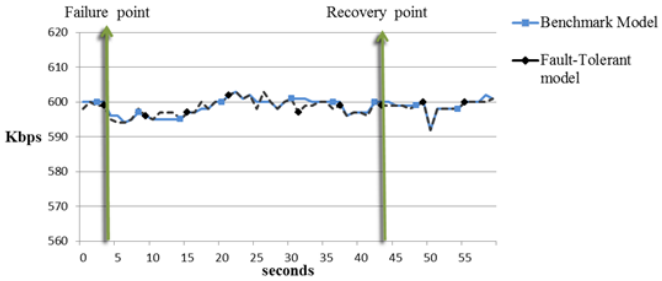


Fig. 6: Comparison of performance between the benchmark and the fault-tolerant models

Fig. 6 shows the comparison of the performance of the fault-tolerant SD-WAN platform with the benchmark model in dealing with controller failure. Here the effect of controller failure is monitored at a finer granularity by inspecting the performance of an individual flow passing through the OVS connected to the failed controller-1. At the 4th second, immediately after the failure the SCM control monitor passes the control of the switch to the secondary slave controller (controller-2) and the associated flow rules are transferred from the cloud. The process takes less than 200 milliseconds and 0.06% of the traffic during the period is lost. This minimal disruption of the social multimedia traffic can be readily handled at the application level in the end systems using various adaptive techniques [5]. Moreover, with a realistic idle timeout period like in the previous experiment, this loss can be further reduced close to zero. At the 44th second, when the controller becomes alive again control is passed back to the original primary controller. There is no loss during this handover process which takes place only after the controller is up and running and has synched the flow rules from the cloud. The result shows the flow rate has been almost unaffected by the controller failure and is identical to the benchmark model. Similarly, it was found that for the aggregate traffic passing through the switch the overall throughput remained unaffected.

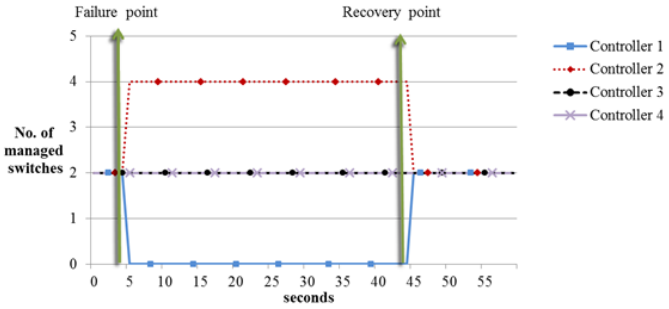


Fig. 7: Impact of timeout period on the throughput of a switch during controller failure

The change in load distribution across the controllers during the experiment run is presented in Fig. 7. After the failure, the number of switches managed by controller-2 goes up from two to four as it now acts as the primary controller for its own two switches as well as for the two switches from controller-1. Detail inspection of controller resource usage also show that the memory and CPU utilization increased from 18% to 30% during this stage. This later again returns

back to the earlier state when controller-1 comes alive and is transferred back the management of its two switches.

## VII. Evaluation and Conclusion

The results demonstrate that the proposed fault-tolerant SD-WAN platform can handle controller failure in a distributed controller-based SD-WAN environment. Both the coarse level aggregate throughput of a switch and the granular level individual flow rate of the social multimedia traffic were negligibly affected by the failure of the primary master controller. Any small minimal loss during the controller migration phase after failure can be offsetted by the idle timeout period as seen in the first experiment or by application level adaptation mechanisms. The additional load on the secondary controller during the failure period did not overload the resources as found from the memory and processor usage.

The proposed solution is robust and provides transparency to the physical and virtual switches and the traffic carried over them from fault and breakdown of the controllers in an SD-WAN. This is one of the key components to maintain reliability of the SD-WAN infrastructure. Adequate orchestration and provisioning of network level QoS along with the proposed fault-tolerant platform can provide consistent QoE of the social multimedia applications on SD-WAN.

There are some optimizations possible on the proposed platform. At this moment, the copies of the flow rules are stored centrally at the cloud. As shown, this provides both redundancy and recovery from failure. Furthermore, the SCM control monitor is presently implemented as a single module. In a scalable SD-WAN network, this could be implemented as a NFV module and deployed in a cloud and offered as a cloud-based network as a service (NaaS) function. These enhancements can further improve the response time of the system and hence the performance of the proposed platform.

## References


[1] L. Skorin-Kapov, M. Varela, T. Hoßfeld, and K.-T. Chen: 'A Survey of Emerging Concepts and Challenges for QoE Management of Multimedia Services', ACM Transactions on Multimedia Computing Communications and Applications (TOMM) (SI on QoE Management), vol. 14, no. 2, Apr. 2018

[2] C. M. Machuca, S. Secci, V. Vizarreta, et al. (20: 'Technology-related disasters: A survey towards disaster-resilient Software Defined Networks', Proc. 8th Int. Workshop on Resilient Networks Design and Modeling (RNDM), Halmstad, 2016, pp. 35-42

[3] C. Cascone, L. Pollini, D. Sanvito, A. Capone and B. Sansó: 'SPIDER: Fault resilient SDN pipeline with recovery delay guarantees', Proc. IEEE NetSoft Conference and Workshops (NetSoft), Seoul, , pp. 296-302G. Eason, B. Noble, and I.N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529-551, April 1955.

[4] POX Wiki Open Networking Lab (2015), [online] https://openflow.stanford.edu/display/ONL/Home (Accessed: 27 Nov 2019).

[5] M. Karakus and A. Durresi: 'Quality of service in software defined networking: A survey', Journal of Network and Computer Applications, Elsevier, vol. 80, pp. 200-218, Feb 2017

[6] D. Mattos, N. Fernandes, V. Costa, L. Cardoso, M. Campista, L. Costa and O. Duarte: 'OMNI: OpenFlow management infrastructure', Proc. International Conference on the Network of the Future (NOF), IEEE, Paris, 2011.

[7] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov and R. Smeliansky: 'Advanced study of SDN/OpenFlow controllers', Proc. 9th Central & Eastern European Software Engineering Conference in Russia, ACM, p. 1, 2013.

[8] A. Bondkovskii, J. Keeney, S. van der. Meer and S. Weber: 'Qualitative comparison of open-source SDN controllers', NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium, Istanbul, pp. 889-894.

[9] M.C. Penna, E. Jamhour and M.L. Miguel: 'A clustered SDN architecture for large scale WSON', Proc. Advanced Information Networking and Applications (AINA), IEEE 28th International Conference on Advanced Information Networking and Applications, Victoria, pp. 374-381, 2014.

[10] F. Botelho, A. Bessani, F. M. Ramos, and P. Ferreira: 'On the design of practical fault-tolerant SDN controllers', Proc. IEEE 3rd European Workshop on In Software Defined Networks (EWSDN 2014), Budapest, pp. 73-78, 2014.

[11] Y. Hu, W. Wang, X. Gong, X. Que and S. Cheng: On reliability-optimized controller placement for software-defined networks. China Communications, 11(2), pp.38-54, 2014.

[12] D. Kreutz, F.M. Ramos, P.E. Verissimo, C.E. Rothenberg, S. Azodolmolky and S. Uhlig: 'Software-defined networking: A comprehensive survey', Proc. IEEE, 103(1), pp.14-76, 2015.

[13] C. Liang, R. Kawashima, and H. Matsuo: 'Scalable and crash-tolerant load balancing based on switch migration for multiple Openflow controllers', International Symposium on Computing and Networking, Shizuoka, pp. 171-177, 2014.

[14] JGroups: Available at: http://www.jgroups.org (Accessed 25 Oct 2019)

[15] H. Kim, M. Schlansker, J. Santos, J. Tourrilhes, Y. Turner and N. Feamster: 'CORONET: Fault tolerance for software defined networks', 2012 20th IEEE International Conference on Network Protocols (ICNP), Austin, TX, pp. 1-2, 2012.

[16] M. Yang and Z. Fei: 'Cooperative Failure Detection in Overlay Multicast', in Boutaba, R. et al (Eds.), NETWORKING 2005. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems. Springer, Berlin, pp 881-892, 2005.

[17] T. Koponen, M. Casado, N. Gude, et al. : 'Onix: A distributed control platform for large-scale production networks', Proc. 9th USENIX conference on Operating systems design and implementation (OSDI'10), Vancouver, pp. 351-364, 2010

[18] S. Sharma, D. Staessens, D. Colle, M. Pickavet and P. Demeester: 'A demonstration of fast failure recovery in software defined networking', Testbeds and Research Infrastructure. Development of Networks and Communities, Springer, Vol. 44, pp. 411-414, 2012.

[19] S. Hommes: 'Fault detection and network security in Software-Defined Networks with OpenFlow', PhD thesis, The Faculty of Sciences, Technology and Communication , Luxembourg University, 2014.

[20] Y. Zhang, S. Natarajan, X. Huang, N. Beheshti and R. Manghirmalani: 'A compressive method for maintaining forwarding states in SDN controller', Proc. ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN), Chicago, pp. 139-144, 2014.

[21] M. Ulema: 'Vulnerabilities and opportunities in SDN, NFV and NGSON', IEEE CQR 2014 International Workshop - Emerging Technology Reliability Roundtable, Manhattan College, USA, 2014.

[22] SDN Hub (2017) Experimenting with ONOS clustering: [online] http://sdnhub.org/tutorials/onos/experimenting-with-onos-clustering/ (Accessed 07 Oct 2019).

[23] iPerf - The ultimate speed test tool for TCP, UDP and SCTP (2017): [online] https://iperf.fr/ (Accessed: 4 Jan 2017).

[24] htop - an interactive process viewer for Unix (2017): [online] http://hisham.hm/htop/ (Accessed: 4 Dec 2019).