Technological University Dublin

ARROW@TU Dublin

Conference papers

School of Electrical and Electronic Engineering

2021-07-01

Bayesian Adaptive Path Allocation Techniques for Intra-Datacenter Workloads

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

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

Chih-Heng Ke
National Quemoy University, Kinmen 892, Taiwan

See next page for additional authors

Follow this and additional works at: https://arrow.tudublin.ie/engscheleart

Part of the Computer and Systems Architecture Commons, and the Systems and Communications Commons

Recommended Citation

Malik, A. et al (2021) Bayesian Adaptive Path Allocation Techniques for Intra-Datacenter Workloads, *The 30th International Conference on Computer Communications and Networks (ICCCN 2021)*July 19 - July 22, 2021, Athens, Greece.

This Conference Paper is brought to you for free and open access by the School of Electrical and Electronic Engineering at ARROW@TU Dublin. It has been accepted for inclusion in Conference papers by an authorized administrator of ARROW@TU Dublin. For more information, please contact arrow.admin@tudublin.ie, aisling.coyne@tudublin.ie.

This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 4.0 License

Funder: Science Foundation Ireland (SFI)

Authors
Ali Malik, Ruairí de Fréin, Chih-Heng Ke, Hasanen Alyasiri, and Obinna Izima

This conference paper is available at ARROW@TU Dublin: https://arrow.tudublin.ie/engscheleart/331

Bayesian Adaptive Path Allocation Techniques for Intra-Datacenter Workloads

Ali Malik*, Ruairí de Fréin*, Chih-Heng Ke†, Hasanen Alyasiri‡, Obinna Izima*
*School of Electrical and Electronic Engineering, Technological University Dublin, Ireland
{ali.malik, ruairi.defrein, d17127923}@tudublin.ie
†Department of Computer Science and Information Engineering, National Quemoy University, Taiwan
smallko@gmail.com
‡Department of Computer Science, University of Kufa, Iraq
hasanen.alyasiri@uokufa.edu.iq

***Abstract*—Data center networks (DCNs) are the backbone of many cloud and Internet services. They are vulnerable to link failures, that occur on a daily basis, with a high frequency. Service disruption due to link failure may incur financial losses, compliance breaches and reputation damage. Performance metrics such as packet loss and routing flaps are negatively affected by these failure events. We propose a new Bayesian learning approach towards adaptive path allocation that aims to improve DCN performance by reducing both packet loss and routing flaps ratios. The proposed approach incorporates historical information about link failure and usage probabilities into its allocation procedure, and updates this information on-the-fly during DCN operational time. We evaluate the proposed framework using an experimental platform built with the POX controller and the Mininet emulator. Compared with a benchmark shortest path algorithm, the results show that the proposed methods perform better in terms of reducing the packet loss and routing flaps.**

***Index Terms*—SDN, Bayesian, datacenter, probability, traffic workloads.**

I. Introduction

Traffic demands in Data Center Networks (DCNs) pose challenges for resource and path allocation mechanisms. A key requirement for DCN-oriented enterprises is to deliver good Quality of Service (QoS) so that end-users are satisfied, and their profit margins are preserved and grown. Network incidents such as link failures have a negative impact on network performance as they impinge on functionalities such as routing which plays a crucial role in forming the substrate of the DCN. Link failures invoke routing flaps –the advertisement of destination networks via multiple different routes in quick succession– that negatively affect QoS metrics such as packet loss. The severity of the problem of infrastructure failures can be measured in terms of financial loss. The financial losses of 28 cloud providers were estimated to sum to approximately $285 million as a consequence of infrastructure and application failures for the period 2007-2013 [1]. The approach in [2] looked to measure the exposure of DCN operators to link failures by quantifying the vulnerability of different online transaction processing workloads to different failures, and assigning a *globality* score to each workload, in the context of a DCN, which managed 75% of Europe's flight bookings. We take a Bayesian approach [3] towards adaptive path allocation in DCNs; belief in the availability of the DCN substrate is updated as more and more link failure events and workloads are observed by the DCN.

Modern DCN topology design is dominated by tree-based structures, for example, fat-tree and clos networks [4]. A review of the architecture of production networks is discussed in [2]. These architectures include redundancy in the path set to improve the fault-tolerance of the networks. Despite this over-provisioning, the authors of [5] report that this redundancy is not fully effective in terms of reducing the impact of link failure incidents. We argue that over-provisioning is not enough when it is implemented without reference to (1) the workloads serviced by the DCN or (2) the probability of link failures on the available links. Instead we show that learning about the use and vulnerability of the network can be used to update "prior" information about the network's state, resulting in an updated "posterior" information view of the network. Path allocation in response to a Bayesian learning approach improves DCN performance.

Software-Defined Network (SDN) is a promising networking paradigm in which the control plane is physically decoupled from the data plane. SDN's centralized network management and network programmability paradigm simplifies the task of embedding learning algorithms in DCNs routing [6]. We exploit SDN to improve DCNs by including run-time topological statistics such as the probability of link failure and the probability of link usage in the mechanism that is used for allocation paths for intra-DCN workloads. The benefits of this approach in the context of packet loss rates are readily demonstrated: assigning links which are more likely to fail than other less likely to fail links, could increase the percentage of packet losses. This type of assignment will also increase the expected percentage of routing changes, i.e. flaps, as the system will need to perform a recovery process every time when a failure occurs. As far as the DCN manager is concerned, the DCN substrate is more reliable for enterprises whose viability depends on its dependability.

This paper is organised as follows. Section II, introduces various SDN-based DCNs techniques from the literature. In section III, we introduce our proposed framework. Simulation and experimental results are presented in Section IV. Finally, in Section V, we conclude and highlight future research.

II. Related work

Although previous studies provide important insights into how to develop SDN-based DCN routing strategies they do not take into consideration the topological information which is available as a consequence of having centralized view of the DCN. We posit that information about the probability of link failure and the probability of link usage have the potential to play a crucial role in keeping end-users of the DCN satisfied. To this end, we propose a new SDN-based routing strategy that aims to enhance DCN performance through by reducing the packet loss and the routing flaps. We discuss the related work under the following headings: (1) Packet loss reduction during convergence time and multi-path mechanisms for packet loss reduction; (2) resource reallocation methods that are achieved with minimum network reconfiguration; and finally, (3) SDN-based machine learning techniques towards reducing the probability of packet loss in DCNs.

The authors in [7] proposed SQR as a mechanism to reduce the packet loss rates in DCNs. SQR is concerned a particular set of packets, e.g. those that might be lost during the convergence time. Convergence time is the time required by the network substrate to amend a path in response to a link failure incident. In comparison a multi-path approach was introduced in [8]. The objective of the approach, namely FUSO, was speed and thus the multi-path approach sought to reduce the packet loss that was induced by failure, congestion or both. To this end, FUSO diverted the traffic of a path onto a better path when the current path was suspected to be lossy. In comparison with our approach, we incorporate historical probability of loss information into the recover procedure, whereas FUSO takes an most local-in-time view of the recovery problem.

Packet loss due to link congestion was the primary issue addressed by a number of SDN-based DCNs approaches [9]–[12]. The authors in [13] introduced DADR, a disaster aware dynamic routing algorithm for DCNs based on SDN. DADR reduces the rate of packet loss and improves the overall network utilisation in response to disaster warning signals. The authors in [14] proposed QNR as a SDN-based QoS-aware approach to reduce the resource reallocation in DCNs. The problem of resource reallocation with minimum network reconfiguration was investigated with a view to minimising packet loss and delay. The authors in [15] introduced a system that provides load-aware virtual networking service in SDN-based DCNs with the aim of reducing packet loss and keeping the network load balanced. The authors in [16] compared the equal cost multi-path (ECMP) routing with static routing and found that the packet loss could be reduced with ECMP. In this paper, the probability of link usage is incorporated into the routing decision process, an approach which resonates with the load-aware packet loss reduction approach in [15].

The readily available network data, due to the global view and centralised management of the SDN paradigm, motivates the application of Machine Learning techniques [17], [18]. Efforts in the pursuit of DCN performance improvement, which leverage machine learning are pushing at the boundaries of what was previously possible. The authors in [19] introduced DQL as a strategy to generate optimal routing paths in SDN-based DCNs. The authors demonstrated that DQL can reduce the average packet loss rate compared with ECMP routing. In the same context, the authors in [20] proposed a new SDN reinforcement learning method, called RL-Routing, to solve various traffic engineering issues. Instead of building a mathematical model, RL-Routing relies on the experience that can be obtained from the underlying network activities. Compared to the Open Shortest Path First (OSPF) algorithm, which is the touchstone of many deployments,experimental results show that RL-Routing can reduce the packet loss and the probability of re-transmission. We continue by introducing a framework for incorporating various run-time statistics into SDN optimization routines.

III. Encoding Observations in Path Allocation

We start by introducing notation to describe the DCN nodes and edges, the operational link set and finally DCN workloads. We then discuss the link failure model. Finally, we introduce a Bayesian framework for embedding learned observations about network state in an adaptive path allocation routine.

A. System Model

The network is modelled as an undirected graph, G, which consists of a combination of vertices and edges, $G = (V, E)$. The set $V = \{v_1, \ldots, v_n\}$ represents a finite set of vertices (OpenFlow switches) in G. The edge-set, E, represents the finite set of bidirectional edges (links) in G. The edge between v_i and v_j is $e_{_{ij}}$. A path p is a sequence of nodes $(v_s, \ldots, v_d)$; each pair of nodes is connected by an edge. The *source* and *destination* nodes of p are v_s and v_d respectively. We consider simple paths, which means that all the nodes in the path are distinct. The ijth edge has a cost associated with it, $C(e_{ij})$. The path cost, C_p, is the sum of the costs of the edges in the path, $C_p = \sum_{e_{ij} \in p} C(e_{ij})$. In current routing approaches, edge costs are frequently deterministic values which characterize how favourable usage of the link is, for example, the cost captures its bandwidth or a hop counts. Our contribution lies in introducing a coherent framework for embedding belief about network state in these costs and updating these costs as time progresses. A path is operational when all its edge pairs can be traversed. The state of a link is represented as being in one of two states at time t,

$$\Psi_t(e_{ij}) = \begin{cases} 1, & \text{if the link is operational} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The shortest path from v_s to v_d is the path which has the minimum cost given that all links are operational.

The workload serviced by a DCN consists of a set of traffic flows or paths, $W = \{p_1, p_2 \ldots\}$. DCNs support some workloads which reoccur periodically. Some examples include database updates, the Map and Reduce of steps MapReduce, and also periodic queries due to DCN and application monitoring functions. It follows that some of the load patterns serviced by the DCN are structured. Similarly, workloads due to user requests can be structured; the types of applications serving requests reflect the phase of the user's day.

Failure Model: We use the link failure model that is presented in [21] in order to generate failure events periodically. Some DCN characteristics have been considered to make the adopted model more appropriate. First, the links length are set to be in the range of few hundred meters based on the recommendations of [22]. Second, only core links are considered to be possible subjects of link failure as they have the highest probability of failure [5].

B. Bayesian dynamic path allocation

The cost assignment approach outlined above, may adversely affect traffic that is delivered over the DCN. It does not express the idea that certain links experience failures more frequently than others. In addition, making routing decisions without considering the periodic nature of some workloads may lead to network instability and performance degradation. What is missing is a mechanism for incorporating sources of information into the route learning procedure adaptively.

Many conventional routing algorithms assign the hop count as the link cost. This can be interpreted as saying that all links in the network are equally likely to experience link-failure. Least cost routing in this scenario corresponds to choosing paths through the DCN with the lowest sum of link likelihoods of failure. As the likelihood of the link being a favourable choice is equal for all links, the DCN is represented as an adjacency matrix $\mathbf{L} \in \mathbb{I}^{n \times n}$, where the entries corresponding to links are drawn from the binary values, $\mathbb{I} = \{0, 1\}$.

To fix ideas, consider a 6 node DCN consisting of the vertex set $V = \{a, b, c, d, e, f\}$. A DCN's adjacency matrix has entries corresponding to each operational edge. If the hop count is the cost on each link, this is similar to saying that the likelihood of link failure of each link is equal,

$$\mathbf{L} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}. \tag{2}$$

In SDN-based DCNs, the network controller determines the flow tables for the selected routing algorithm, by running Dijkstra's Algorithm [23] on the link costs, which are captured here in $\mathbf{L}$. The learned routing topology is distributed to OpenFlow switches in the form of flow entries. When path recovery is required, the routing algorithm modifies the flow tables in the relevant switches. In the case where all links have an equal probability of failure, we can express this adjacency matrix as a likelihood of link failure matrix, $\hat{\mathbf{L}}$, by scaling the entries by the entry-sum of $\mathbf{L}$, which we denote as $|\mathbf{L}|$ which produces the following matrix:

$$\hat{\mathbf{L}} = \frac{1}{|\mathbf{L}|}\mathbf{L} \tag{3}$$

The probability of link failure of on link $e_{i,j}$, $p_{ij}(\theta)$, the prior, is computed by taking the ratio of failure events with the total number of events. An event in this case is an attempt to send traffic down a link, where the link may or may not be operational. The likelihood of failure on link $e_{i,j}$ given information about the probability of failure, θ, is $p_{i,j}(y = failure|\theta) = \mathbf{L}_{ij}$. Every time that we send traffic down a link, we can update the uncertainty that the link is operational or not, using Bayes rule, $p_{i,j}(\theta|y)$, which gives us the posterior probability of failure. Even without knowing what the frequency of failure is initially, by running traffic down links we can update the posterior probability of failure on a link given this sequence of observations using Bayes Law,

$$p_{i,j}(\theta|y) = \frac{p(y|\theta)p(\theta)}{\int p(y|\theta)p(\theta)d\theta} \tag{4}$$

Each time a link experiences failure we update the posterior probability $p_{i,j}(\theta|y)$. Each of the link probabilities is bounded by zero and one, $0 \leqslant \hat{\mathbf{L}}_{ij} \leqslant 1$ due to the normalization term, the integral $\int p(y|\theta)p(\theta)d\theta$. Routing using hop-counts, where each link has a weight of one, is the same as a routing solution which assumes that the link failure likelihoods on all links are equal. It is unreasonable to assume that networks are composed of links which have a uniform likelihood of failure. Coupled with this, we would like to have the flexibility to express the idea that links can have different frequencies of link failure, and to have the ability to be able to update our uncertainty about the health of links as we observe the DCN over time. If the link failure probabilities are not equal this effects the solution arrived at by Dijkstra's algorithm. For example, consider the following link failure probability matrix:

$$\hat{\mathbf{L}(t)} = \frac{1}{20}\begin{bmatrix} 0 & \frac{3}{2} & 0 & \frac{1}{2} & 1 & 0 \\ \frac{3}{2} & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ \frac{1}{2} & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \tag{5}$$

which indicates that the link from node a to b is more likely to fail than any other link. Conversely, the link from a to d is more reliable than the other links. Using this likelihood of failure matrix instead of the likelihood of failure matrix where all links are equally likely to fail is likely to give a different initial Dijkstra's algorithm solution to the uniform link failure matrix. We have indicated a time index t in order to show that the matrix is changing with time.

We also extend the flexibility of our approach to be able to account for the different types of workloads serviced by the DCN. The likelihood of failure of a link depends on the frequency of usage of the link. Given that the DCN workloads might involve some repetitive patterns, if we can identify what these patterns are, it is useful to embed this information in the uncertainty updating process. Consider a data-centre that supports two workload patterns, $\mathbf{W_1}, \mathbf{W_2} \in \mathbf{W}$:

$$\mathbf{W}_1 : \mathtt{a} \rightarrow \mathtt{d},\ \mathtt{e} \rightarrow \mathtt{c},\ \mathtt{d} \rightarrow \mathtt{e},\ \mathtt{c} \rightarrow \mathtt{f}$$
$$\mathbf{W}_2 : \mathtt{b} \rightarrow \mathtt{d},\ \mathtt{c} \rightarrow \mathtt{b},\ \mathtt{d} \rightarrow \mathtt{b},\ \mathtt{b} \rightarrow \mathtt{e},\ \mathtt{b} \rightarrow \mathtt{c},\ \mathtt{e} \rightarrow \mathtt{b}$$

where the operator $\rightarrow$ indicates that traffic traverses the link separated by the two nodes. The frequency of usage of the links associated with these workloads is defined as:

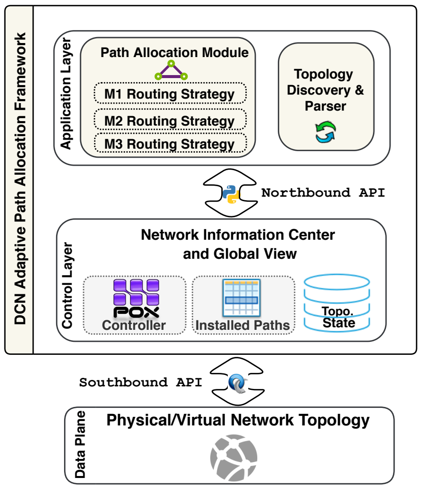

Fig. 1. Architecture of the proposed framework and its components: the primary contribution is on the path allocation module block. Openflow is used on the southbound interface and POX APIs are used on the northbound interface.

$$\mathbf{W}_1 = \frac{1}{|\mathbf{W}_1|}\begin{bmatrix}0&0&0&1&0&0\\0&0&0&0&0&0\\0&0&0&0&1&1\\1&0&0&0&1&0\\0&0&1&1&0&0\\0&0&1&0&0&0\end{bmatrix},$$
$$\mathbf{W}_2 = \frac{1}{|\mathbf{W}_2|}\begin{bmatrix}0&0&0&0&0&0\\0&0&1&1&1&0\\0&1&0&0&0&0\\0&1&0&0&0&0\\0&1&0&0&0&0\\0&0&0&0&0&0\end{bmatrix} \tag{6}$$

We contribute a new framework that runs Dijkstra's algorithm on a set of costs which are updated using Bayes law, using information about the link failure rates in the network and the frequency of usage patterns.

C. The proposed framework

The architecture of the proposed framework is illustrated in Fig. 1, which encompasses the following components:
SDN controller: The SDN controller is considered the brain of the network where the intelligence and decision making resided. Due to its fast prototyping, the POX controller is used in this framework [24]. The standard OpenFlow protocol [25] is employed as a southbound API for establishing the communication between the data and control planes. The POX API set is used on the northbound interface for developing the network applications.
Topology discovery: This component is responsible for learning the underlying network topology information to be delivered to the path allocation module. To do so, the standard POX discovery module[1] is used. In order to represent network topology as a graph, G, we utilised the NetworkX [26] for manipulating and simplifying the underlying graph topology.

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

Path allocation: In order to allocate the appropriate resources to the network workloads, three routing strategies with different QoS views are considered here. First, we developed M1 as a routing method that serves as a reasonable benchmark algorithm to compare with our proposed methods. M1 runs Dijkstra's algorithm to find the shortest path with the minimum number of hops for new requests. This strategy is in line with Eqn. 2, where all links are of identical cost, i.e. $\omega_{(e_{ij})} = 1$. When a link failure event, f, occurs, each path whose edge sequence involves the failed link is excluded from the installed paths set $\mathcal{P}$ and therefore, the subsequent flow packets of the affected workload will be treated as new request. The pseudo-code of M1 is given in Alg. 1.

Algorithm 1: Shortest Path With Dijkstra Cost (M1)

```
▷ Initialisation :
1 $\forall(e_{ij}) \in E$, $\omega_{(e_{ij})} = 1$
▷ Path Allocation:
2 foreach new pkt do
3     Run Dijkstra's algorithm to find the shortest path with link
        costs $\{\omega_{(e_{ij})}\}$
4 end
5 foreach link failure (f) do
6     Remove each path includes the failed link from $\mathcal{P}$
7 end
```

In contrast to M1, we propose M2 as a routing strategy in which the probability of failure, $\mathbb{P}$, is used as a metric to quantify the cost that can be arrived at:

$$\mathbb{P}_{e_{ij},(t)} = \frac{f_{e_{ij}}}{\sum_{e_{ij}\in E} f_{e_{ij}}} \times 100 \tag{7}$$

This could aid in dynamically allocating appliances which are less likely to fail and also to break the ties when there are identical costs. The strategy of M2 is in line with Eqn. 5. In the case of link failure, the same procedure as M1 is applied in addition to a step that updates the link's failure information based on Eqn. 7. The pseudo-code of M2 is demonstrated in Alg. 2.

Algorithm 2: Dijkstra-Like to Find Path With Minimum Failure Probability (M2)

```
▷ Initialisation :
1 $\forall(e_{ij}) \in E$, $\mathbb{P}_{(e_{ij})} = 1$
▷ Path Allocation:
2 foreach new pkt do
3     Run Dijkstra's algorithm to find the shortest path with link
        costs $\{\mathbb{P}_{(e_{ij})}\}$
4 end
5 foreach link failure (f) do
6     Remove each path includes the failed link from $\mathcal{P}$
7     $\forall(e_{ij}) \in E$, update $\mathbb{P}_{(e_{ij})}$
8 end
```

Finally, we present M3 as an extension to M2 in which the information of link usage probability, $\mathbb{U}$, is exploited along with $\mathbb{P}$ to shape the cost. The value of $\mathbb{U}$ can be arrived at:

$$U_{e_{ij},(t)} = \frac{|\delta_p(e_{ij})|}{|\mathcal{P}|} \times 100 \tag{8}$$

where $\delta_p(e_{ij})$ denotes the number of routes that include the link e_{ij} in their sequence. The M3 strategy is an embodiment of Eqn. 6. The information of the probability of links usage is updated based on Eqn. 8 every time when a new path is computed. At the moment of link failure incident, M3 recalls the same procedure of M1 plus updating both the probability of link failure information based on Eqn. 7. The pseudo-code for M3 is demonstrated in Alg.3.

Algorithm 3: Dijkstra-Like to Find Path With Minimum Failure and Usage Probabilities (M3)

```
▷ Initialisation :
$\forall(e_{ij}) \in E$, $\mathbb{P}_{(e_{ij})} = 1$
$\forall(e_{ij}) \in E$, $\mathbb{U}_{(e_{ij})} = 1$
▷ Path Allocation:
foreach new pkt do
    $\forall(e_{ij}) \in E$, update $\mathbb{U}_{(e_{ij})}$
    Run Dijkstra's algorithm to find the shortest path with link
      costs $\{\mathbb{P}_{(e_{ij})} \times \mathbb{U}_{(e_{ij})}\}$
end
foreach link failure (f) do
    Remove the affected installed paths from $\mathcal{P}$
    $\forall(e_{ij}) \in E$, update $\mathbb{P}_{(e_{ij})}$
end
```

It is worth mentioning that the necessary flow entries of the computed paths are installed with permanent `hard` and `idle` timeouts. This means that the forwarding rules will be removed only if its belonging path affected by a failure incident. The performance of M1, M2 and M3 will be evaluated and discussed in the next section.

IV. Performance Evaluation

In order to gauge the performance of the methods presented in Sec. III-C, three metrics were considered: (1) packet loss, (2) routing flaps and (3) path length. The packet loss indicates the dropped packet percentage and reflects the quality of the selected paths in terms of packet delivery. Routing flaps indicate the number of re-configurations and reflects the quality of the selected paths in terms of stability. The path length indicates the number of hops associated with the computed path and reflects the quality of selected paths in terms of shortness. Also, we considered a fat-tree with k=4 pods, which depicted in Fig. 2, as the DCN experimental topology in this evaluation. The Distributed Internet Traffic Generator (D-ITG) [27] was used to generate network workloads. It is a platform for producing a realistic packet-based network traffic by accurately emulating the workload of real world traffic and current Internet applications. D-ITG supports several modes of traffic generation, e.g. single-flow, multiple-flow and daemon, and it was shown to be more reliable than the existing traffic generators [28].

A. Experiment design

In this work, we define the workload as the number of requests/loads/demands that need to be processed by the DCN at a time t. In other words, the workloads are represented by a temporal sequence of traffic matrices, where each matrix is the demand during a certain time interval. The dynamic traffic matrix, T, is represented as follow:

$$T_{sd}(t) = \chi_{sd}(t) + \mu_{sd}(t) \tag{9}$$

where $\chi_{sd}(t)$ represents the mean traffic volume from s to d, periodic of period $T = 24h$ and $\mu_{sd}(t)$ represents the zero-mean random variable modeling random fluctuations of traffic volumes. The random fluctuation between the network pairs realised by the zero-mean normal random variable as described in [29], where the standard deviation of such fluctuation σ and the mean traffic demands can be arrived through the following power law:

$$\bar{x}_{sd}(t) = \psi\sigma_{sd}^{\gamma} \tag{10}$$

where we choose the values of $\gamma = 0.8$ and $log\psi = -0.33$ to fit the distribution. We used the fast network simulation setup (FNSS) tool [30] for modeling network traffic over a short period (up to 1.5 hour) by generating 3 stationary sequences traffic matrices. In the generated traffic matrices, we selected the servers of `pods, 0, 1` to be the traffic sources. The remaining servers were set to be the traffic destinations. Fig. 3 shows the number of the generated workloads with its frequencies over the three traffic matrices. It can be observed that some of the workloads appear just once, while, some appear more than once , i.e. twice or three times, to constitute a repetitive pattern. We deployed a D-ITG transmitter onto the terminal of each sender server and a D-ITG receiver on each receiver server that attached to `pods, 2,3`. In essence, D-ITG is employed to accurately replicate the workload of the 3 traffic matrices obtained by FNSS. The packets of the generated workloads following the Internet Control Message Protocol (ICMP). The size of the packets is 50 bytes and we use a constant inter-departure time between packets. The duration of each workload is described by its traffic matrix volume. We set the capacities of the links to 1000 Mbps, the link propagation delays to 100ms and the switches buffer size to 50 packets. Based on the failure model, link failure events were generated periodically during the simulation time. Finally, the proposed framework was implemented and evaluated by using the container-based emulator, Mininet [31]. As evidenced in the survey [32], Mininet is a widely used emulation system for

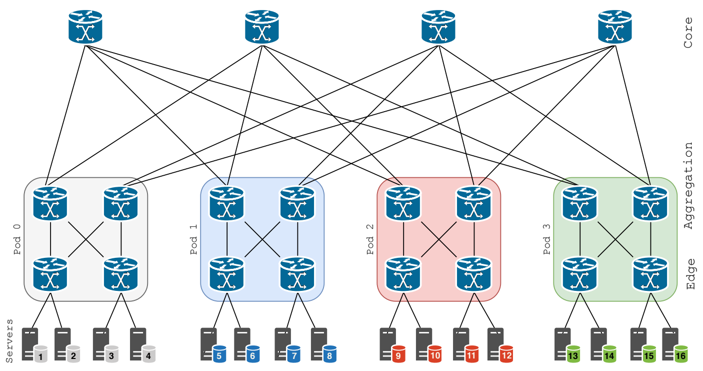

Fig. 2. Fat tree with $k = 4$, $|E| = 32$, $|V| = 20$, Servers= 16.

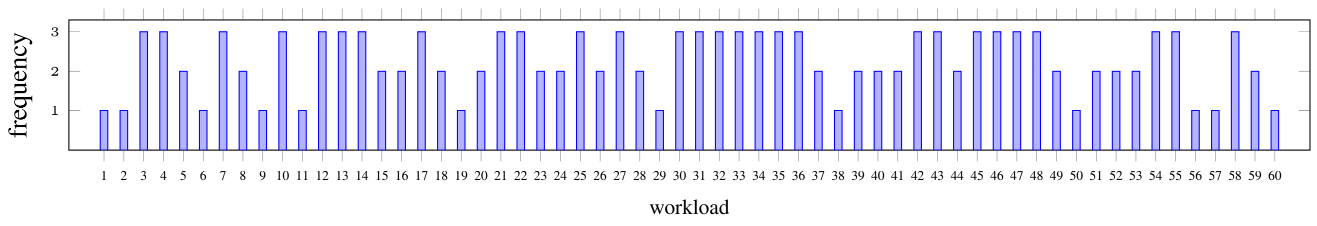

Fig. 3. The frequency of network workloads over the 3 generated traffic matrices.

emulating/simulating network architecture with various experimental scenarios as well as to evaluate and prototype SDN protocols and applications. 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.

B. Simulation results

Various experiments were conducted to study the performance of the proposed methods. We discuss the effect of the proposed methods on packet loss, routing flaps and path length.

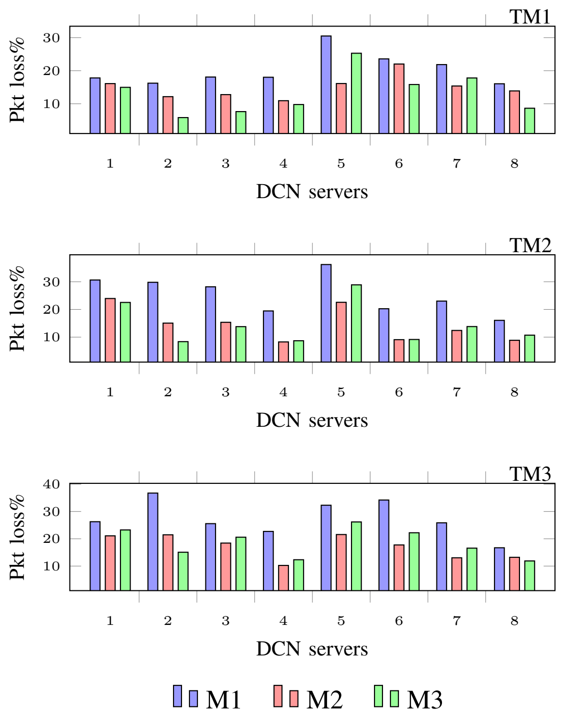

Fig. 4. Packet loss measure over the three traffic matrices with methods, M1, M2 and M3. This measurement is reported via the DCN sender servers. The packet loss percentages of this measure does not reflect the whole picture as the percentage is broken down into a respective traffic matrix.

1) Packet loss measure: Packet loss usually occurs due to a variety of reasons such as link failure events and congestion. In this experiment, we focus on measuring the packet loss percentage that results due to link failure causes. To do so, we first measure the packet loss rate over the three traffic matrices, i.e. TM1, TM2 and TM3, individually.

Fig. 4 shows the different packet loss rates that are experienced by M1, M2 and M3, which we derived from the server's log of `pods, 0,1`. It can be clearly inferred from the results that M2 and M3 perform better than M1. Also, one can notice that the performance of M2 and M3 are vary over the three matrices. It is conceivable that, sometimes, the packet loss rate with this type of individual measurement seems to be high especially when the affected workload is of marginal volume. Therefore, it is necessary to measure the overall packet loss percentage over the whole traffic matrix in order to gain an inclusive insight. Fig. 5 shows the total packet loss that gained from the server's log of `pods, 2,3`.

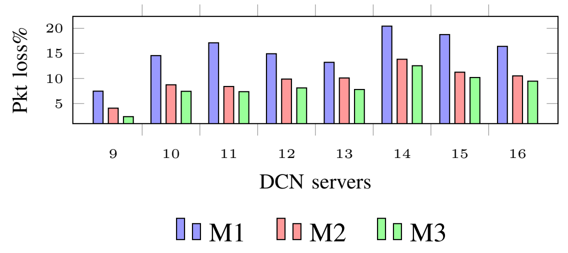

Fig. 5. Overall packet loss measure with methods, M1, M2 and M3. This measure shows the total loss percentages reported by the receivers during the three traffic matrices.

Now, we can conclude that M3 performs better than M2 in terms of packet loss rate. This is because M3 relies on two metrics, i.e. failure and usage probabilities, therefore, it is highly likely that the workloads will be distributed over diverse paths in order to satisfy these two metrics.

2) Routing flaps measure: Routing changes are only observed as a consequence of failure incidents. The number of routing oscillations is correlated with the number of times that a path allocation method is invoked. Operators are sensitive to the network stability, hence, it is important to measure the routing flaps associated with each method. Fig. 6 shows the routing flaps reported over the duration of the simulated traffic matrix. On one hand, it can be observed that M3 achieves the lowest flaps, compared to M1 and M2, and this justifies its effectiveness in reducing the rate of packet loss. On the other hand, M2 performs better than M1 as the later has the highest rate of routing flaps. In summary, the proposed methods can play a positive role in reducing the flaps rate and to stabilise the network.

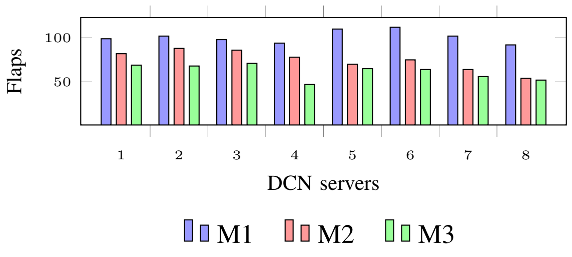

Fig. 6. Routing changes measurements over the three methods, M1, M2 and M3. This measurements is important for the network administrators to check the number of routing oscillation whether it is impacting the network stability.

3) Shortest path measure: The performance of the proposed approach was also studied with respect to the hop counts metric. When an OpenFlow controller detects a link failure status, the failed path is first identified. This step is followed by installation of the backup path that detours the disrupted flow packets, to achieve network recovery. This process could result in a new path length, which might be longer in terms of hop count than the replaced path. Fig. 7 shows the average hop counts of paths that have been established using the three methods during the experimental evaluation. In general,

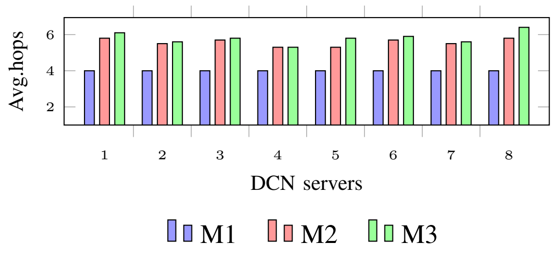

Fig. 7. Average path length measurements over the three experimental methods, M1, M2 and M3.

M1 paths have the lowest number of hops compared to the proposed methods. In fact, M1 produces the optimal paths in terms of the hops count. However, M3 paths have the greater number of hops among the others. Since M2 and M3 do not only considers the hop count metric, but also the probability of link failure and the usage frequency, we conclude that a disadvantage of securing the shortest path is that this may come at the cost of minimising the rate of both packet loss and routing flaps.

V. Conclusion

This paper demonstrated the benefit of using the SDN to enhance the performance of DCNs. Based on the global view of the controller, we presented new SDN-based Bayesian approaches, i.e. M2 and M3, that aim to mitigate the performance degradation that would result due to link failure incidents. In M2, the information of link failure probability was utilised. While, in M3, the information of both link failure probability and the frequency of link usage were used. We demonstrated how the proposed methods can be implemented. The performance of the proposed approach was tested and evaluated through simulation experiments on a fat-tree topology with $k = 4$. Two performance indicators: packet loss and routing flaps, were considered to gauge the performance of the proposed methods. The experimental findings shows the effectiveness of M2 and M3 in reducing the packet loss as well as the routing flaps. The experimental results also revealed that there is a trade-off between the achieved enhancement and the path length as the proposed methods yielded higher hop counts compared to the conventional shortest path algorithm. As part of our future work, we intend to explore the use of another Machine Learning models in order to enrich the intelligence used by the SDN controller by learning the latent factors that determine the probabilities of link failures and link usage.

Acknowledgment

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] M. Gagnaire, F. Diaz, C. Coti, C. Cerin, K. Shiozaki, Y. Xu, P. Delort, J.-P. Smets, J. Le Lous, S. Lubiarz *et al.*, "Downtime statistics of current cloud solutions," *International Working Group on Cloud Computing Resiliency, Tech. Rep*, 2012.

[2] 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.

[3] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian Data Analysis*, 2nd ed. Chapman and Hall/CRC, 2004.

[4] Y. Liu, J. K. Muppala, M. Veeraraghavan, D. Lin, and M. Hamdi, *Data center networks: Topologies, architectures and fault-tolerance characteristics*. Springer Science & Business Media, 2013.

[5] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," in *Proceedings of the ACM SIGCOMM 2011 conference*, 2011, pp. 350–361.

[6] W. Xia, P. Zhao, Y. Wen, and H. Xie, "A survey on data center networking (dcn): Infrastructure and operations," *IEEE communications surveys & tutorials*, vol. 19, no. 1, pp. 640–656, 2016.

[7] T. Qu, R. Joshi, M. C. Chan, B. Leong, D. Guo, and Z. Liu, "Sqr: In-network packet loss recovery from link failures for highly reliable datacenter networks," in *2019 IEEE 27th International Conference on Network Protocols (ICNP)*. IEEE, 2019, pp. 1–12.

[8] G. Chen, Y. Lu, Y. Meng, B. Li, K. Tan, D. Pei, P. Cheng, L. Luo, Y. Xiong, X. Wang *et al.*, "Fuso: Fast multi-path loss recovery for data center networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 3, pp. 1376–1389, 2018.

[9] X. Zeng, D. Wang, S. Han, W. Yao, Z. Wang, and R. Chen, "An effective load balance using link bandwidth for sdn-based data centers," in *International Conference on Artificial Intelligence and Security*. Springer, 2019, pp. 256–265.

[10] H. A. Khosravi and M. R. Khayyambashi, "Load-aware virtual network service over a software defined data center network," in *7'th International Symposium on Telecommunications (IST'2014)*. IEEE, 2014, pp. 623–628.

[11] T. Zhu, F. Wang, Y. Hua, D. Feng, Y. Wan, Q. Shi, and Y. Xie, "Mctcp: Congestion-aware and robust multicast tcp in software-defined networks," in *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*. IEEE, 2016, pp. 1–10.

[12] J. Hao, Y. Shi, H. Sun, M. Sheng, and J. Li, "Rerouting based congestion control in data center networks," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2019, pp. 1–6.

[13] W. Zhang, L. Ma, and X. Jiang, "Disaster-aware dynamic routing for sdn-based active-active data center networks," in *2019 International Conference on Networking and Network Applications (NaNA)*. IEEE, 2019, pp. 160–165.

[14] M. M. Tajiki, B. Akbari, and N. Mokari, “Optimal qos-aware network reconfiguration in software defined cloud data centers,” *Computer Networks*, vol. 120, pp. 71–86, 2017.

[15] H. A. Khosravi and M. R. Khayyambashi, “A system for providing load-aware virtual network service in a software-defined data center network,” *International Journal of Network Management*, vol. 27, no. 5, p. e1989, 2017.

[16] F. Rhamdani, N. A. Suwastika, and M. A. Nugroho, “Equal-cost multipath routing in data center network based on software defined network,” in *2018 6th International Conference on Information and Communication Technology (ICoICT)*. IEEE, 2018, pp. 222–226.

[17] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, “A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 393–430, 2018.

[18] J. Xu and K. Wu, “Living with artificial intelligence: A paradigm shift toward future network traffic control,” *Ieee Network*, vol. 32, no. 6, pp. 92–99, 2018.

[19] Q. Fu, E. Sun, K. Meng, M. Li, and Y. Zhang, “Deep q-learning for routing schemes in sdn-based data center networks,” *IEEE Access*, vol. 8, pp. 103 491–103 499, 2020.

[20] Y.-R. Chen, A. Rezapour, W.-G. Tzeng, and S.-C. Tsai, “Rl-routing: An sdn routing algorithm based on deep reinforcement learning,” *IEEE Transactions on Network Science and Engineering*, 2020.

[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] C. Xie, “Datacenter optical interconnects: Requirements and challenges,” in *2017 IEEE Optical Interconnects Conference (OI)*. IEEE, 2017, pp. 37–38.

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

[24] 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*, 2013, pp. 1–6.

[25] 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.

[26] 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.

[27] A. Botta, A. Dainotti, and A. Pescapè, “A tool for the generation of realistic network workload for emerging networking scenarios,” *Computer Networks*, vol. 56, no. 15, pp. 3531–3547, 2012.

[28] A. Botta, A. Dainotti, and A. Pescapé, “Do you trust your software-based traffic generator?” *IEEE Communications Magazine*, vol. 48, no. 9, pp. 158–165, 2010.

[29] A. Nucci, A. Sridharan, and N. Taft, “The problem of synthetically generating ip traffic matrices: Initial recommendations,” *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 3, pp. 19–32, 2005.

[30] L. Saino, C. Cocora, and G. Pavlou, “A toolchain for simplifying network simulation setup.” *SimuTools*, vol. 13, pp. 82–91, 2013.

[31] 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.

[32] 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.