A Comparative Study on Fault Tolerance Methods in IP Networks versus Software Defined Networks Introduction

Article · April 2016

CITATIONS
5

READS
268

3 authors, including:

Mohammad Reza Parsaei
Shiraz University of Technology
33 PUBLICATIONS **66** CITATIONS
SEE PROFILE

Reza Javidan
Shiraz University of Technology
95 PUBLICATIONS **177** CITATIONS
SEE PROFILE

International Academic Institute for Science and Technology

International Academic Journal of Science and Engineering
Vol. 3, No. 4, 2016, pp. 146-154.

ISSN 2454-3896

International Academic Journal of Science and Engineering

www.iaiest.com

A Comparative Study on Fault Tolerance Methods in IP Networks versus Software Defined Networks

Mohammad Reza Parsaei [a*], Seyed Habib Khalilian [b], Reza Javidan [c]

[a] *M.Sc. Student of IT engineering, Shiraz University of Technology, Shiraz, Iran,*
[b] *M.Sc. Student of IT engineering, Shiraz University of Technology, Shiraz, Iran*
[c] *Faculty member, Department of IT and Computer Engineering, Shiraz University of Technology, Shiraz, Iran*

Abstract

Today, to create and maintain a secure communication in a network and implementation of sophisticated management policies on network devices, operators have to deal with low-level and exclusive configuration of vendors devices. Current IP networks by combining control and data surfaces, have kept flexibility of management rules at a very low level. It is both difficult to configure the network according to predefined policies, and to reconfigure it to respond to faults, load, and changes. However, a new style of computer networks, i.e. Software Defined Networks (SDNs) has introduced modern facilities to manage and configure the networks by separating the control level from the data level and this results in smarter, more flexible and controllable management and introduce new abstractions in networking, simplifying network management and facilitating network evolution. However, SDN is unable to survive when facing failure, in particular in large scale data-center networks. Due to the programmability of SDN, mechanism could be designed to achieve fault tolerance. Fault tolerance in a computer system can be obtained through redundancy in software, data or calculations. Such redundancy can be implemented in a static, dynamic or combined configuration. In this paper, we present a comprehensive survey on SDN. We start by introducing the motivation for SDN, explain its main concepts and how it differs from traditional networking, its roots, and the standardization activities regarding this novel paradigm. Next, we present the key building blocks of an SDN infrastructure using a bottom-up, layered approach. We also look at cross-layer problems such as debugging and troubleshooting. In particular, We study on fault tolerant techniques and methods in current IP networks and SDN. Finally an analysis of all current methods is discussed.

Keywords: Software Defined Networking, fault tolerance, OpenFlow, programmable networks

Introduction:

Fault tolerance is the ability of a communication system to face the faults when an event accrues, in a case that other user services of the system do not face fault. Usually fault tolerance is done in two stages (Sterbenz et al, 2013): fault detection which consists of a set of measures that determines failure of a link, switch or controller. And fault recovery which is a set of measures to ensure availability, reduction of packets lost and increasing throughput of the network in case of any fault. There are two models for fault recovery. The first one is protection in which at first the controller calculates two career and backup paths and leaves to the switch. In case of fault detection without the need to inform to the automatic controller, the backup path is used to send packets. The second is recovery in which the switch informs the fault to the controller after fault detection. Then the controller identifies the paths that this fault had an effect and calculates an alternative path. Then, the controller updates the network switches.

Software-based Networks:

It is a new architecture that has four essential features (Kreutz et al, 2015):

1. Control planes and data planes are separated from each other. The devices have become just a simple by separating control unit from hardware.
2. Packets transmission decision, unlike traditional networks that are based on the packet's destination, is defined flow-based. Flows are defined by a set of pack and actions parameters.
3. Control section has been transferred to a foreign entity called controller or Network Operating System (NOS). Controller is a software that provides platform resources and abstraction required for programming transmitter hardware.
4. The network is programmable by applied programs in the planes higher than NOS.

The difference between traditional architecture and SDN and simplicity of network management because of devices control focus are shown in figure 1.

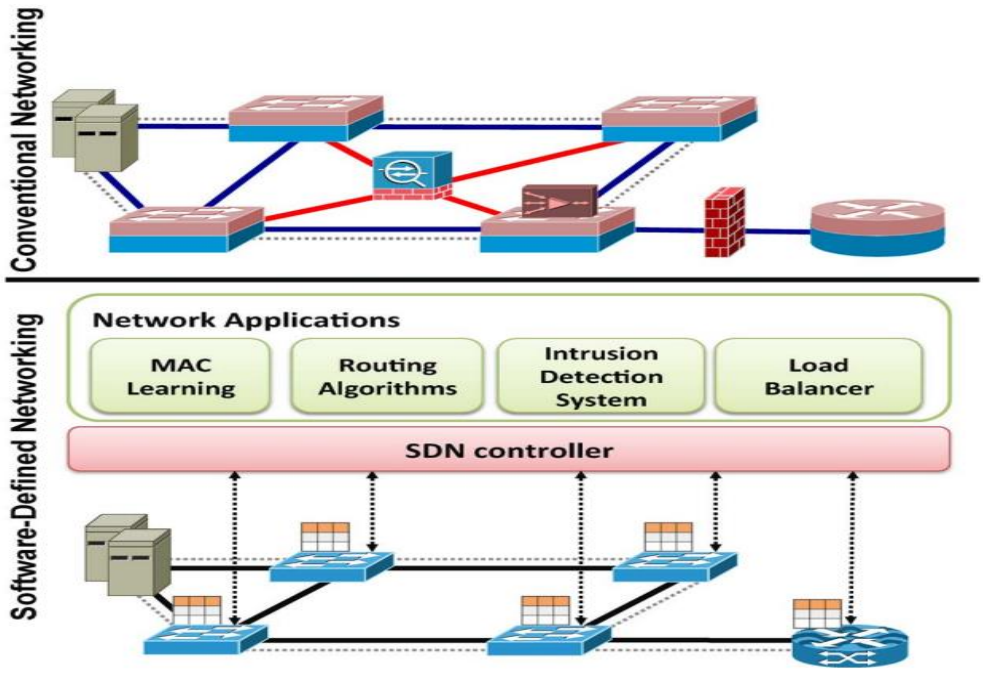

Figure 1: Software-based and traditional networks (Kreutz et al, 2015)

OpenFlow Protocol

OpenFlow protocol is used for communication between OpenFlow switches and SDN controller. SDN switches directs packets based on the rules for which the controller is sent and updated. The controller stores the rules to lead the pack in the form of flow tables or group table in the switches. The communication between the controller and the switches takes place in a secure channel by OpenFlow protocol, which is a standard and open source(Rowshanrad et al, 2015). OpenFlow protocol architecture and components are presented in Figure 2.

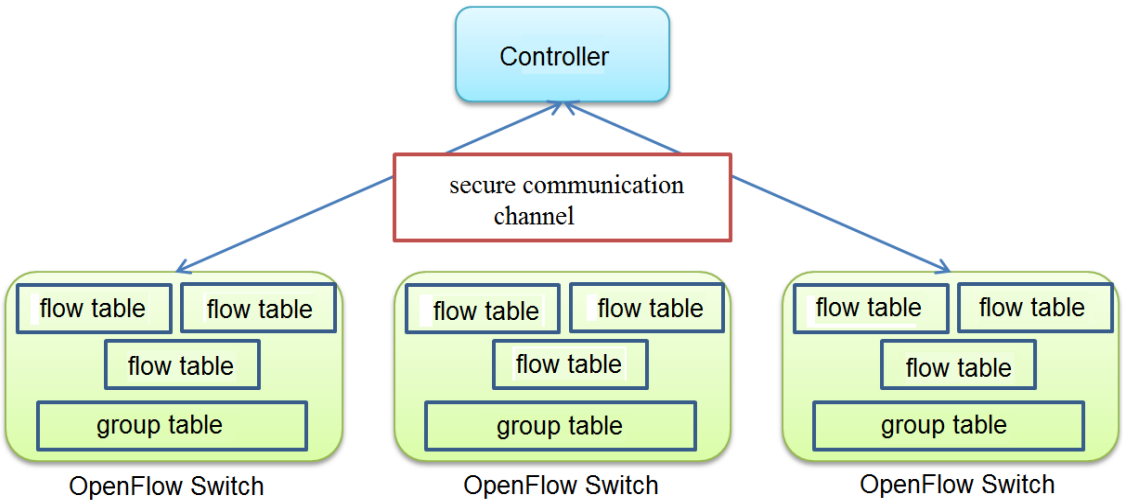

Figure 2: OpenFlow protocol architecture

OpenFlow Tables

From version 1.1 onwards in OpenFlow, each SDN can have one or many flow tables and a group table.

Flow Table

According to (Openflow switch specification, 2011) each entry of flow table include the following:
First, Match Fields which contains information by which the packets are distinguished. This field contains information about the packets header, ingress port and optionally metadata from earlier flow table.
Second, Counters which maintains statistical information of flows consistent with match field.
Finally, Instructions which set of measures that are executed on streaming after matching.

Group Table

Each group table includes several entry groups. According to (Openflow switch specification, 2011) each entry of group table includes Group Identifier, which 32-bit unsigned integer that uniquely identifies groups and Type, which specifies the group semantics. "all", "select", "indirect" and "fast failover" are values that can be taken by the kind of group then Counters, which provides statistical data of the packs that are processed by groups and last Action Buckets, which includes a set of dependent measures and parameters.

If a pack is transferred to the group to select the measure, the following measures are taken depending on the type of group:

1. All: All measures introduced in action bucket are run. This is used for multicasting or broadcast.
2. Select: Based on the selection algorithm in the switch, a measure is selected and run.
3. Indirect: It allows the different flows and groups to refer to a common group.
4. Fast Fault End: In case of disconnection of the main path and lack of implementation of the first measure, this feature allows to select the next step without informing the controller.

If the match tables are not found in the entry, the measure at the table-miss entry is done by switch on the pack. It can be packet dropping, search in the next flow table or referral to the controller.

A Review of Previous Research

In this section, we will review previous research to increase fault tolerance in the traditional networks and software-based networks.

Fault Tolerance in Traditional Networks

In traditional networks, there are two approaches to deal with faults: Reactive and Proactive.

Reactive

In this method, alternative path is calculated dynamically after occurring fault (Rak, 2015). Fault Tolerance in this method consists of 4 stages of failure detection, failure announcement, calculation of a new path, updating the routing table. Efforts are taken to reduce the time of each of these steps in order to reduce the total time to repair:

To reduce the time of fault detection, the network different planes characteristics can be used, and in each of these planes ten fault can be detected. In (Wang et al, 2007) a method is introduced to check the health of the link using Ethernet protocol specification in the physical plane. In the presented method, the device sends heartbeats in the range of 16±8 milliseconds to check if there is any connection in case of no connection. If there is not any response in the range of 50-150 milliseconds, a connection is identified. The time interval of fault detection in this method is not suitable for SDN networks (vanet al, 2014).

In the linked plane, the fault can be detected using STP or RSTP protocols. But these protocols because of the large size of the detection window in the second period are considered as the slow methods (van et al, 2014).

In (Katz and Ward, 2010) a method is proposed a that can be used to check the path health or link. In BFD (Bidirectional Forwarding Detection), a mechanism for sending controlling messages and establishing a meeting is used to check activity of the path or link between two end nodes. In this method, a meeting should be made between two nodes to check the health of end-to-end connection between two nodes. In the time intervals specified, each node sends the status of its meeting to another node. The receiver node in response sends an echo message. Failure to receive the packs on each side is equivalent to the loss of end-to-end connection. BFD is not depended on any protocol and can be directly implemented on IP, MPLS and Ethernet. It is implemented on Open vSwitch by the TCP/UDP flow (van et al, 2014).

Timer can be used to avoid the declaration of transient faults (Bonaventure et al, 2007). To reduce the computation time of routing, routing algorithms can be improved. To update routing tables the batch updates can be used (Chen et al, 2015).

Proactive

In this method, the alternative transmission way is calculated before fault is occurred. Compared to a reactive approach, research shows that this approach in the face of fault will have less missing packs (Rak, 2015).
The proactive method of fault tolerance follows different patterns depending on the occurrence of fault in one or more links. The reactive methods focus more on routing algorithms.

Fault Tolerance in Software-based Networks

Emerging technologies with the advantages have challenges. OpenFlow has some features that are used in fault tolerance. Fault tolerance in data plane takes place in two stages of fault detection and repair.

Fault Detection

LOS (Loss Of Signal) is widely used in carrier-grade networks (Sharma et al, 2013). OpenFlow detects the fault by switching locally a port from proactive to reactive in a specific port of guiding devices. A fault message is announced to the controller. This method has the ability to detect several faults and it does not use data channel and message sending to detect.
In (Lee et al, 2014), he recommends a way to monitor limited multi-link status based on a cycle in the beginning and end of the controller. In this case, the control packs in the cycle will arrive to the controller passing the nodes and in case of any fault, it will be run in two stages. In this case, each node in addition to the control packet is sent to the next node it must send a respond to the back and controller. This method does not have the ability to detect several faults and can detect only the first fault. To detect fault, data is used. To send the back packs in the same direction of entrance, there is a need to keep the flow entries for the packs:
In (Sharma et al, 2013), DFD mechanism is used to detect the fault. In this method, a meeting is made between each pair of end nodes. Nodes in the time interval send the control packs. If you do not receive packs, fault in track is detected.
This method, if applied to a path with multiple links do not have the ability to detect the location of one or more faults and can only confirm that the path has encountered an fault. For each switch on the path at each time interval, a BFD control message will be sent. To detect fault, data transmission channel is used. The greater the length of the supervision path, more the time of fault detection.
In (van et al, 2014) DFD mechanism has been used for each link. If the number of nodes is more than O(n) rank the number of meetings per node for all paths will be O(n2) rank, and will have a lot of computational and communication soldier. In the proposed method, the meeting will be held for the links, then each node needs to hold O(n) meeting. According to the limitation of the course of monitoring period, the RTT time will be reduced and in combination with LOS method, fault detection time will be reached below 50 milliseconds. A control pack BFD will have up to 24 bytes for authentication, 8 bytes for UPD packs header, 20 bytes for IPv4 and 38 bytes for Ethernet header and totally 90 bytes are transferred per 1 milliseconds. The amount of overhead data equivalent to 0.067 and 0.0067 percent are created in 1 and 10 Gbps links, respectively.

Fault Recovery

In the fault recovery, methods are considered in two ways of retrieval and protection. In the following, we will present an overview of the research conducted in this manner.

In (Stevens et al, 2011) in examining retrieval method, the Shortest Path Algorithm has been selected to calculate a backup path. In this way after notifying the fault to the controller, the paths controller finds the nodes affected by the fault and then it selects for the existing network an alternative path by Shortest Path Algorithm. This way, if any path, can find it. The task of controller in this method is to calculate an alternative path and update switch flow tables. The procedure in case of malfunction of several links or switch malfunction can also be accountable. According to the results of this research, achieving the fault recovery less than 50 milliseconds on wide networks by the retrieval method will not be possible.
In (Li et al, 2014) a method to increase fault tolerance in data centers has been provided. Three features of data centers include high scalability, centralized management and network topology and using special protocols. Fat-tree architecture as one of the future data centers structures has a very high tolerability due to its added paths. In fat-tree architecture, switches of each plane are linked only to their upper and lower planes. If the top of the tree is closer, the communication links between the planes is increased. In the proposed method for fault tolerant, in case of fault, control messages are published in the entire network. In the event of a fault in the tree with depth K, maximum K-1 of control message will be sent to the controller. In this way the fault tolerance architecture has five components. Tables Initializer, Link Failure Detector, Traffic Statistic Collector, Path Calculator and Flow Table Updater. After the fault detection, the path calculator specifies an alternate path and announces to the switches by updater. Figure 3 presents network topology and proposed components of architecture.
This method can discover and manage any fault in some link or switch failure. It also allows you to take into account traffic load of each path in calculating an alternative path. A limitation of this method is the use of Fat-tree network topology.

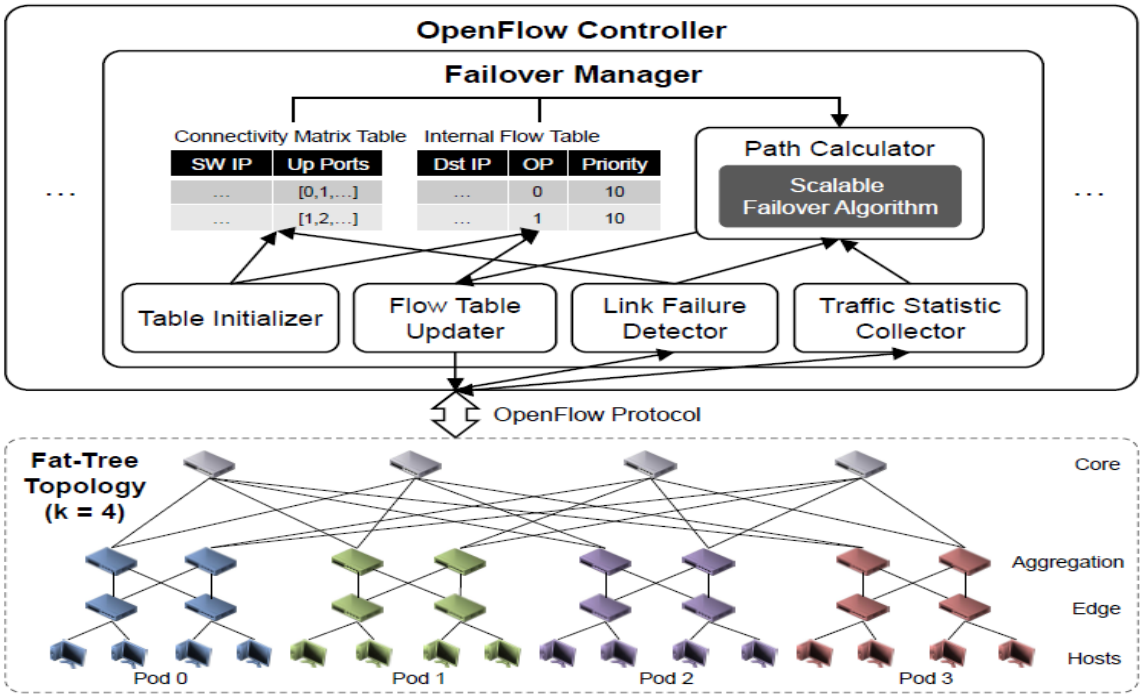

Figure 3: The proposed architecture in increasing fault tolerance in Fat-Tree network topology (Li et al, 2014)

(Sgambelluri et al, 2013) provides a way to protect using the OpenFlow properties. In addition to the match fields in the flow tables, counters and measures, priority levels and timers can be stored. In the proposed method, working path and backup path are simultaneously calculated and stored in the flow tables of switches with different priorities. The working path with high priority and backup path with low priority are stored. As long as the working path is not faulted, the switch navigates the flows to the working path. In the event of a fault, the switch eliminates the high priority entry with no need to controller and navigates packs through the backup path. In this method, in case of the right link, the switch sends a OFP_FLOW_RESTORE message to the controller to calculate the working path again.
The high speed of reaction in dealing with the fault is of benefits of this approach. If a fault occurs in several links and a number of faults are in working path and a number of them in backup path, this method can not be responsible.
In (Sharma et al, 2013) the protection method to increase fault tolerance is introduced using the concept of Group OpenFlow group table in version 1.1. In this method, the whole paths in the group table will fall with the measure of "fault fast end". After detection of any fault by the switch, other proactive paths will be replaced. The most important disadvantage of this method is need to a large number of entries for tables. In this method, if multiple links from different paths are faulted, this method cannot choose a backup path, and even if there is a more path, it is unable to find any path.
In (van et al, 2014) a method is proposed to increase fault tolerance that is a combination of and retrieval and protection methods. This method tries to reduce the weaknesses by using the strengths of each of the methods introduced. The method comprises the following steps: At first two primary and backup paths is calculated and stored in switches. In case of any fault detection, switch quickly sends packs through the backup path. It is necessary to communicate end to end. If the switch has not any backup path, the pack is returned to the previous path through the same path and the routing is performed by Crankback algorithm. Selecting a backup path is done by group table and selection of the type of “fault fast end”.
In the second stage, at the same time that the switches transmit packs through the backup path, the fault is reported to the controller and the controller calculated other optimal path and informs the switches. Fault detection is done by BFD meetings for each link.
BFD meetings for each link instead of each path leads to:

1. faster detection time because of RTT time reduction
2. network complexity reduction because of reduction of number of BFD meetings

If BFD meeting is established for the final pair of nodes, very large number of these meetings should be established. Because meetings are established for each link, the false positive detection caused by congestion in the network is low.
In this method, the overhead of BFD meetings is equal to the time interval of sent BFD meetings that is considered the upper limit of RTT. In this method, because BFD meeting is established for each link, this amount is very low. At each link, RTT is average of 0.5 milliseconds and the time interval of sent meeting is considered as 1 millisecond. This short time interval is considered as overhead, but if a meeting is established for each path, the number of meetings of each node is of O(N*N) rank, while the rank is equal to O(N) by selecting a meeting for each link. Each node will have a proactive meeting for the number of its links. A control pack BFD will have up to 24 bytes for authentication, 8 bytes for UPD packs header, 20 bytes for IPv4 and 38 bytes for Ethernet header and totally 90 bytes are transferred per 1 milliseconds. The amount of overhead data equivalent to 0.067 and 0.0067 percent are created in 1 and 10 Gbps links, respectively.

Discussion and Conclusion

Fault Tolerance in current networks is examined by two methods: reactive and proactive. In reactive method, the most important challenge is the time to repair and update the network. While this method guarantees to find the path, if any, in the proactive method, the fault fast repair is possible and there is lowest pack loss than a reactive method. Challenges of proactive methods of computational and capacity overhead are resulted from computing paths in network normal time and the impossibility of fault recovery in multiple links as of part of malfunction is in the working path and the other part in the backup path. While there is a path that has not been selected the proactive method cannot guarantee using it.

In the software-based networks, benchmark of computational overhead in the controller, the number of entries stored in the flow table of switches, finding an alternative path, if any, the ability to repair the malfunction in several links, affiliation to a particular topology, repair time and the lowest pack loss are the criteria to compare the algorithms.

The proposed recovery method in (Stevens et al, 2011) is computational overhead only in case of fault. Additional entries in the flow table of switches will not be stored. In case of failure, it finds several links of this way, if there is an alternative path. This method is not limited to a specific network topology but it has very much time compared to other methods. Method (Li et al, 2014) has the recovery time faster than other recovery methods due to the use of Fat-tree network topology of controller computational overhead in case of low fault and the number of optimal table entries, but if all the links in a branch with a higher level are broken, fault recovery is not possible.

(Sgambelluri et al, 2013) has high reaction speed in dealing with the fault, but there are calculation overhead of alternative methods in case of safe network. This method should save a lot of entries in switch while in case of fault in several links cannot choose an alternative path, even if there is a path. (Sharma et al, 2013) tries to improve evaluation criteria by utilizing the characteristics of different methods. Due to division of crisis in case of the health of the network, there is a need only for calculation of a backup link for each link in switches and in case of malfunction, calculation of optimal link. Entries stored in comparison with other methods and in case of fault in multiple links can calculate backup path. This method is the least amount of fault loss due to Crankback compared to other methods.

This paper compares the methods of the fault tolerance of current software-based networks.

References:

Bonaventure, O., Filsfils, C., & Francois, P. (2007). Achieving sub-50 milliseconds recovery upon BGP peering link failures. Networking, IEEE/ACM Transactions on, 15(5), 1123-1135.

Chen, J., Chen, J., Xu, F., Yin, M., & Zhang, W. (2015). When Software Defined Networks Meet Fault Tolerance: A Survey. In Algorithms and Architectures for Parallel Processing (pp. 351-368). Springer International Publishing.

Katz, D., & Ward, D. (2010). Bidirectional forwarding detection (BFD).

Kreutz, D., Ramos, F. M., Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S., & Uhlig, S. (2015). Software-defined networking: A comprehensive survey. Proceedings of the IEEE, 103(1), 14-76.

Lee, S. S., Li, K. Y., Chan, K. Y., Lai, G. H., & Chung, Y. C. (2014, April). Path layout planning and software based fast failure detection in survivable OpenFlow networks. In Design of Reliable Communication Networks (DRCN), 2014 10th International Conference on the (pp. 1-8). IEEE.

Li, J., Hyun, J., Yoo, J. H., Baik, S., & Hong, J. W. K. (2014, May). Scalable failover method for data center networks using OpenFlow. In Network Operations and Management Symposium (NOMS), 2014 IEEE (pp. 1-6). IEEE.

Openflow switch specification: version 1.1.0. Feburuary 2011.

Rak, J. (2015). Resilient Routing in Communication Networks. Springer.

Rowshanrad, S. Parsaei, M. R. & Keshtgari, M. (2015). implementing ndn using sdn: a review on methods and applications. IIUM Engineering Journal.

Sgambelluri, A., Giorgetti, A., Cugini, F., Paolucci, F., & Castoldi, P. (2013). OpenFlow-based segment protection in Ethernet networks. Journal of Optical Communications and Networking, 5(9), 1066-1075.

Sharma, S., Staessens, D., Colle, D., Pickavet, M., & Demeester, P. (2013). OpenFlow: Meeting carrier-grade recovery requirements. Computer Communications, 36(6), 656-665.

Staessens, D., Sharma, S., Colle, D., Pickavet, M., & Demeester, P. (2011, October). Software defined networking: Meeting carrier grade requirements. In Local & Metropolitan Area Networks (LANMAN), 2011 18th IEEE Workshop on (pp. 1-6). IEEE.

Sterbenz, J. P., Cetinkaya, E. K., Hameed, M. A., Jabbar, A., Qian, S., & Rohrer, J. P. (2013). Evaluation of network resilience, survivability, and disruption tolerance: analysis, topology generation, simulation, and experimentation. Telecommunication systems, 52(2), 705-736.

van Adrichem, N. L., Van Asten, B. J., & Kuipers, F. A. (2014, September). Fast recovery in software-defined networks (EWSDN), 2014 Third European Workshop on (pp. 61-66). IEEE.

Wang, L., Chang, R. F., Lin, E., & Yik, J. C. S. (2007). U.S. Patent No. 7,260,066. Washington, DC: U.S. Patent and Trademark Office.