2020-06-04

A Proactive-Restoration Technique for SDNs

Ali Malik
Technological University Dublin, ali.malik@tudublin.ie

Ruairí de Fréin
Technological University Dublin, ruairi.defrein@tudublin.ie

Recommended Citation

Malik, A. & De Frein, R. (2020). A proactive-restoration technique for SDNs. *The 25th IEEE Symposium on Computers and Communications (ISCC)*, Rennes, France, June. doi:https:10.21427/nrqj-8v82

A Proactive-Restoration Technique for SDNs

Ali Malik, Ruairí de Fréin
School of Electrical and Electronic Engineering
Technological University Dublin
Dublin, Ireland
{ali.malik, ruairi.defrein}@tudublin.ie

***Abstract*—Failure incidents result in temporarily preventing the network from delivering services properly. Such a deterioration in services called *service unavailability*. The traditional fault management techniques, i.e. protection and restoration, are inevitably concerned with service unavailability due to the convergence time that is required to achieve the recovery when a failure occurs. However, with the global view feature of software-defined networking a failure prediction is becoming attainable, which in turn reduces the service interruptions that originated by failures. In this paper, we propose a proactive restoration technique that reconfigure the vulnerable routes which are likely to be affected if the predicted failure indeed occurs. The proposed approach allocates the alternative routes based on the probability of failure. Experimental evaluation on real-world and synthetic topologies demonstrates that the network service availability can be improved with the proposed technique to reach up to 97%. Based on the obtained results, further directions are suggested towards achieving further advances in this research area.**

***Index Terms*—SDN, openflow, proactive, restoration, recovery, failure prediction, service availability.**

I. Introduction and Background

Due to its programmable interfaces, Software-Defined Networking (SDN) offers exciting opportunities for network designers to implement new routing strategies, customised traffic engineering, dynamic allocation of network resources and many other programmable functionalities [1]. Its adoption by leading companies such as Google, Microsoft and Huawei, who have employed SDNs in their data centers, underlines the positive perception of its performance capabilities. Currently, the concept of software-defined "everything"– SD-{Mobile [2], Storage [3], IoT [4], Blockchain [5], Cloud [6]}, getting broader to become an umbrella term for wide range of modern technologies. Furthermore, in the near future, SDN is expected to play crucial a part in 5G [7].

Unlike the traditional IP networks, in SDN, the control plane, i.e. controller, has been decoupled from the data plane to construct a centralised networking system whose forwarding elements are dump and commanded by the controller. Currently, the OpenFlow protocol [8] is commonly used to have the data plane governed by the controller.

In fact, communication networks are prone to either planned failures, e.g. maintenance, or unplanned failures, e.g. overload, bugs, cable cut and disaster situations [9]. Such failures have a negative impact on the network performance since it could harm some of its activities such as routing. In addition, failures could also cause a financial loss to the service providers, for instance, the financial losses of 28 cloud providers were estimated at approximately $285 million as a consequence of infrastructure and application failures for the period 2007-2013 [10]. Also, an evaluation of the susceptibility to link failure of business critical processes in a data-centre, which manages 75% of Europe's flight bookings, was undertaken in [11]. Hence, fault management and recovery is a very necessary requirement for networking systems to ensure their reliability and service availability. In general, failure recovery mechanisms of carrier-grade networks can be classified into two groups, these are: protection (proactive) and restoration (reactive) [12]. Yet, the concept of fault management in SDNs is inherited from the legacy networking systems, where the previous studies were focused on proactive, reactive and hybrid techniques [13]–[15]. On one hand, from SDN point of view, the proactive approach is a memory space consuming technique as backups will need to be pre-planned. Therefore, the Ternary Content Addressable Memory (TCAM) of forwarding elements will be affected by the additional loaded information of backups. Moreover, this technique is activated after the occurrence of failures, which means that it cannot protect the network from the service disruption and unavailability.

On the other hand, the reactive approach is time consuming. Given that SDN is a centralised networking system, thus, when a failure scenario occurs, three steps will need to be performed by controller. First, the failure detection phase by using some mechanisms like Loss of Signal (LoS). Second, the alternative route computation phase by using one of the routing strategies such as Dijkstra [16]. Third, the reconfiguration phase by updating the network through removing the old forwarding rules of faulty routes and installing new ones. This technique is slower than the former and also, it cannot protect the network from the service disruption and unavailability.

Obviously, the current recovery techniques are not capable of preventing network from being affected by service disruption and unavailability. Therefore, we propose a proactive-restoration technique that has the ability to reduce the service disruption and increase the network service availability through the use of online failure prediction mechanism, thanks to the global view of SDN.

This paper is organised as follows. In section II, we introduce our proposed method along with the network models and framework. Our simulation and experimental results are presented in Section III. Finally, we present our conclusions and outline our future work in Section IV.

II. Proposed Method and Framework

A. Online failure prediction

Most of the previous studies that dealt with fault management have succeeded in mitigating the failure impact such as the downtime, however, they remain far from avoiding its effects such as service unavailability. This issue stems from the delay of the convergence scheme that is involved in every recovery process. Such convergence includes three factors, these are: failure detection, alternative path planning and network update. For the purpose of enhancing the network service availability, an online failure prediction is utilised to avoid some failure incidents. In fact, the evaluation of failure prediction by itself is beyond the scope of this paper, as many studies with remarkable achievements have done so. Instead, we intend to employ the online failure prediction as a technique to realise the proactive-restoration technique. Fig. 1 illustrates a generic overview upon the online failure prediction process and the different time periods involved with it.

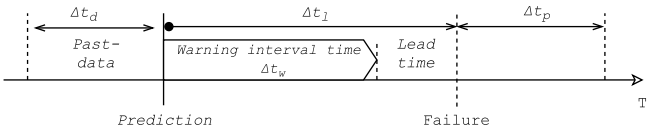


Fig. 1. Real-time failure prediction, [17].

According to Fig. 1, Δt_d represents the historical data used to forecast the upcoming failure incidents. Δt_l is the lead time in which a failure alarm is generated, which can be also defined as the minimum duration between the prediction and failure. Δt_w is the warning time in which the proactive-restoration solution will need to be activated. Δt_p represents the time, usually within few minutes, for which the prediction will be assumed to be true.

The quality of failure prediction is typically evaluated by two metrics: the unidentified incidents (FN) and the wrongly identified incidents (FP). While, the overall performance is measure by the precision, which is ratio of correctly identified incidents (TP) to the number of both correctly plus wrongly identified incidents, and recall, which is the ratio of correctly identified incidents to the total number of actual incidents.

B. Network model

The network is modeled as an undirected graph, which is widely used to model computer networks [18]. A graph G is defined as a combination of vertices and edges, $G = (V, E)$, where V represents the finite set of vertices (i.e. routers) in G that ranges over by $\{v_1, \ldots, v_n\}$ for $n \in \mathbb{N}$, while, E represents the finite set of bidirectional edges (i.e. links) that connect the vertices to one another. The set of all edges in a graph can be defined as a 2-element subset of vertices, $E \subseteq V \times V$. We define f_{ij} to be a demand flow traffic between the router i and j. We also define $flow$ to be a set of the demand traffic flows that have to be serviced by the network. A path p_{ij} is a route between two given routers, i.e. v_i and v_j, which is defined as a sequence of vertices $(v_i, \ldots, v_j)$. Pairs of vertices in a path are members of the edge-set, $(v_i, v_{i+1}) \in E, \forall 1 \leqslant i < j$. Here, v_i and v_j are called the *source* and *destination* routers of p_{ij}, respectively. We suppose that p_{ij} is a simple path if all of its routers are distinct. The length of p_{ij} is the total weights, i.e. cost, of its edges; that is $\omega(p_{ij}) = \sum_{v_i, v_{i+1} \in p_{ij}} \omega(v_i, v_{i+1})$, where the edge cost can be any additive metric such as hop count, delay and so on. The shortest path, $\hat{p}_{ij}$, from v_i to v_j is the path with minimal cost among all available paths from v_i to v_j. We define $\breve{p}_{ij}$ to be a shortest path instance with link dis-joint, which could serve as an alternative path that has no common edges with the former shortest path, such that $E(\breve{p}_{ij}) \cap E(\hat{p}_{ij}) = \emptyset$. We consider that p_{ij} is operational when all its edge pairs can be traversed. Hence, we define the following test operational function (δ) over a link, which reflects the current state of links as follows:

$$\delta(v_i, v_{i+1}) = \begin{cases} 1 & \text{the link is operational} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

We define $fail$ as the failed links set as follows:

$$fail = \{(v_i, v_{i+1}) \mid (v_i, v_{i+1}) \in E \wedge \delta(v_i, v_{i+1}) = 0\} \quad (2)$$

We define $predict$ to be a set that temporarily holds the risky flows whose constructed routes encompass at least one link with a certain level of failure probability, denoted T_Ω. In this paper, three different levels of T_Ω are considered; these are, low: when the probability of failure is less than 50%, med: when the probability of failure around 50% and high: when the probability of failure is over 50%.

C. Failure model

Due to the lack of a public network dataset that includes details such as failures, we have developed and used a failure model that generates failure events periodically. Two metrics are considered in this model: *mean-time between failure* (MTBF) and *mean-time to recover* (MTTR). These metrics are important for calculating the availability and reliability of each network repairable component [19]. MTBF is defined as the average time a particular component functions before it fails. It is calculated using the following equation:

$$\frac{\sum(start_{down\_time} - start_{up\_time})}{number\ of\ failures}. \quad (3)$$

MTTR is the average time required to repair a failed component. Each link is characterized by its own values of both MTBF and MTTR, and they are commonly independent from other components in the network. Metrics such as cable length, i.e. $\ell(v_i, v_{i+1})$, and Cable Cut (CC), can be used as alternatives for measuring the two availability metrics. According to [19], MTBF can be calculated as follows:

$$MTBF(hours) = \frac{CC \times 365 \times 24}{\ell(v_i, v_{i+1})} \quad (4)$$

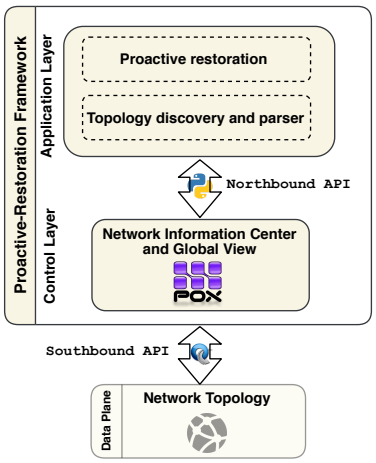


Fig. 2. Architecture of the proposed framework and its components: the primary contribution of this paper is on the proactive restoration routing block. OpenFlow is used on the southbound interface and POX Python APIs are used on the northbound interface.

For instance, when CC is equal to 100 km, it means that per 100 km there will be on average one cut per year. Besides this, the MTTR of a link is influenced by its length [20], which expresses the fact that the longer link has a higher MTTR value. On this basis, we have designed the following formula for calculating the MTTR value for each link in the network.

$$MTTR(hours) = \psi \cdot \ell(v_i, v_{i+1}) \tag{5}$$

The time (per kilometer) required to fix the cable, ψ, has units hour/kilometer format. Due to the fact that links are physically distributed in different locations and environments, ψ differs from one link to another. Even if some links have the same length, their ψ could be different as it relies on the physical location and the ambient conditions. A more comprehensive and detailed description of failure model is given in [21].

D. Framework design

From a high level point of view, Fig. 2 illustrates the main components of our proposed framework. In next, we discuss the components we used and developed in this framework in more detail.

1) SDN controller: The SDN controller represents the network's brain. It is where the intelligence and decision making is performed by the framework. We use the POX controller as it facilitates fast prototyping [22]. The standard OpenFlow protocol is used as a southbound API for establishing the communication between the data and control planes, whereas the set of POX APIs is used on the northbound interface for developing various network control applications.

2) Topology discovery and parser: This module is responsible for fetching the underlying network topology characteristics and building a topological view with the aid of the POX *OpenFlow discovery*[1]. In order to represent the network topology as a graph, G, we utilised the NetworkX [23] tool in order to be able to manipulate and simplify the underlying network topology.

[1] `https://github.com/noxrepo/pox/`

Algorithm 1: Proactive restoration

```
▷ Initialisation :
1  ∀f_ij ∈ flow, set p̂_ij as a primary path
▷ Proactive restoration :
2  foreach f_ij ∈ predict do
3      if Probability ⩾ T_Ω then
4          Replace each potential p̂_ij with p̆_ij
5          wait: Δt_p
6          if [ ∃(v_i, v_{i+1}) ∈ p̂_ij ∧ δ(v_i, v_{i+1}) = 0 ] then
7              Mark as: correctly identified
8          else
9              Mark as: wrongly identified
10             Replace each p̆_ij with p̂_ij
11         end
12     end
13 end
▷ Recovery :
14 foreach (v_i, v_{i+1}) ∈ fail do
15     Run Dijkstra's algorithm to find p̂_ij for every
         affected f_ij
16 end
```

3) Proactive restoration: In order to maintain the network flows, the proactive restoration technique is developed and illustrated in Algorithm 1. Firstly, the shortest path, based on Dijkstra's algorithm with complexity of $O(|V|+|E|\log|V|)$, is constructed for each flow (line 1). Then, based on links probability of failure, a disjoint path based on Bhandari's algorithm [24] with complexity of $O((K+1).|E|+|V|\log|V|)$, is configured for each flow exceeds T_Ω (line 2-4). After that, the reconfiguration is considered valid if the failure is appeared within the predefined Δt_p (line 5-7). Otherwise, the reconfiguration is considered invalid and this will incur another reconfiguration to set the former shortest path again (line 8-11). Finally, in the presence of link failure that was not predicted, an alternative path will be appointed based on Dijkstra's algorithm for every affected flow (line 14-16).

III. Performance Evaluation

A. Simulated network topologies

Both real-world and synthetic network topologies were used to construct the data plane layer and in order to allow us to evaluate the proposed technique. These topologies are depicted

(a) us [25] (b) german [25] (c) waxman

Fig. 3. Experimental topologies.

TABLE I
TOPOLOGY CHARACTERISTICS

Topology	Nodes	Edges	Type
us	26	42	real-world
german	50	88	real-world
waxman	70	140	synthetic

in Fig. 3 and their characteristics are detailed in Table I. We used the Internet topology generator Brite [26] to generate the synthetic topology based on Waxman model [27].

B. Experiment design

The proposed framework was implemented and evaluated by using the container-based emulator, Mininet [28]. As evidenced in the survey [29], Mininet is a widely used emulation system for emulating/simulating network architecture with various experimental scenarios as well as to evaluate and prototype SDN protocols and applications. The experiments of this paper were designed based on the out-of-band connection mode, where the control and data data are transmitted over a separated medium. In the emulation environment, we employed two servers; one acted as the OpenFlow controller and the other simulated the network topology. For each server, we used Ubuntu v.14.04 LTS with Intel Core-i5 CPU and 8 GB RAM. Based on the failure event model, Sec. II-C, the general reliability theory [30] has been applied to generate failure events using the exponential distribution ($mean = MTBF$) for the next time to failure of each link, and lognormal distribution $E(\mu, \sigma)$ with $\mu = \log(MTTR) - ((0.5) \times \log(1 + ((0.6 \times MTTR)^2 / MTTR^2)))$ and $\sigma = \sqrt{\log(1 + ((0.6 \times MTTR)^2 / MTTR^2}$ for time to recover. For failure anticipation, false and true positive have been generated during the simulated time using the uniform distribution following the designated threshold level.

To measure the network service availability, we first measure the unavailability. In general, the network service unavailability over a specific time can be arrived at:

$$U = \frac{\sum_{i=1}^{e} affected\ f_{ij}}{e\ \times\ \ell(flow)} \tag{6}$$

where, e represents the link failure incidents. To measure the unavailability of the proactive-restoration technique, denoted by U^+, the impact of recall is considered as follows:

$$U^+ = (1 - Recall) \times U \tag{7}$$

Consequently, the network availability is measured by subtracting the unavailability from 1.

Eventually, we measure the routing flaps, R_f, as a metric to reflect the impact of the proposed method on network's stability. In fact, the main goal of proactive-restoration technique is to enhance the network service availability, however, this would come at the price of network instability due to the unnecessary reconfiguration that would result from the wrongly identified incidents. The routing flaps of the proactive-restoration, denoted by R_f^+, can be arrived at:

$$R_f^+ = \sum TP + \sum FP + R_f^{sdn} \tag{8}$$

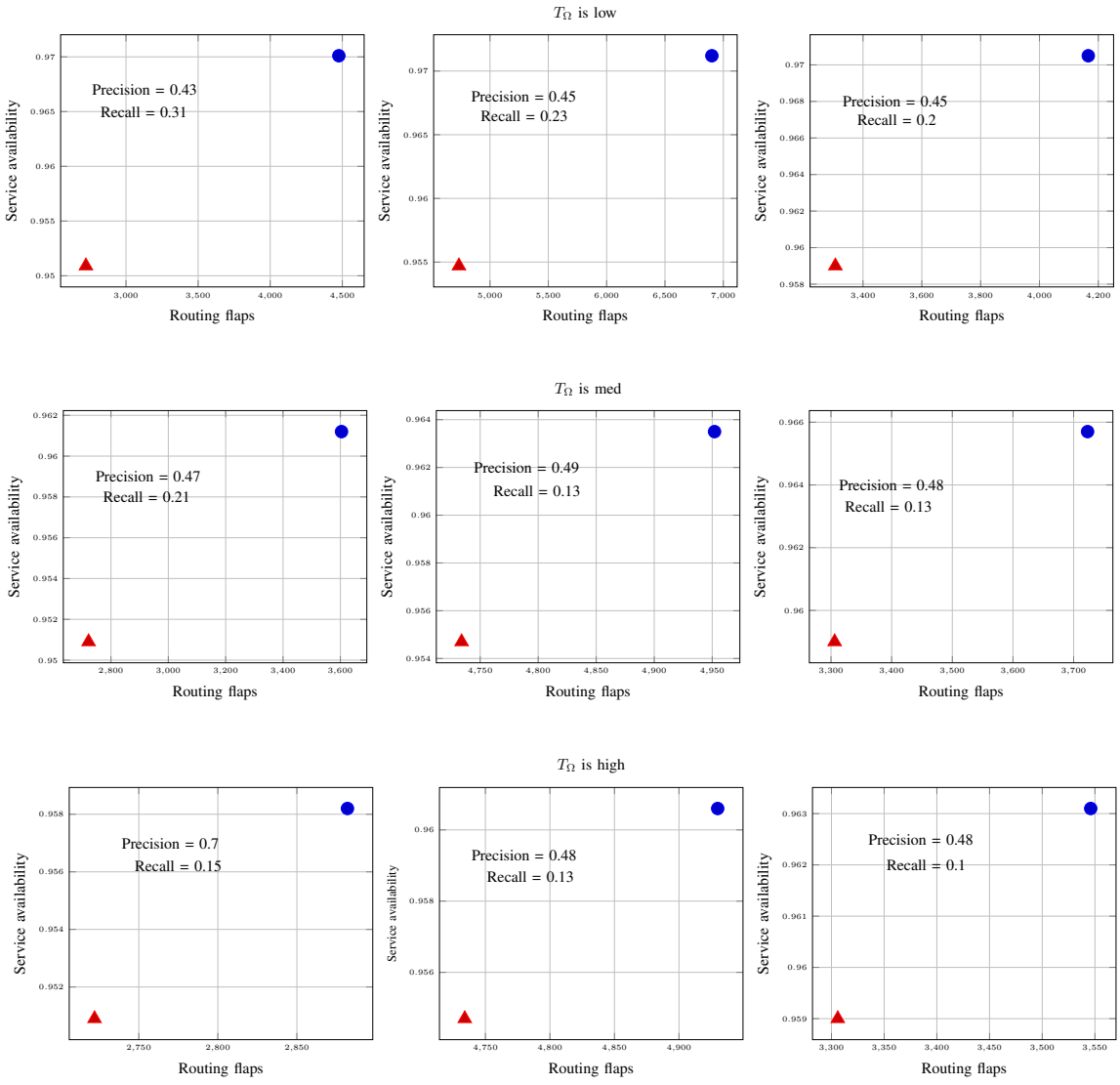
where R_f^{sdn} represents the routing flaps of the SDN network without using the proactive-restoration technique, which is measured by the number of times that the network is reconfigured due to up and down links incidents.

C. Simulation results

To evaluate our proposed method, each experimental topology was simulated in the system for 48 hours. Fig. 4 shows the performance of the network service availability with and without the use of the proactive-restoration method and at three different levels of T_Ω; namely low, med and high. It can be clearly seen that the network with proactive-restoration method presents a better service availability than the performance of network without using it. Despite the low recall and precision rates, there is still a gain in service availability. One can observe a gain in availability of 2% when T_Ω is low to achieve on average 97%, while, up to 1.5% when T_Ω is medium to gain on average 0.964% and almost 1% when T_Ω is high to obtain 0.96% on average. It can also be observed that the precision and recall are affected by T_Ω. The recall is inversely proportional to T_Ω. This is due to the fact that the amount of captured events is increased when the probability of failure is at low level. Therefore, the highest rate of recall can be obtained when the number of identified events is large. In contrast, precision decreases when recall increases since many FP have to be accepted.

Although the network service availability is improved, the rate of the routing flaps generated when use the proactive-restoration is found to be higher than the case of none using it. The routing instability, by means of unnecessary flaps, is correlated with the value of precision. The useless flaps that were generated during the simulation are measured here for each topology using the three threshold levels.

The amount of useless flaps is shown in Fig. 5, which shows the unnecessary flaps that have been reported based on the FP rate of each topology. It can be observed that the number of unnecessary flaps increases when T_Ω decreases. The largest amount of unnecessary flaps is reported when T_Ω is at low level, while, the smallest amount of unnecessary flaps is reported when T_Ω is at high level. Each FP event is associated with two useless flaps, that is, one for the reconfiguration, i.e. switch over to an alternative disjoint path, and the other for the reversion, i.e. switch back to the primary shortest path. Therefore, the performance of failure prediction plays an important role in reducing the quantity of unnecessary routing flaps. Thus, there is a trade-off to be made by the network operators in order to determine whether or not the gain in service availability using the approach of proactive restoration justifies the cost.



(g) us (h) german (i) waxman

Fig. 4. Routing flaps and service availability of different T_Ω measures based on parameter settings of $\Delta t_l = 120s$ and $\Delta t_p = 30s$. The (●) represents the result achieved when using the proactive-restoration technique, while the (▲) represents the results without the proactive-restoration.

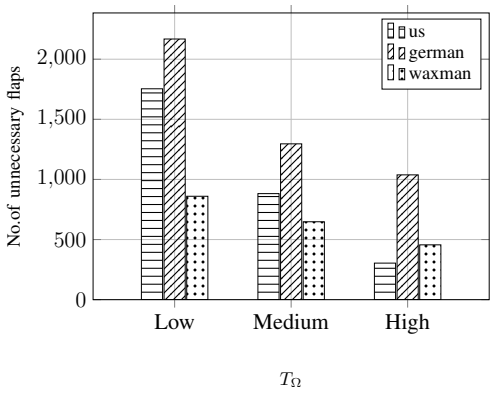


Fig. 5. The measurements of the unnecessary routing flaps that was reported over the three experimental topologies and by using three $T\Omega$ levels. It indicates that the high the $T\Omega$ the less flaps rate.

IV. Conclusions

This paper demonstrated the promise of using a online failure prediction mechanism to enhance the network service availability in SDN environment. Despite the fact that network fault management is a well established research area, the next-generation networks, like SDNs, still needs further investigation. We presented a new proactive-restoration algorithm in which the network controller will have a time window to reconfigure the network before the anticipated failure occurs and hence some potential interruption of service availability becomes avoidable. We demonstrated how the proposed method can be implemented. The performance of the proposed approach was tested and evaluated through simulation experiments on a real-world and synthetic network topologies. Two metrics were used to examine our approach, these are: service

availability and routing flaps. The experimental findings have shown the effectiveness of the proposed method in improving the SDN service availability and its ability to avoid some failure events before they take place. The service availability gain was up to 97% and it can be further improved when use a good prediction model. In closing, the obtained improvement came at the cost network stability through generating an additional routing flaps.

As part of our future work, we intend to incorporate machine learning techniques for more precise and realistic prediction. In addition, we will extend this study to consider multiple-failure scenarios that, for example, could result from disasters and cause severe disruption in service availability.

ACKNOWLEDGEMENT

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under the Grant Number 15/SIRG/3459.

REFERENCES

[1] T. Koponen, S. Shenker, H. Balakrishnan, N. Feamster, I. Ganichev, A. Ghodsi, P. Godfrey, N. McKeown, G. Parulkar, B. Raghavan *et al.*, "Architecting for innovation," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 3, pp. 24–36, 2011.

[2] M. Liyanage, A. Gurtov, and M. Ylianttila, *Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture*. John Wiley & Sons, 2015.

[3] J. Soumagne, R. Ross, G. Shipman, G. Amvrosiadis, N. Fortner, D. Robinson, P. Carns, M. Dorier, R. Latham, S. Snyder *et al.*, "Final technical report-a software defined storage approach to exascale storage services," The HDF Group, Tech. Rep., 2019.

[4] P. Mishra, D. Puthal, M. Tiwary, and S. P. Mohanty, "Software defined iot systems: Properties, state of the art, and future research," *IEEE Wireless Communications*, vol. 26, no. 6, pp. 64–71, 2019.

[5] J. Wu, M. Dong, K. Ota, J. Li, and W. Yang, "Application-aware consensus management for software-defined intelligent blockchain in iot," *IEEE Network*, vol. 34, no. 1, pp. 69–75, 2020.

[6] Y. Jararweh, M. Al-Ayyoub, E. Benkhelifa, M. Vouk, A. Rindos *et al.*, "Software defined cloud: Survey, system and evaluation," *Future Generation Computer Systems*, vol. 58, pp. 56–74, 2016.

[7] Z. Zaidi, V. Friderikos, Z. Yousaf, S. Fletcher, M. Dohler, and H. Aghvami, "Will sdn be part of 5g?" *IEEE Communications Surveys & Tutorials*, 2018.

[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[9] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and C. Diot, "Characterization of failures in an operational ip backbone network," *IEEE/ACM transactions on networking*, vol. 16, no. 4, pp. 749–762, 2008.

[10] C. Cérin, C. Coti, P. Delort, F. Diaz, M. Gagnaire, Q. Gaumer, N. Guillaume, J. Lous, S. Lubiarz, J. Raffaelli *et al.*, "Downtime statistics of current cloud solutions," *International Working Group on Cloud Computing Resiliency, Tech. Rep*, vol. 1, p. 2, 2013.

[11] R. de Fréin, J. Pfaff, and T. Paré, "Enterprise data center globality measurement," in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*. IEEE, 2015, pp. 1861–1869.

[12] J.-P. Vasseur, M. Pickavet, and P. Demeester, *Network recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. Elsevier, 2004.

[13] P. C. da Rocha Fonseca and E. S. Mota, "A survey on fault management in software-defined networks," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2284–2321, 2017.

[14] A. Rehman, R. L. Aguiar, and J. P. Barraca, "Fault-tolerance in the scope of software-defined networking (sdn)," *IEEE Access*, vol. 7, pp. 124 474–124 490, 2019.

[15] A. Malik, R. de Fréin, and B. Aziz, "Rapid restoration techniques for software-defined networks," *Applied Sciences*, vol. 10, no. 10, p. 3411, 2020.

[16] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[17] F. Salfner, M. Lenk, and M. Malek, "A survey of online failure prediction methods," *ACM Computing Surveys (CSUR)*, vol. 42, no. 3, pp. 1–42, 2010.

[18] K. Ramarao, "A simple variant of node connectivity is np-complete," *International journal of computer mathematics*, vol. 20, no. 3-4, pp. 245–251, 1986.

[19] S. De Maesschalck, D. Colle, I. Lievens, M. Pickavet, P. Demeester, C. Mauz, M. Jaeger, R. Inkret, B. Mikac, and J. Derkacz, "Pan-european optical transport networks: an availability-based comparison," *Photonic Network Communications*, vol. 5, no. 3, pp. 203–225, 2003.

[20] A. J. Gonzalez and B. E. Helvik, "Characterisation of router and link failure processes in uninett's ip backbone network," *International Journal of Space-Based and Situated Computing*, vol. 2, no. 1, pp. 3–11, 2012.

[21] A. Malik, B. Aziz, M. Adda, and C.-H. Ke, "Smart routing: towards proactive fault handling of software-defined networks," *Computer Networks*, vol. 170, p. 107104, 2020.

[22] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, "Advanced study of sdn/openflow controllers," in *Proceedings of the 9th central & eastern european software engineering conference in russia*. ACM, 2013, p. 1.

[23] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.

[24] R. Bhandari, *Survivable networks: algorithms for diverse routing*. Springer Science & Business Media, 1999.

[25] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "Sndlib 1.0—survivable network design library," *Networks: An International Journal*, vol. 55, no. 3, pp. 276–286, 2010.

[26] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: An approach to universal topology generation," in *MASCOTS 2001, Proceedings Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE, 2001, pp. 346–353.

[27] B. M. Waxman, "Routing of multipoint connections," *IEEE journal on selected areas in communications*, vol. 6, no. 9, pp. 1617–1622, 1988.

[28] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010, pp. 1–6.

[29] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2014.

[30] M. Ohring, *Reliability and failure of electronic materials and devices*. Elsevier, 1998.