This paper was appeared in IEEE JSAC Special Issue on Spectrum Sharing and Aggregation for Future Wireless Networks, 2016. Compared with the camera-ready, this is the latest author-reserved version.

Near-Optimal Routing Protection for In-Band Software-Defined Heterogeneous Networks

Huawei Huang *Member, IEEE,* Song Guo, *Senior Member, IEEE,* Weifa Liang, *Senior Member, IEEE,* Keqiu Li, *Senior Member, IEEE,* Baoliu Ye, *Member, IEEE,* and Weihua Zhuang, *Fellow, IEEE*

***Abstract*—Facing the spectrum supply-demand gap, heterogeneous network (HetNet) is a promising approach to achieve drastic gains in network coverage and capacity compared with macro-only networks, thus making it especially attractive to network operators. On the other hand, Software-Defined Networking brings a number of advantages along with many challenges. One particular concern is on the resilience for in-band fashioned control-plane. Existing approaches mainly rely on a *local rerouting* policy when performing the routing protection for the target sessions in Software-Defined Networks (SDNs). However, such a policy would potentially bring congestions in the neighbouring links of the failed one. To this end, we study a weighted cost-minimization problem, where the traffic load balancing and control-channel setup cost are jointly considered. Because this problem is NP-hard, we first propose a near-optimal *Markov approximation* based approach for in-band fashioned software-defined HetNets. We then extend our solution to an online case that handles a single-link failure. We also conduct theoretical analysis on the performance fluctuation due to the single-link failure. We finally carry out experiments by experimental simulation. Extensive numerical results show that the proposed algorithm has fast convergence and high efficiency in resource utilization.**

***Index Terms*—Routing Protection, In-Band, Software-Defined Heterogeneous Networks, Markov approximation.**

I. Introduction

Driven by the drastic growth in wireless data traffic, more denser, heterogeneous network deployment in the context of limited spectrum supply are becoming a reality for operators of carrier networks and Internet of Things. In order to improve the spectrum usage aiming at eliminating the spectrum supply-demand gap, the heterogeneous network (HetNet) is a promising method for achieving substantial gains in coverage and capacity, compared with macro-only networks, and especially attractive to network operators. For the example demonstrated in Fig. 1, in a heterogeneous Long Term Evolution-Advanced (LTE-A) wireless cellular network, a large number of small-cells (SCs) such as micro, pico and relay base stations (BSs), are deployed with macrocells to improve spatial reuse [1]–[3] via cell splitting, because these small-cell BSs can operate on the same wireless channel (a certain spectrum band) as the macro-cellular network.

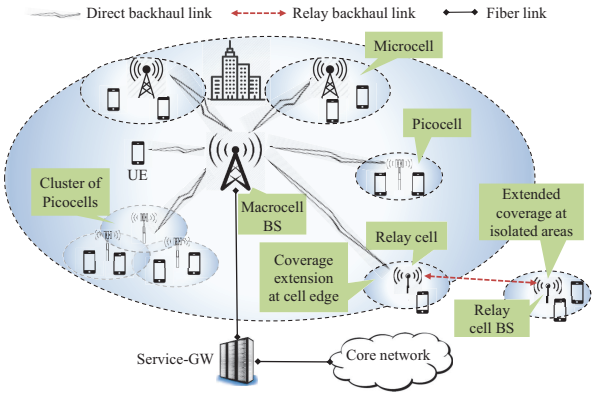


Fig. 1. The heterogeneous network uses a combination of macrocell and small-cell base stations to extend network coverage and capacity.

A. Relaying in HetNets

Relaying is considered as one of the key components for 3GPP releases 10 and 11 of LTE-A in order to improve the cell-edge user throughput, and to extend coverage to new areas by flexible and easy deployments [4]. A relay in an LTE-A access network is a base station that uses collaborative multi-hop communications at the cell edges. It can receive a weak signal and retransmit it with an enhanced quality. With relaying, User Equipments (UEs) can communicate with the network via a relay BS that is connected to a macrocell, using the LTE radio interface technology [4]. The macrocell BS may serve one or multiple relays in addition to directly serving the macro UEs [3]. Particularly, the multi-hop relaying has been identified as a valuable wireless paradigm in future HetNets, particularly for urban sparse area deployments. The use of multi-hop relaying can significantly improve the network coverage and capacity, due to the reduction of path loss by replacing the direct low quality link between the macrocell BS and UEs with multiple high-quality links through one or multiple relay BSs.

It has shown that better performance in terms of high-quality coverage, network capacity and lower expenditures (CAPEX and OPEX) can be achieved by supporting multi-hop

H. Huang is with the University of Aizu, Aizu-Wakamastu, 965-8580, Japan. (e-mail: davyhwang.cug@gmail.com).

S. Guo is with the Department of Computing, The Hong Kong Polytechnic University (e-mail: song.guo@polyu.edu.hk).

W. Liang is with the Research School of Computer Science, the Australian National University, Canberra, ACT 0200, Australia (e-mail: wliang@cs.anu.edu.au).

K. Li is with the School of Computer Science and Technology, Dalian University of Technology, Dalian 116024, China, and also with the School of Computer Science and Technology, Tianjin University, Tianjin 300072, China (e-mail: keqiu@dlut.edu.cn).

B. Ye is with the National Key Laboratory for Novel Software Technology, Nanjing University, Xianlin Campus, 163 Xianlin Avenue, Qixia District, Nanjing 210046, China (e-mail: yebl@nju.edu.cn).

W. Zhuang is with the Department of Electrical and Computer Engineering, University of Waterloo, ON, Canada, N2L 3G1 (e-mail: wzhuang@bbcr.uwaterloo.ca).

relay architecture [5]–[7]. However, using a multi-hop relay functionality requires more resources to transmit data through different hops, which may degrade the quality of service (QoS). Therefore, efficient spectrum management schemes is crucial to achieving high QoS while improving the network capacity.

B. Software-Defined Networking

Software-Defined Networking offers programmable features and functionalities (e.g., OpenFlow [8]) to dynamically manage network by shifting the control plane to centralized controllers. Thus, network operators can perform flexible management easily and fast realize a number of optimal network policies, e.g., traffic engineering [9], data security [10], fault diagnosis [11] and failure recovery (failover) [12].

In Software-Defined Networks (SDNs), the connection between a forwarding device (such as a switch/router which connects to a BS in wireless heterogenous networks) and a controller is used to exchange control-plane traffic, e.g., OpenFlow messages and the collected global network statistics [8]. The global network information is critical for control policies to make centralized decisions. Network status shall be collected as much as possible, such as the traffic rate in each link, the available flow table size in each forwarding switch, and reported to the controller via secure channels. The controller may respond with new instructions to each forwarding device. Such bidirectional communications contribute to the control traffic by a non-negligible fraction [13].

A controller usually interacts with forwarding devices via out-of-band control [14] connections in a dedicated network [15], [16]. The advantages of such an out-of-band network are in two-fold: (i) High security is provided for control signals because a dedicated network is used for communication; (ii) The control-to-device connection is still available through the dedicated network even if failures occur in the data plane. However, building such an out-of-band dedicated network could be very expensive in terms of resource-consumptions under some scenarios such as the widely distributed wireless access networks. Therefore, in a large-scale network with hundreds even thousands of forwarding devices, an *alternative* economic way is to use the so called *in-band* connection [17] for constructing the control plane. In this fashion, a controller communicates with a target forwarding device through a multi-hop routing path consisting of multiple intermediate relays. Using the Transport Layer Security (TLS) or Transmission Control Protocol (TCP) connections, the control-plane traffic can be relayed over the in-band controller-to-device channels. The usage of in-band connection can be found in either wired networks [18]–[20], or wireless networks [21]–[24].

C. Motivations

In the following, we address the importance of the routing protection for the in-band fashioned control plane of software-defined HetNets from three perspectives.

Firstly, although in-band connection is a practical approach, it poses many challenges. One is how to provide resilient communications between forwarding devices and the controller in case of link failures. In recent studies, Google has reported high delays and failures in configuring switches with a failure rate between 0.1% and 1% [16]. In a large network, the failures of the data plane occur more frequently. Each disconnected link can last 30 minutes on average [25]. In an in-band fashioned SDN, where control-plane traffic shares medium with the data plane traffic, even a single link failure may disconnect a large number of devices from their controllers, resulting in much worse damages than those of the out-of-band fashion [12]. Consequently, packets may not be forwarded correctly in the control-lost switches, thus leading to performance degradation, such as packet loss, loop routing, suboptimal or infeasible routing actions [8], [11]. The damage is even worse under wireless networks. Therefore, to deal with routing protection at the control plane for in-band HetNets is a fundamental issue.

Secondly, we notice that the emphasized control-plane oriented routing protection problem looks very similar to the data plane routing protection, which has received much attention in literature primarily utilizing the *policy* of *local rerouting* [15], [18], [20], [26]–[29], such as *detour*, *forward local rerouting* and *backward local rerouting*. However, this policy potentially brings congestions to the links near to the failed one, resulting in a higher control latency to the affected controller-to-device channels. In contrast, we strive to find a robust end-to-end *global rerouting* solution, corresponding to the *flow swapping* scenario [30], [31], especially in a dynamic environment where flows are frequently added or removed in certain groups of links simultaneously.

Finally, with respect to *methodologies* to address the routing recovery problem, there are two major categories: *restoration* [19], [32] and *protection* [20], [33]. In the former scheme, the recovery paths can be either preplanned or calculated on-demand, but network resources (such as forwarding rules and link bandwidth) will not be allocated until a failure is detected. We can see that such an approach inherently results in a long recovery time and high packet losses. In contrast, in the *protection* scheme, backup resources are always pre-planned and reserved such that once a failure is detected, recovery can be made immediately. As a result, when fast recovery is a major concern, the *protection* is preferable. In addition, the experimental studies in literature [19], [32] revealed that path *protection* is more qualified than *restoration* with respect to the sub-50 ms fast failure recovery requirement [34] of carrier-grade networks. Therefore, in this paper we adopt the *protection* scheme.

D. Goal

In software-defined HetNets [35], [36], when realizing the efficient spectrum utilization for the wireless backhaul link, there are two crucial conflicting issues to be addressed when deploying routing protection to the in-band fashioned control plane: (i) maximizing the spectrum-usage efficiency for data-plane traffic, and (ii) improving the resilience on control-plane traffic.

Since control-plane traffic shares the bandwidth resource with data plane traffic, network operators should carefully

utilize the spectrum bandwidth towards control traffic, such that the network performance (e.g., throughput) can be optimized. One to achieve this goal is to balance the control traffic over all backhaul links. Another is the setup cost of the control channels when providing the resilience for the control-plane traffic. Because the establishment and maintenance of TCP/TLS connections require a number of routing table entries [37] and message exchanges. Generally, the setup cost is positively proportional to the length of the control-channel path. Consequently, the length of a selected path should also be taken into consideration when launching the control channels.

To this end, in the software-defined heterogeneous networks, we study a weighted cost minimization problem, in which the control-plane traffic load balancing and control-channel setup cost are jointly considered when selecting the protection paths for control channels. Since the multiple resource constrained routing is NP-complete [38], [39], we propose a near-optimal algorithm, using the Markov approximation technique [40]. Particularly, we extend our solution to an online case that can handle the dynamic single-link failure one a time. The incurred performance fluctuation is also theoretically analyzed. Finally, extensive simulations are conducted to show that the proposed algorithm converges quickly and is efficient on resource utilization than the existing benchmarks.

Our study leads to the following major contributions:

- We strive to provide an optimal routing protection for control-plane traffic in in-band fashioned software-defined HetNets. Our approach can be extended to the routing protection in data plane.
- To solve the weighted cost-minimization problem, a near-optimal algorithm has been proposed, using the Markov approximation techniques. In particular, we design a Markov chain with a state space of all feasible protection solutions and a well devised transition rate matrix such that the analytical performance of the proposed algorithm can be guaranteed.
- Compared with existing benchmarks, the proposed algorithm can provide more robustness and efficiency by handling a single-link failure with fast convergence. The theoretical performance fluctuation of our algorithm due to a single-link failure is also thoroughly studied with closed-form expression.

The rest of the paper are organized as follows. Section II reviews related work. Section III introduces preliminaries and the system model. Then, the optimization problem is formulated using integer programming. Our proposed Markov approximation based algorithm is presented in Section IV. Further, Section V extends the proposed algorithm to dynamically handle the single-link failure. Extensive evaluation results are shown in Section VI. Finally, Section VII concludes this work and identifies the future work.

II. Related Work

A. Multi-hop Relay System in HetNets

As a hot research topic with significant application potential, relay technologies have been actively studied in literature. For example, in a multi-user, multi-relay Orthogonal Frequency-Division Multiple Access (OFDMA) based cellular network, an energy-efficient subcarrier allocation problem is studied in [41]. BenMimoune et al. [42] demonstrate that multi-hop relay is one of the most promising technologies to overcome the coverage and capacity problem in LTE-A networks. To reduce the inter-cell handoff frequency and handoff failure ratio, they propose a relay selection strategy that takes handoff into consideration in LTE-A multi-hop relay networks. The same authors [43] later propose a new resource allocation framework for 5G multi-hop relay systems, aiming to overcome the additional challenges introduced by multi-hop relay stations. Recently, Sapountzis et al. [44] claim that, the backhaul constraints are emerging as a key performance bottleneck in future HetNets, because of the continuous improvement of the radio interface and the desire for inexpensive multi-hop backhaul links that can reduce CAPEX/OPEX. They propose an analytical framework for user association that jointly considers radio access and backhaul performance in backhaul-limited HetNets. Particularly, an SDN-based implementation of the proposed approach is also presented. The centralized programmable control framework is described with the consideration of the operating mechanisms in the application, controller, network and user planes.

Although the current LTE-A relay standard is restricted to a single-hop relaying to reduce the system complexity, it should be noticed that this principle is specifically for the data-plane, where the traffic data volume is significantly large. In contrast to the existing studies based on the multi-hop relaying architecture of a HetNet, we find out that the steering towards the in-band fashioned control-plane traffic in software-defined HetNets has not been well studied.

B. Taxonomy of Failure-Recovery Strategies

With respect to the link failure recovery for SDNs, the proposed strategies from literature can be generally classified into three categories: restoration [19], [32], *cold-backup* protection [19], [32], [45], and *hot-backup* protection [46]. We first review the existing studies and then compare them with our approach.

Restoration. Sharma et al. present restoration mechanisms in [19], [32] for OpenFlow networks. In case of a link failure, the controller reacts to the link failure according to the following steps: (a) remove the affected forwarding rules; (b) compute backup paths; and (3) install the new required rules.

Cold-backup protection. In this protection, only the forwarding rules are allocated in the beginning, but traffic is not redirected to the backup paths until the failure occurs. For example, in the same work [19], [32], Sharma et al. also implement a *group table* based *fast-failover* mechanism [8] for OpenFlow networks. Backup paths are pre-computed and installed to the group table. Their experimental studies show that path protection is more qualified than restoration with respect to the sub-50 ms fast failure recovery requirement [34] of carrier-grade networks. Moreover, Borokhovich et al. [45] introduced classic graph search algorithms (depth-first search

and breadth-first search) to OpenFlow networks. The controller invokes one of these algorithms to compute backup paths, along which routing rules are pre-installed into the group-table of switches.

Hot-backup protection. In this scheme, the bandwidth resource of backup paths is fully allocated in the initial setting, such that the backup paths carry the same traffic as the primary working path to avoid disruption of connection. For example, authors in [46] apply a '1+1' protection to the data plane of an OpenFlow network, where backup paths are pre-configured and carry the duplicated traffic. Thus, the destination-switch can still receive packets when a link failure occurs. Similarly, as the most closely related work regarding the protection of control traffic, to achieve resilient control traffic forwarding, Hu et al. [20] investigate the protection of control traffic in SDNs. Particularly, they utilize the local rerouting and constrained reverse forwarding protection to combine each device with multiple controllers. This scheme enables a device to locally react to link failures and redirect the control traffic to a controller through one of the backup forwarding options.

C. Comparison with Our Work

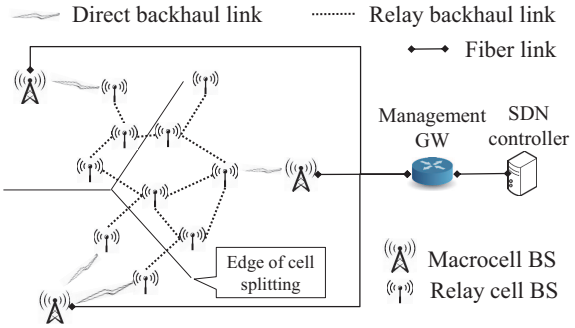
Compared with the conventional routing-resilience approaches, we have the following observations: 1) Existing approaches rarely specifically address the control-plane routing protection; 2) None of the existing approaches can provide an optimal fast recovery solution with efficient resource utilization, when a single-link failure occurs in large-scale networks. To fill this gap, we emphasize on the routing protection mechanism for the control-plane traffic in a software-defined HetNet. In particular, a Markov approximation based routing protection scheme for the in-band fashioned control-plane of SDNs is proposed. Our approach can yield a near-optimal *global rerouting* solution, which is suitable for *flow swapping* scenarios [30], [31] in a dynamic environment, where traffic flows are frequently refreshed simultaneously in certain groups of links.

III. System Model and Problem Formulation

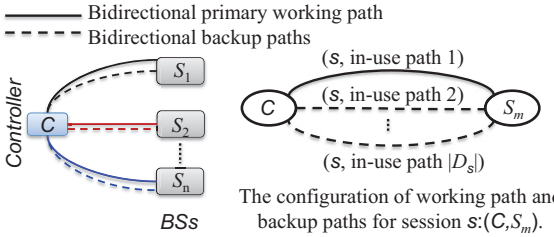
A. Preliminary and System Model

1) Relay and Backhaul Link Model of HetNets: We consider a simplified software-defined heterogeneous network where the sets of deployed BSs are stationary. Without loss of generality, we only consider macrocell BSs and small-cell BSs as the target BSs that need the routing protection for the in-band fashioned control plane. The model can be applied to scenarios with multiple types of small cells.

A challenge for the deployment of HetNets is to cope with inter-layer and intra-layer interferences when multi-hop sharing links are deployed within a specified spectrum band. By adopting the "frequency division" approach, i.e., allocating the orthogonal spectrum to the backhaul links in different layers based on the OFDMA scheme, we assume that there is no interference between neighbouring cells. OFDMA is a spectrum-efficient scheme that enables high data rates and permits multiple users to share a common channel.



(a) An illustration of software-defined HetNet with multi-hop relaying backhaul links based on the future HetNet topology used in [44].



(b) The protection model for the control-plane traffic from controller-to-device sessions.

Fig. 2. The system model of this paper. Note that, there can be multiple controllers over network. Here we only illustrate an example with only one controller.

2) Establishment and Maintenance of Control Channel: In the perspective of SDN associated system model, one controller or multiple controllers coordinately manage an SDN forwarding device remotely over a multi-hop relaying network. The connecting fashion of this network can be constructed by intermediate BSs according to the in-band controller connection [8]. The only requirement is that the intermediate BSs should support TCP/IP connectivity. Typically, every controller-to-device channel (also called a controller-to-device *session*) can be established as a single network connection between the target BS and the controller, using TLS or plain TCP protocols [8]. Once a control connection is built, it must be maintained by the underlying TLS or TCP connection mechanisms, until the connection is terminated by TCP timeouts or TLS session timeouts [8].

3) Protection for Control Connection: We consider the *Dedicated Backup Path Protection (DBPP)* scheme [47], [48], which belongs to the *hot-backup* protection category. For example, one popular DBPP is the '1+1' protection [19], [33], [49], where a primary path is protected by one dedicated backup path and traffic is duplicated on both the primary path and the only backup path. Note that, the proposed approach is a general framework that can be applied to other recovery mechanisms as well.

4) System Model: We consider a software-defined HetNet $G = (\mathbb{V}, \mathbb{E})$ with BS set $\mathbb{V}$ and link set $\mathbb{E}$. As shown in Fig. 2(a), $\mathbb{V}$ includes the management gateway (GW) node, all macrocell and small-cell BS nodes, especially the edge relay BSs. The link set $\mathbb{E}$ contains *fiber links* between the management GW and the macrocell BSs, the *direct backhaul links* and *relay backhaul links*. All backhaul links are bidirectional

with the fixed given spectrum bandwidth. Let c_l and $d_l (l \in \mathbb{E})$ denote the total capacity of link l and the portion allocated to all data-plane traffic over the link, respectively. The currently available link capacity for control-plane traffic is presented by $c_l - d_l$, ($l \in \mathbb{E}$). The currently available session-serving capacity on node $v \in \mathbb{V}$ is denoted by ϕ_v, because each node v (e.g., a cell BS) can only serve a finite number of sessions due to limited space of forwarding table and number of TCP connection ports.

Given a set of controller-to-device sessions $\mathbb{S}$, each controller-to-device session $s \in \mathbb{S}$ has to be equipped with a set of required in-use paths, denoted as D_s, which includes one primary path and $|D_s|-1$ *hot-backup* paths. For example, in Fig. 2(b), all in-use paths for session s between controller C and BS S_m at a time are illustrated as (s,1), (s,2), $\cdots$, (s,$|D_s|$). The adopted DBPP scheme belongs to *link-sharing protection* [47], and therefore the provided candidate paths in J_s for each session s should not have a high *link-shared* degree. This is because *both the primary path and backup paths are not likely to fail at the same time by a single link-failure* when the candidate paths are highly *link-shared*. Note that, how to find *non link-shared* candidate paths for each session is not a focus of this paper. The major notations used in this paper are summarized in Table I.

TABLE I
SYMBOLS AND VARIABLES

Notations	Description
($\mathbb{V}$,$\mathbb{E}$)	software-defined HetNet with node set $\mathbb{V}$ and link set $\mathbb{E}$
$\mathbb{S}$	a set of controller-to-device sessions
J_s	a set of candidate paths for session $s \in \mathbb{S}$
D_s	a set of required in-use paths for session $s \in \mathbb{S}$, including one working path and $\|D_s\|-1$ ($\|D_s\| \geq 2$) backup paths
R_s	demanding traffic rate of session $s \in \mathbb{S}$
r_l	aggregated control-plane traffic load on link $l \in \mathbb{E}$
d_l	aggregated data plane traffic load on link $l \in \mathbb{E}$
c_l	channel capability on link $l \in \mathbb{E}$
ϕ_v	currently available session-serving capacity on node $v \in \mathbb{V}$
$\|.\|$	size of a set or length of a path
z_s^p	a binary variable indicating whether session $s \in \mathbb{S}$ selects path $p \in J_s$ as one of its required in-use paths
f	a feasible routing-protection configuration for all sessions
$\mathcal{F}$	the set of all feasible configurations for the whole system
$u_s(f)$	system cost of a session $s \in \mathbb{S}$ under configuration $f \in \mathcal{F}$
u_f	overall system cost under a given configuration $f \in \mathcal{F}$, i.e., $u_f = \sum_{s \in \mathbb{S}} u_s(f)$

B. Problem Formulation

The Control-Plane Routing Protection (shorten as **CPRP**) problem is stated and formulated as follows.

1) Path Selection: The target of **CPRP** problem is to find an optimal path set for all controller-to-device sessions. To denote whether a candidate path $p \in J_s$ is selected by session $s \in \mathbb{S}$ as one of its required in-use paths, we define a binary variable z_s^p as:

$$z_s^p = \begin{cases} 1, & \text{if candidate path } p \in J_s \text{ is selected by} \\ & s \text{ as one of its required in-use paths;} \\ 0, & \text{otherwise.} \end{cases}$$

With this definition, the entire decision space of all possible path-selections can be expressed as:

$$\mathcal{F} = [\{z_s^p\} | z_s^p \in \{0,1\}, \sum_{p \in J_s} z_s^p = |D_s|, s \in \mathbb{S}].$$

2) Minimization of Joint Weighted System Cost: SDN network operators perform traffic engineering to improve resource utilization, which is normally measured in terms of two objectives:

- To ensure *load balance* by decreasing the aggregated traffic load on the most severely congested link;
- To reduce the *configuration cost* in each node.

Consequently, the overall weighted system cost described in our objective includes two terms: 1) the *largest control flow rate over all links*, and 2) the *average connection-setup cost on each device node*.

As a result, the **CPRP** problem can be formulated by the following integer programming:

$$\textbf{CPRP}: \textbf{Minimize}(\max_{l \in \mathbb{E}}(r_l) + \frac{\omega}{|\mathbb{V}|} \sum_{s \in \mathbb{S}} \sum_{p \in J_s} z_s^p \cdot |p|) \quad (1a)$$

$$\text{s.t.:} \sum_{p \in J_s} z_s^p = |D_s|, \forall s \in \mathbb{S} \quad (1b)$$

$$r_l = \sum_{s \in \mathbb{S}} \sum_{p \in J_s, l \in p} z_s^p \cdot R_s, \forall l \in \mathbb{E} \quad (1c)$$

$$r_l \leq c_l - d_l, \forall l \in \mathbb{E} \quad (1d)$$

$$\sum_{s \in \mathbb{S}} \sum_{p \in J_s, v \in p} z_s^p \leq \phi_v, \forall v \in \mathbb{V} \quad (1e)$$

$$Variables: z_s^p \in \{0,1\}, \forall p \in J_s, \forall s \in \mathbb{S}$$

Objective function (1a) is the proposed overall weighted system cost that captures both mentioned objectives. Particularly, the first term $\max_{l \in \mathbb{E}}(r_l)$ denotes the *largest control traffic rate over all backhaul links* while the second term $\frac{1}{|\mathbb{V}|} \sum_{s \in \mathbb{S}} \sum_{p \in J_s} z_s^p \cdot |p|$ calculates the average connection-setup cost in each node. Note that, the tradeoff between these two cost terms is allowed to be freely tuned by introducing a weight factor ω. Furthermore, constraint (1b) claims that for each session s, exact $|D_s|$ numbers of in-use paths must be selected from its candidate path set J_s. Then, (1c) calculates the aggregated traffic rate on each link, as the sum of traffic demand from all passing-through sessions. The capacity constraints on links and nodes are specified by constraints (1d) and (1e), respectively.

IV. DISTRIBUTED NEAR-OPTIMAL PATH SELECTION ALGORITHM

Path selection under constrained resources is NP-complete [38], [39]. The **CPRP** problem is a combinatorial optimization, in which the global optimal solution consists of partial path-selection decision for each session. Since there is no computationally efficient solution in a centralized manner, we strive for designing a distributed algorithm by adopting a framework of Markov approximation technique [40]. In the following, we detail the two stages in the design of the

algorithm under the Markov approximation framework: *log-sum-exp approximation* and *implementation of Markov chains*.

A. Log-Sum-Exp Approximation Approach

Let $f = \{z_s^p, \forall p \in J_s, \forall s \in \mathbb{S}\}$ denote a configuration for the **CPRP** problem, and $\mathcal{F}$ the set of all feasible configurations that are already known. For the convenience of presentation, we denote by u_f the system objective function (1a) corresponding to a given configuration f. To better understand the log-sum-exp approximation, let each configuration $f \in \mathcal{F}$ associate with a probability p_f, indicating the percentage of time that configuration f is in use. Then, **CPRP** can be approximated as follows via applying the approximation technique in [40]:

$$\begin{aligned} \textbf{CPRP}(\beta): \min & \sum_{f\in\mathcal{F}} p_f u_f + \frac{1}{\beta}\sum_{f\in\mathcal{F}} p_f \log p_f \\ \text{s.t.:} & \sum_{f\in\mathcal{F}} p_f = 1 \end{aligned} \tag{2}$$

where β is a large positive constant and related to the performance of this approximation approach. The motivation behind is that it potentially leads to distributed solutions. Let $p^*_{f\in\mathcal{F}}$ be an optimal solution of the **CPRP**(β), λ denotes the Lagrangian multiplier associated with the equality constraint in (2) under p^*_f. Then, by solving the following Karush-Kuhn-Tucker (KKT) conditions [50] of the problem in (2):

$$\begin{cases} u_f + \frac{1}{\beta}\log p^*_f + \frac{1}{\beta} + \lambda = 0, \forall f \in \mathcal{F} \\ \sum_{f\in\mathcal{F}} p^*_f - 1 = 0 \\ \lambda \geq 0 \end{cases} \tag{3}$$

An optimal solution can be derived which is:

$$p^*_f = \frac{\exp(-\beta u_f)}{\sum_{f'\in\mathcal{F}}\exp(-\beta u_{f'})}, \forall f \in \mathcal{F}. \tag{4}$$

Remark 1: With the log-sum-exp approximation approach described above, we obtain an approximate version of the **CPRP** problem with the assistance of an *entropy* term $\frac{1}{\beta}\sum_{f\in\mathcal{F}} p_f \log p_f$. If we can time-share among different configurations according to the optimal solution p^*_f in (4), then **CPRP** can be solved approximately within a bound $\frac{1}{\beta}\log|\mathcal{F}|$, which can be small by a proper choice of a large β.

B. Markov Chain Design

Here we design a Markov Chain (MC) with a state space being the set of all feasible configurations $\mathcal{F}$ and a stationary distribution denoted as p^*_f in (4). Since the system operates under different configurations, the transition between two states in the designed MC indicates replacing an in-use path for any session. Therefore, in the implemented MC, if the transitions among states can be trained to converge to the desired stationary distribution p^*_f, the system can achieve near-optimal performance.

To construct a time-reversible MC [40] with stationary distribution p^*_f, let $f, f' \in \mathcal{F}$ denote two states of MC, and use $q_{f,f'}$ as the nonnegative transition rate from state f to f'. Furthermore, we have to ensure that: (a) in the resulting MC, any two states are reachable from each other, and (b) the *detailed balance equation* $p^*_f q_{f,f'} = p^*_{f'} q_{f',f}$ should be satisfied, $\forall f, f' \in \mathcal{F}$. Our design is as follows.

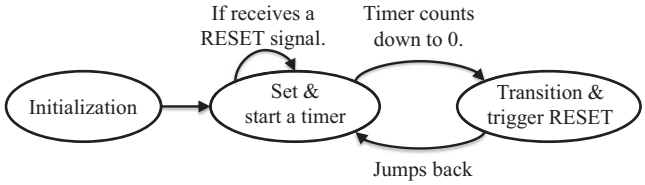


Fig. 3. State machine for each session in the proposed algorithm.

1) State-Space-Structure: Recall that a configuration $f \in \mathcal{F}$ represents a set of in-use paths for all sessions. Initially, we set the transition rate between two configurations f and f' to be 0, unless they satisfy the following two conditions:

- **C1:** $|f \cup f'| - |f \cap f'| = 2$
- **C2:** $f \cup f' - f \cap f' \in J_{\bar{s}}$

where $\bar{s}$ is the session which causes the state transition $f \to f'$. That is, if session $\bar{s}$ makes a single path swapping, the state f transits to f'.

2) Transition Rate Matrix Training: Both traffic statistics on links and the number of forwarding rules installed in devices can be easily pulled by controllers. For example, in OpenFlow networks [8], the traffic rate in each path can be measured via `meter table` entries and inquired through controller-to-device `Echo` messages. On the other hand, the consumption of the flow table size in a device can be obtained by the controller via sending `Read-State` messages. Therefore, the cost of the current network configuration can be acquired by the controller at any time. Particularly, we let the transition rate $q_{f,f'}$ positively correlated to the difference of system performance under two adjacent configurations f and f' in the state matrix. In detail, the transition rate $q_{f,f'}$ is designed as:

$$\begin{cases} q_{f,f'} = \frac{1}{\exp(\tau)}\exp(\frac{1}{2}\beta\sum_{s\in\mathbb{S}}(u_s(f) - u_s(f'))) \\ q_{f',f} = \frac{1}{\exp(\tau)}\exp(\frac{1}{2}\beta\sum_{s\in\mathbb{S}}(u_s(f') - u_s(f))) \end{cases} \tag{5}$$

where τ is a conditional positive constant that avoids overflow computing of exp(.). The design of $q_{f,f'}$ in (5), in practice, is likely to make the system jump to a configuration with better performance. This is because when $\sum_{s\in\mathbb{S}}(u_s(f) - u_s(f')) > 0$ and the performance gap between f and f' is greater, the transition rate $q_{f,f'}$ will be bigger, and vice versa.

C. Implementation of MC based Algorithm

The implementation based on our designed Markov chain is shown in Algorithm 1, in which the controller creates a dedicated processing-thread for each of its holding sessions. Therefore, this algorithm can execute on a single controller or

Algorithm 1 *Sojourn-and-Transit* Algorithm to Solve **CPRP**

for each $s \in \mathbb{S}$ **do**
  execute **Procedure Initialization**
  execute **Procedure Set-timer** for s
end for
while system is still running **do**
  /$\star$**Procedure Transit**$\star$/
  if T_s expires **then**
    $z_s^{p_{old}} \leftarrow 0$
    $z_s^{p_{new}} \leftarrow 1$
    execute **Procedure Set-timer** for s
    broadcast RESET($u_{f'}, \{s\}$) signal for other sessions
  end if
  /$\star$**Procedure RESET**$\star$/
  if session s receives RESET($u_{f'}, \bar{\mathbb{S}}$) message **then**
    $u_f \leftarrow u_{f'}$
    refresh and start timer T_s ($s \in \mathbb{S} - \bar{\mathbb{S}}$) invoking (6)
  end if
end while

Algorithm 2 Startup for a session (**Procedure Initialization**)

Input: a session $s \in \mathbb{S}, J_s$
Output: D_s

initializes a dedicated processing-thread for s
$D_s \leftarrow$ randomly select $|D_s|$ feasible paths from J_s

multiple controllers, which can apportion all the processing-threads. We consider how to assign threads to multiple controllers in our future work. Typically, each dedicated thread follows a general state machine shown in Fig. 3, using which we detail the algorithm for the single-controller case as follows.

- **Procedure Initialization:** For each session $s \in \mathbb{S}$, the controller creates an associated thread, then randomly selects $|D_s|$ feasible not-in-use paths that satisfy the resource requirement from candidate path set J_s.
- **Procedure Set-timer:** Let f and $f^{'}$ denote the current and next targeting configurations, respectively. For each session $s \in \mathbb{S}$, the controller first randomly selects one feasible path from the not-in-use path set (i.e., $J_s \backslash D_s$), and one in-use path from D_s. The system cost of the current configuration u_f can be measured by the controller, which then estimates the performance of the target configuration, i.e., $u_{f'}$, if swapping these two paths p_{old} and p_{new}. Meanwhile the controller triggers an exponentially distributed timer T_s for session s with a mean value of $\exp(\tau - \frac{1}{2}\beta(u_f - u_{f'})) \cdot (|D_s| \cdot (|J_s| - |D_s|))^{-1}$. The controller then broadcasts a RESET($u_{f'}, s$) message for other sessions to notify them the updated system cost $u_{f'}$.
- **Procedure Transit:** When a timer $T_{s \in \mathbb{S}}$ expires, the controller swaps the chosen pair of paths p_{old} and p_{new} for s, then the execution thread repeats **Procedure Set-timer** for session s.
- **Procedure RESET:** When a session $s \in \mathbb{S}$ receives a RESET($u_{f'}, \bar{\mathbb{S}}$) message, the controller refreshes all the other timers T_s ($\forall s \in \mathbb{S} - \bar{\mathbb{S}}$), according to (6) with the updated system cost $u_{f'}$.

Algorithm 3 Set timer for a session (**Procedure Set-timer**)

Input: session $s \in \mathbb{S}$
Output: T_s, p_{old}, p_{new}

$p_{old} \leftarrow$ one in-use path randomly selected from D_s
$p_{new} \leftarrow$ one feasible not-in-use path randomly selected from $J_s \backslash D_s$
measures the current system cost u_f
estimates the system cost $u_{f'}$ of the target configuration if replace p_{old} with p_{new}
generates a random exponentially distributed timer T_s for thread s with mean equal to

$$\frac{\exp(\tau - \frac{1}{2}\beta \sum_{\bar{s} \in \mathbb{S}} (u_{\bar{s}}(f) - u_{\bar{s}}(f')))}{|D_s| \cdot (|J_s| - |D_s|)}, \quad (6)$$

and begins counting down

D. Algorithm Analysis

Now, we have the following theorem.

Theorem 1: Algorithm 1 realizes a time-reversible Markov chain with the stationary distribution given in (4)*.*

The proof is given in Appendix-A. Furthermore, we make some notable remarks:

Remark 2: The computation complexities of ***Algorithm 2****,* ***Algorithm 3*** *and* ***Algorithm 4*** *are* $\boldsymbol{O}(|J_s|)$*,* $\boldsymbol{O}(|J_s| + |D_s|)$ *and* $\boldsymbol{O}(\sum_{s \in \mathbb{S}} |J_s|)$*, respectively.*

Remark 3: The proposed algorithm can be extended to other traffic engineering problems in SDN systems, e.g., finding the resilient routing paths for *data plane traffic* between any pair of devices.

Remark 4: Because this algorithm executes in a distributed manner for each session, it can be applied to more practical scenarios where multiple controllers are deployed over large-scale networks. This requires the system to know the performance under a target configuration $f^{'}$ via a probing phase. It can be achieved by the minimum number of information exchanges among relevant controllers. For a transition from f to $f^{'}$, where only one session changes a single path, its holding controller only has to notify this event to the other "invariant" controllers. Then, the "next" target system performance $\sum_{s \in \mathbb{S}} u_s(f^{'})$ can be unveiled immediately in each controller. In order to embed the proposed algorithm to the SDN controller module, new OpenFlow messages should be included to denote the timer expiration and the RESET event. Furthermore, extra Eastern-Western interfaces [8] should be developed to enable information exchanges among multiple controllers.

We then study the convergence property of the proposed algorithm by estimating its *convergence time*. In general, the convergence time of a Markov chain can be examined by the *mixing time*. Let $\boldsymbol{H}_t(f)$ denote the probability distribution of all states in $\mathcal{F}$ at time t if the initial state is f. Recall that $\boldsymbol{p}^*$ is the stationary distribution of the designed Markov chain. We define the mixing time:

$$t_{mix}(\epsilon) := \inf\{t \geq 0 : \max_{f \in \mathcal{F}} \|\boldsymbol{H}_t(f) - \boldsymbol{p}^*\|_{TV} \leq \epsilon\}, \quad (7)$$

where $\epsilon>0$ describes the gap between the optimal performance and that of the converged solutions, and term $\|.\|_{TV}$ denotes the total variance distance between the probability distributions $\boldsymbol{H}_t(f)$ and $\boldsymbol{p}^*$. Then, letting $u_{\max} = \max_{\forall f\in\mathcal{F}} u_f$, $u_{\min} = \min_{\forall f\in\mathcal{F}} u_f$, $\varpi = \prod_{s\in\mathbb{S}} \binom{|J_s|}{|D_s|}$ and $\zeta = \sum_{s\in\mathbb{S}} |D_s|\cdot(|J_s|-|D_s|)$, we have the following results.

*Theorem 2: The mixing time $t_{mix}(\epsilon)$ for the constructed Markov chain in **Algorithm** 1 is bounded as follows:*
for general $\beta\in(0,\infty)$,

$$t_{mix}(\epsilon) \geq \frac{\exp[\tau - \frac{1}{2}\beta(u_{\max}-u_{\min})]}{2\zeta}\ln\frac{1}{2\epsilon}, \tag{8}$$

and

$$\begin{aligned} t_{mix}(\epsilon) \leq & 2\zeta\varpi^2\exp[\frac{3}{2}\beta(u_{\max}-u_{\min})+\tau]\cdot \\ & [\ln\frac{1}{2\epsilon}+\frac{1}{2}\ln\varpi+\frac{1}{2}\beta(u_{\max}-u_{\min})]. \end{aligned} \tag{9}$$

The proof is given in Appendix-B.

Remark 5: We here discuss the trade-off between the optimality gap ($\frac{1}{\beta}\log|\mathcal{F}|$) and the mixing time under different β. As $\beta\to\infty$, the optimality gap approaches zero, but the upper bound of mixing time scales with $\exp(\Omega(\zeta\varpi^2))$ and approaches infinity, resulting in slow convergence.

V. Online Handling and Theoretical Analysis under Single-Link Failure

In this section, we extend the proposed Algorithm 1 to the online case that handles the dynamic single-link failure [12].

A. Operations When A Link Fails

When a single link fails, any candidate paths and in-use paths which include the failed link become invalid and should be removed for each session. Since we assume that only a single link can fail at each time, there is at least one in-use path working for each session. Therefore, the connection between the controller and any device will not be disturbed. Then, we present the additional operations with respect to a single-link failure in Algorithm 4.

After removing all invalid paths in **Step 1**, **Step 2** fills up the vacancy of desired in-use paths and ensures control traffic protection. Finally, other sessions should be notified of the updated overall system cost via RESET messages to refresh their timers in **Step 3**. The controller then continues listening by invoking **Procedure Transit** and **Procedure RESET** of Algorithm 1. In addition, it is worth noting that the controller can deploy the required protection paths according to a sub-optimal solution after a link failure and before achieving the convergence.

B. Theoretical Performance Fluctuation of Single-Link Failure

When the invalid paths are removed, the configurations involving those paths should be deleted from the original Markov chain M. Let $\hat{M}$ denote the new Markov chain after removing all invalid configurations based on M, and $\mathcal{G}$ as the *survived* configuration space in $\hat{M}$. Accordingly,

Algorithm 4 Online Dynamic Handling of Single-Link Failure

1: **Step 1:** Remove all the candidate paths and in-use paths which involve in the failed link for each session.
2: **Step 2:**
3: $\bar{\mathbb{S}}\leftarrow\emptyset$
4: **for** $s\in\mathbb{S}$ **do**
5: **if** s lost any in-use path **then**
6: $D_s\leftarrow$ controller randomly picks up feasible not-in-use paths from the updated $J_s\backslash D_s$
7: $\bar{\mathbb{S}}\leftarrow s$
8: **end if**
9: **end for**
10: **Step 3:** Controller broadcasts RESET($u_f,\bar{\mathbb{S}}$) signals.

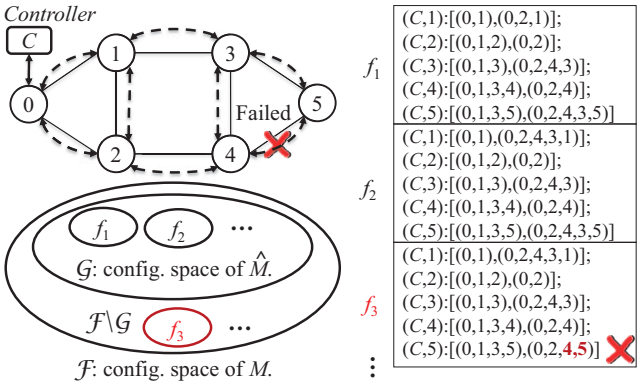


Fig. 4. An example of operations when a single-link failure occurs.

the *disappeared* configuration space is $\mathcal{F}\backslash\mathcal{G}$. For example, as shown in Fig. 4, when link (4,5) fails, any configuration including this link, e.g., f_3, will be moved to $\mathcal{F}\backslash\mathcal{G}$. The remaining configurations f_1, f_2, $\cdots$ are moved to $\mathcal{G}$. Note that, it can be proved that $\hat{M}$ is still irreducible.

We are interested in the robustness of the proposed Algorithm 1 in the online case. This motivates us to further study the performance fluctuation from the state when a link failure occurs to the converged performance in $\hat{M}$.

At first, the stationary distribution of the *configurations* in $\hat{M}$ is denoted by $\mathbf{q}^*:[q_g^*(\mathbf{u}),g\in\mathcal{G}]$. Furthermore, we define another vector $\hat{\mathbf{q}}:[\hat{q}_g(\mathbf{u}),g\in\mathcal{G}]$ to indicate the distribution of configurations $g\in\mathcal{G}$ in M when a link failure occurs and before **Step 3** of Algorithm 4. We use the *total variation distance* [51] $d_{TV}(\mathbf{q}^*,\hat{\mathbf{q}})$ to quantify the distribution difference of all configurations $g\in\mathcal{G}$ between $\hat{M}$ and M. We then have the following lemma.

Lemma 1: (a) The total variation distance between $\boldsymbol{q}^*$ and $\hat{\boldsymbol{q}}$ is bounded by

$$d_{TV}(\boldsymbol{q}^*,\hat{\boldsymbol{q}}) \triangleq \frac{1}{2}\sum_{g\in\mathcal{G}}|q_g^*-\hat{q}_g| \leq \frac{|\mathcal{F}\backslash\mathcal{G}|}{|\mathcal{F}|}. \tag{10}$$

(b) By denoting $S_1\subseteq\mathbb{S}$ as the set of sessions which lost a candidate path due to the link-failure, and $S_{im}\subseteq S_1$ as an imaginary *set of sessions which select the disappeared path if*

it still exists, we have

$$\frac{|\mathcal{F}\backslash\mathcal{G}|}{|\mathcal{F}|}=\frac{\sum_{\forall S_{im}\subseteq S_1}\{\prod_{s\in S_{im}}\binom{|J_s|-1}{|D_s|-1}\times\prod_{s\in S_\perp}\binom{|J_s|-1}{|D_s|}\}}{\prod_{s\in S_1}\binom{|J_s|}{|D_s|}} \quad (11)$$

where J_s, D_s are the path sets for session $s\in\mathbb{S}$ before link failure, and $S_\perp=\{S_1\backslash S_{im}\}$.

The proof is given in Appendix-C.

C. Case Study under '1+1' Protection Scheme

Based on Lemma 1, we now study a special case, which is named as **'1+1' protection with equal numbers of available candidate paths**. We adopt the '1+1' protection mechanism [19], [33] and provide each session with the same number of initial candidate paths, i.e., $|J_s|$ is same for $\forall s\in\mathbb{S}$ in the initial stage of controller connection setup. In addition, when a single-link failure occurs, to ensure that at least one candidate path can be chosen for each session, we assume that $(|J_s|-1)-|D_s|\geq 1$, $\forall s\in\mathbb{S}$. Consequently, Equation (11) can be rewritten as

$$\frac{|\mathcal{F}\backslash\mathcal{G}|}{|\mathcal{F}|}=\frac{\sum_{\forall S_{im}\subseteq S_1}\{\prod_{s\in S_{im}}\binom{|J_s|-1}{1}\times\prod_{s\in S_\perp}\binom{|J_s|-1}{2}\}}{\prod_{s\in S_1}\binom{|J_s|}{2}}. \quad (12)$$

Letting $u_{\max}=\max_{g\in\mathcal{G}}u_g$, we have the following theoretical bound on the performance perturbation under this special case when a single-link failure occurs. Note that, $u_{\max}$ is the performance under the worst configuration during $\mathcal{F}$.

*Theorem 3: The performance perturbation of a single-link failure under the special case **'1+1' protection with equal number of available candidate paths** is bounded by*

$$\|\boldsymbol{q}^*\boldsymbol{u}^T-\hat{\boldsymbol{q}}\boldsymbol{u}^T\|\leq\min(u_{\max},2u_{\max}[1-(1-\frac{2}{|J_s|})^{|\mathbb{S}|}]). \quad (13)$$

The proof is given in Appendix-D. We then have the following remark.

Remark 6: Suppose that there are a large fixed number of sessions in a given network, i.e., $|\mathbb{S}|$ is a large constant. We have the following extreme case: when $|J_s|\to\infty$, $[1-(1-\frac{2}{|J_s|})^{|\mathbb{S}|}]\to 0$, making the fluctuation bound approach 0. Generally, a larger $|J_s|$ makes the term $1-(1-\frac{2}{|J_s|})^{|\mathbb{S}|}$ become smaller, leading to a smaller fluctuation bound.

VI. Performance Evaluation

A. Methodology and Simulation Settings

It is worth noting that, the proposed algorithm is to be embedded to the controller module of SDNs. However, to our best knowledge, the existing SDN emulator Mininet does not support the evaluation of control plane traffics in the in-band SDNs. Therefore, we have implemented a simulator in Python to emulate an SDN with in-band control and conduct the evaluation of routing protection for control plane traffics.

Benchmarks: Four benchmarks are used to compare the performance with the proposed Alg. 1.

As the first, the ***K-shortest*** path finding algorithm [39], [52], [53] is a classical static heuristic, in which each session is provisioned with the first K shortest paths from the given candidate path set. Note that, in our simulation $K=|D_{s\in\mathbb{S}}|$.

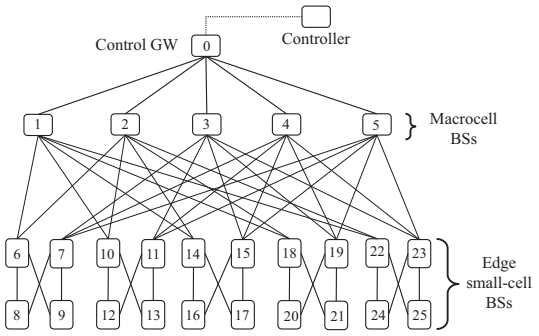


Fig. 5. A 26-node Fattree-like mesh topology for in-band control-plane used in simulation.

The second one is called ***Alg. Iterative*** [54], and also based on the Markov approximation technique. In ***Alg. Iterative***, although the system is similarly allowed to transit from one configuration to another by swapping a pair of paths, the transition rate is designed as $q_{f\to f'}\propto\exp^{-1}(-\beta u_{f'})$. Furthermore, it keeps tracing the best configuration observed so far which is used as the final solution.

The third one is *Local Rerouting* (***Alg. LR***), which is widely adopted in literature [15], [18], [20], [26]–[29]. When applying Alg. LR in our simulation, once a link fails, the affected traffic is rerouted to neighbouring links in a fashion of *detour* [28], [29].

Finally, the ***Optimal*** solutions will be obtained by utilizing the state-of-the-art optimizer Gurobi [55]. The solvers in the Gurobi Optimizer are designed from the ground up to exploit modern architectures and multi-core processors, using the most advanced implementations of the latest algorithms. The Gurobi Optimizer also supports a variety of programming and modeling languages including: Object-oriented interfaces for C++, Java, .NET, and Python. For the comparison with the proposed Alg. 1, we solve the integer programming **CPRP** problem using the Python-oriented application programming interfaces, under the circumstances both before and after a link failure.

Other settings and metrics: Simulations are conducted under the online dynamic case with occurrence of a single-link failure. The traffic demand of each controller-to-device session and the link bandwidth capacity are randomly generated within a given range. Before executing the algorithms, by invoking a simple depth-first path finding algorithm, we try to provide each controller-to-device session $s\in\mathbb{S}$ with a number $|J_s|$ of candidate paths with *low link-shared degree* under a Fattree-like mesh topology. Having fixed $\omega=1$, $\tau=1$ and $\beta=10$, we select a number $|D_s|=2$ of paths as in-use paths by executing the proposed algorithm and the other benchmarks.

The weighted ***Joint system cost*** is defined as the sum of both the ***Largest link overhead*** measured in traffic rate (Mb/s) and the ***Average (Avg) node configure overhead*** measured in the average configuration times at each device node.

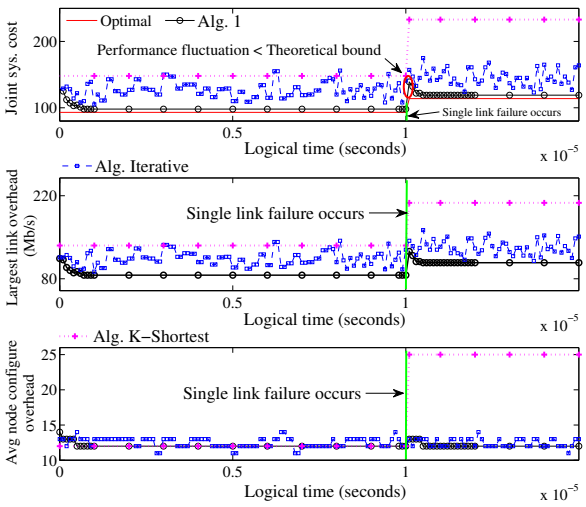


Fig. 6. Representative execution of algorithms under the mesh topology (initial $|J_s|$=5, $|D_s|$=2, $\forall s \in \mathbb{S}$). It can be seen that Alg. 1 converges in both initial stage and after the link failure. Note that, the numerical ***Joint numerical system cost*** includes both the ***Largest link overhead*** measured by the traffic rate (Mb/s) and the ***Average (Avg) node configure overhead*** measured by the average configuring times in each device node.

B. Representative Execution Case of Algorithms

The first group of simulations is conducted under a Fattree-like mesh topology with 26 nodes and 50 bidirectional links (shown in Fig. 5). In this topology, the controller directly connects to gateway node 0, and indirectly connects to other device nodes via in-band connections. The control-plane traffic demand of each session is randomly generated within a range [1, 15] Mb/s, and the link bandwidth capability for both control-plane and date-plane traffic in each link is set at 1000 Mb/s.

By fixing $|J_s|$=5 and $|D_s|$=2 for each session $s \in \mathbb{S}$, the numerical result shown in Fig. 6 illustrates the representative execution case of algorithms over a logical period in 15 microseconds. A single-link failure occurs in link (0,3) at the 10^{th} microsecond. As link (0,3) is one of the most critical links under the mesh topology, its failure brings a worst damage that 20% of both the total candidate paths and in-use paths cannot be used anymore.

Before a link-failure, we observe that the proposed algorithm converges to an optimal solution with a cost 93 in the initial 0.5 microseconds. In contrast, K-shortest algorithm keeps a high system cost 141. Although the best solution of *Alg. Iterative* is traced around the 5.5^{th} microsecond, this algorithm shows a fluctuant performance all the time. Thus it is not clear when the best solution can be obtained. On contrary, the near-optimal solution can be traced quickly using the proposed Alg. 1 because of its fast convergence. As shown in Fig. 6, at 0.8^{th} microsecond in the simulation, we get the converged near-optimal solution.

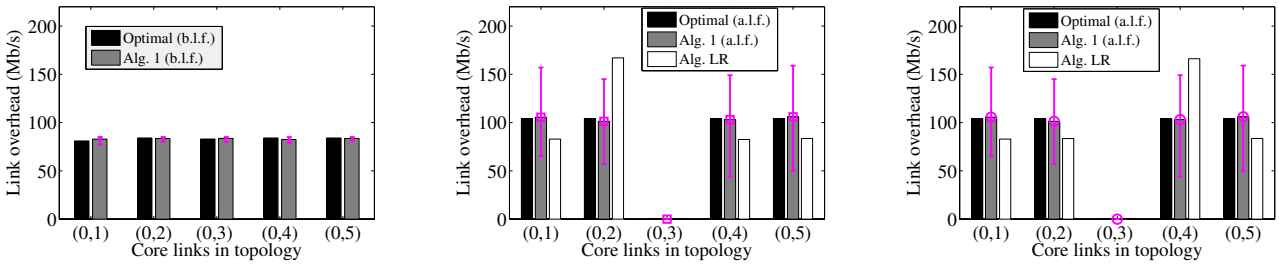
When the single-link failure occurs, the cost fluctuation is observed for all algorithms. The failed link shrinks the candidate path set and in-use path set for each session. As a result, the updated in-use paths are selected by sharing more common links, making the total system cost grows. Comparing the fluctuation gaps among the mentioned algorithms, we find that the proposed algorithm outperforms the other two algorithms. Moreover, its performance converges to the optimal solution quickly at 10.5^{th} microsecond. However, the *K-shortest* still holds the highest cost 163 and *Alg. Iterative* shows a more severe perturbation. On the other hand, right after the link-failure occurs, the cost of *Alg. 1* is shown as 148, and the converged cost is 117, leading to the fluctuation gap is 31. According to Theorem 3, the theoretical fluctuation bound is 191.7 using (13). The fluctuation gap is within the theoretical bound under the failure of link (0,3).

Furthermore, in the same group of simulation shown in Fig. 6, the middle and bottom figures demonstrate the largest link overhead (in terms of aggregated control-plane traffic rate in Mb/s, the average node configuration overhead (in terms of times of configuration), respectively. Note that, the joint numerical system cost is calculated by summing the two overhead terms that are shown in the bottom two sub-figures. Overall, both the two terms of costs show the similar performance with the joint system cost. However, in the bottom figure with respect to the average node configure overhead, we observe that the *K-shortest* algorithm interestingly maintains at a quite low level. This is because the *K-shortest* algorithm always selects the shortest $|D_s|$ candidate paths for each session. Consequently, the total length of the selected paths is very short. After the link failure, because the candidate path space becomes smaller than before, some longer paths have to be chosen, making the average node overhead become higher. For example, we observe that the average node overhead of *K-shortest* increases from 11 to 25 when a single link failure occurs.

C. Case Study of Single Link Failure

With the same single link failure in the critical core link (0,3) at the 10^{th} microsecond, we compare the link overhead in terms of the aggregated control-plane traffic rate in other core links such as (0,1), (0,2), (0,4) and (0,5) of the mesh topology. Note that the *Alg. LR* can be evaluated only after link failure happens. For a fair comparison with our *Alg. 1*, we apply *Alg. LR* based on the converged routing solution yielded by *Alg. 1* just before the link failure. Particularly, when the core link (0,3) fails, the affected traffic will be rerouted via its neighboring links (0,2) and (0,4), respectively. In the following, we show the numerical results of both situations based on the average performance obtained over 100 instances.

At the first, as shown in Fig. 7(a), the link overhead of the converged solution obtained by our *Alg. 1* is very close to the optimal solution, before link failure. When link (0,3) fails, we can observe the performance of *Alg. LR* from Fig. 7(b) that the control-plane traffic rate in the core link (0,2) increases sharply from 80 Mb/s to around 165 Mb/s, if the affected traffic is rerouted only via link (0,2). In contrast, the average performance of *Alg. 1* is still very close to the *Optimal* one. This is because they amortize the affected traffics to the other 4 core links. The similar results can be observed in Fig. 7(c)

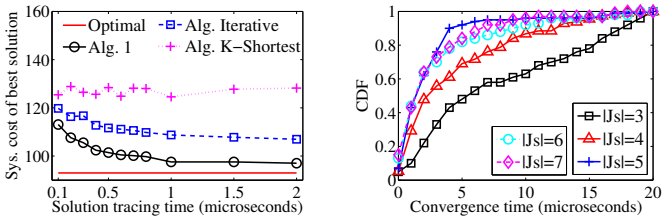


(a) Before link failure (shorten as b.l.f.), the converged link overhead obtained by *Alg. 1* shows very close to the optimal solution found by *Gurobi* [55].

(b) In *Alg. LR*, when link (0,3) fails, the affected traffic flows are rerouted via the neighbouring link (0,2). Here only the control-plane traffics a.l.f. are shown.

(c) In *Alg. LR*, when link (0,3) fails, the affected traffic flows are rerouted via the neighbouring link (0,4). Here only the control-plane traffics a.l.f. are shown.

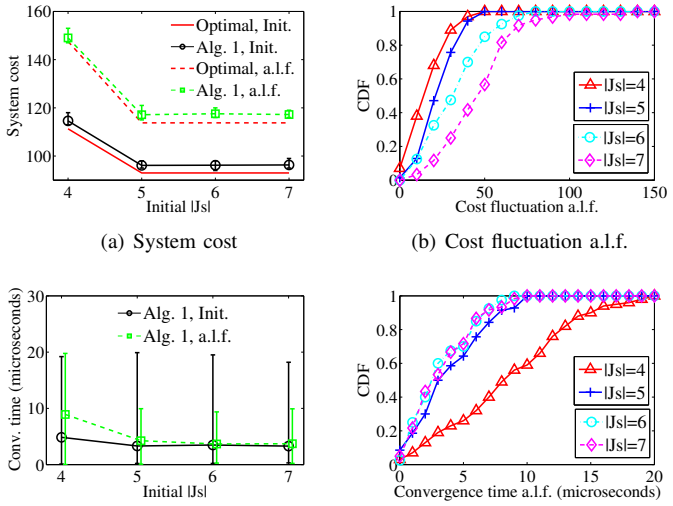
Fig. 7. Link overhead distribution in the core links before/after the critical link (0,3) fails. *Alg. 1* always shows near-optimal performance in terms of the aggregated traffic rates in the core links. *Alg. LR* exhibits sharp increasing aggregated traffic rate in the neighboring links of the failed one.



(a) System cost of algorithms while varying the best-solution tracing time

(b) Convergence time

Fig. 8. Convergence property of algorithms. Alg. 1 shows overwhelming performance over benchmark algorithms.



(a) System cost

(b) Cost fluctuation a.l.f.

(c) Convergence time

(d) CDF of convergence time a.l.f.

Fig. 9. Performance of Alg. 1 when varying the number of initial candidate paths for each session in the initial stage (shorten as Init.) and after-link-failure (shorten as a.l.f.). We find that although a larger candidate path set increases the cost fluctuation, it reduces the convergence time after the link failure.

when the affected traffic is locally rerouted over the core link (0,4).

D. Performance of Alg. 1 in the Initial Stage

Now, we study the performance of algorithms in the initial connection-setup stage of control-plane traffic. In this group of simulation, each parameter setting is evaluated with 100 instances. By fixing $|J_s| = 5$, $|D_s| = 2$ ($\forall s \in \mathbb{S}$) and varying the solution tracing time from 0.1 to 2 microseconds, we record the cost of the best solution of all algorithms. From Fig. 8(a), we observe that the performance of all algorithms are similar when the solution-tracing time is 0.1 microsecond. However, the proposed *Alg. 1* demonstrates the significant advantage over the two benchmarks *Alg. Iterative* and *Alg. K-Shortest* when the solution-tracing time grows, and it converges to the optimal solution when tracing time exceeds 1 microsecond.

Since convergence only occurs when using *Alg. 1*, we further study the cumulative distribution function (CDF) of its convergence time, by varying the candidate path scale $|J_s|$ within $\{3, 4, 5, 6, 7\}$. The results are shown in Fig. 8(b), from which it can be observed that approximate 45% of convergence times are within the first microsecond and over 90% of convergence times are less than 5 microseconds when $|J_s| \geq 5$. On the other hand, the near-optimal solution yielded by *Alg. 1* is hard to find while $|J_s| < 5$, leading to that the increase on convergence time accordingly.

E. Performance of Alg. 1 under Single-Link Failure

Next, we study the impact introduced by a single-link failure while applying *Alg. 1*. Based on the same settings as the former group of simulation, we compare the performance of Alg. 1 between the initial convergence stage (Init.) and the stage after-link-failure (a.l.f.).

The average system costs of Alg. 1 in both stages are compared in Fig. 9(a). Here $|J_s|$ is the number of candidate paths before a single-link failure, and varies from 4 to 7 due to the adopted '1+1' protection. We see that there is an increment of approximate 20 units of the system cost with each $|J_s|$ under a.l.f. case. Fig. 9(b) shows the CDF of system cost fluctuation a.l.f. of Alg. 1. We surprisingly find out that the fluctuation is proportional to $|J_s|$. For example, 90% of the cost fluctuation is less than 30 when $|J_s| = 4$, but this percentage reduces to 75%, 48% and 25% when $|J_s|$ =5, 6 and 7, respectively. Here, the probability of having very bad configurations becomes higher at the time-slot when the link

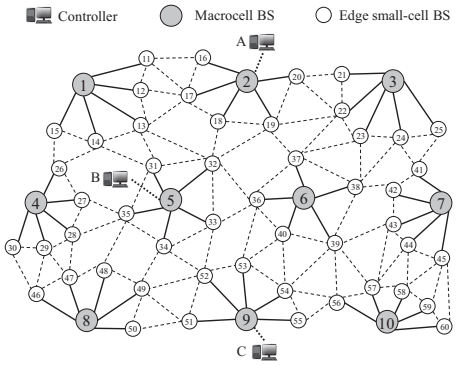


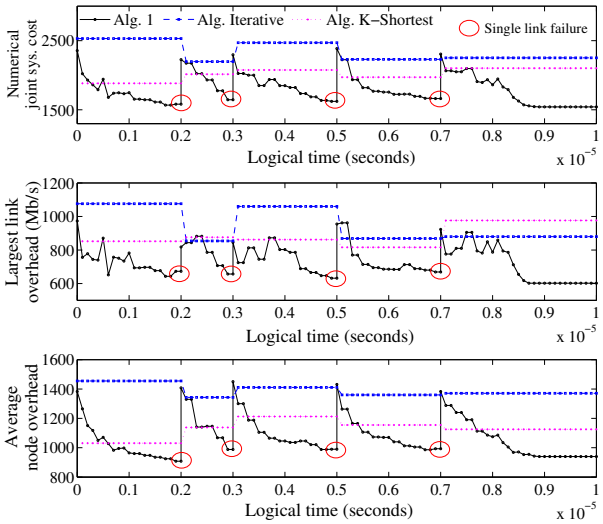
Fig. 10. A 60-node HetNet topology for in-band control-plane used in another group of simulation to examine random link failures. Particularly, we deploy 3 controllers A, B and C, which directly connects to Macrocell BS 2, 5 and 9, respectively.

failure occurs, under the situation that more candidate paths are being provided. However, all the observed cost fluctuations are within the theoretical bound described in Theorem 3. Then, in Fig. 9(c), the convergence time a.l.f. is observed larger than that in the initial stage. This is because the decreased candidate paths make the best configuration found in a longer time. We also see that the average convergence time is less than 5 microseconds when $|J_s| \geq 5$. Further, Fig. 9(d) illustrates the CDFs of convergence time a.l.f.. We see that the convergence times are almost same when $|J_s| = 5, 6, 7$ and better than that under $|J_s| = 4$. It can be seen that although a larger candidate path set increases the cost fluctuation, it benefits the convergence time after link failure.

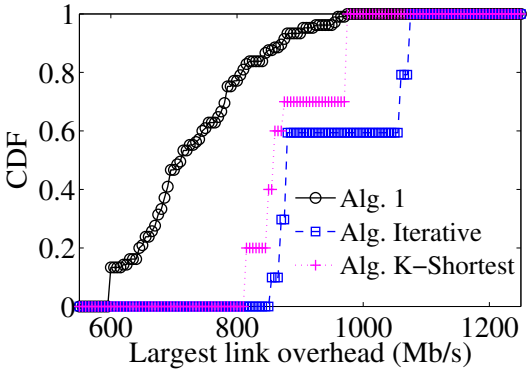
F. Impact of Multiple Random Single-Link Failures

Another set of simulations is conducted under a HetNet topology with 10 Macrocell BSs, 50 small-cell BSs and 144 bidirectional links shown in Fig. 10. We deploy 3 controllers A, B and C, which directly connect to Macrocell BS 2, 5 and 9, respectively. We specify that A, B and C are in charge of configuring BSs 1-25, 26-45, and 46-60, respectively. Thus, the number of controller-to-BS sessions is 57. The traffic rate of each control-plane session is randomly generated within a range in [10, 100] Mb/s, and the available link bandwidth capability for control-plane traffic in each link is set to 5 Gb/s. In order to examine the robustness of algorithms under multiple random single link failures, we then generate a link failure trace in an interval in 10 microseconds, during which each link may fail with a probability 0.3 per logical microsecond.

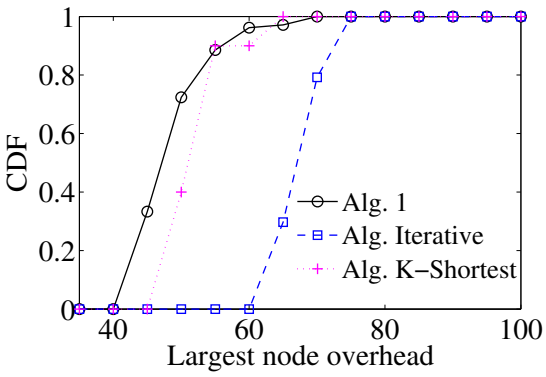
We still provide $|J_s|$=5 candidate paths for each session, each of which needs to select $|D_s|$=2 in-use paths. In particular, we tune the weight factor ω to 100 to increase the weight of node overhead. Fig. 11(a) demonstrates the performance of the three algorithms. We can observe that each single link failure causes huge fluctuations to all the metrics for each algorithm. Compared with *Alg. Iterative* and *K-Shortest*, *Alg. 1* converges quickly after each link failure and achieves 10%-50% lower converged cost/overhead than that of the other algorithms. Meanwhile, Fig. 11(b) and 11(c) present the CDFs of the



(a) The traced joint numerical system cost as well as overhead on links and nodes. We observe that Alg. 1 converges quickly after each random link failure and achieves 10%-50% lower converged cost/overhead than benchmarks.



(b) CDF of largest link overheads.



(c) CDF of largest node overheads.

Fig. 11. Representative execution of algorithms under a HetNet topology with 57 sessions ($|J_s|$=5, $|D_s|$=2). Particularly, the single link failure randomly occurs per microsecond with a certain probability.

traced largest link overhead and node overhead, respectively. It can be seen that *Alg. K-Shortest* performs better than the *Alg. Iterative* on both overheads, and *Alg. 1* yields the most efficient resource utilization in terms of the lowest link bandwidth

consumption and node configuration cost.

In summary, from the numerical results shown so far, we have the following observations: 1) The proposed *Sojourn-and-Transit* algorithm (*Alg. 1*) has a near-optimal performance; 2) *Alg. 1* has higher resource utilization efficiency than the conventional *Local Rerouting algorithm*; 3) *Alg. 1* achieves 10%-50% lower overhead in terms of the link bandwidth consumption and the connection setup cost, as compared with the two benchmark algorithms, *Alg. Iterative* and *Alg. K-Shortest*.

VII. Conclusion and Future Work

In this paper, to ensure fast recovery of the control-plane traffic in in-band fashioned Software-Defined HetNets, we studied a joint weighted cost-minimization problem. The traffic load balancing and control-channel setup cost are jointly considered when performing routing protection for control-plane traffic. To solve this multiple resource-constrained routing problem, we proposed a near-optimal solution, using the Markov approximation technique. We also extend the proposed approach to the online case that can promptly handle a single-link failure. The induced performance fluctuation is also studied with theoretical derivation. Finally, extensive simulations are conducted to evaluate the performance of the proposed approach. Compared with existing benchmarks, the proposed algorithm shows much better performance in terms of both resource utilization and the robustness on handling a single-link failure. As part of the future work, we shall apply our framework to the data plane of SDNs. Some other recovery mechanisms, such as restoration and cold-backup protection, will be further studied.

References

[1] R. Madan, J. Borran, A. Sampath, N. Bhushan, A. Khandekar, and T. Ji, "Cell association and interference coordination in heterogeneous lte-a cellular networks," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 9, pp. 1479–1489, 2010.

[2] A. Khandekar, N. Bhushan, J. Tingfang, and V. Vanghi, "Lte-advanced: Heterogeneous networks," in *Proc. IEEE European Wireless Conference*, 2010, pp. 978–982.

[3] A. Damnjanovic, J. Montojo, Y. Wei, T. Ji, T. Luo, M. Vajapeyam, T. Yoo, O. Song, and D. Malladi, "A survey on 3gpp heterogeneous networks," *IEEE Wireless Communications*, vol. 18, no. 3, pp. 10–21, 2011.

[4] S. Parkvall, A. Furuskär, and E. Dahlman, "Evolution of lte toward imt-advanced," *IEEE Communications Magazine*, vol. 49, no. 2, pp. 84–91, 2011.

[5] L. Le and E. Hossain, "Multihop cellular networks: Potential gains, research challenges, and a resource allocation framework," *IEEE Communications Magazine*, vol. 45, no. 9, pp. 66–73, 2007.

[6] M. Salem, A. Adinoyi, H. Yanikomeroglu, and D. Falconer, "Opportunities and challenges in ofdma-based cellular relay networks: A radio resource management perspective," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 5, pp. 2496–2510, 2010.

[7] A. B. Saleh, S. Redana, J. Hämäläinen, and B. Raaf, "On the coverage extension and capacity enhancement of inband relay deployments in lte-advanced networks," *Journal of Electrical and Computer Engineering*, vol. 2010, pp. 1–12, 2010.

[8] "Openflow switch specification," https://www.opennetworking.org/sdn-resources/technical-library, April 2015.

[9] S. Agarwal, M. Kodialam, and T. Lakshman, "Traffic engineering in software defined networks," in *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, April 2013, pp. 2211–2219.

[10] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Avant-guard: Scalable and vigilant switch flow management in software-defined networks," in *Proc. ACM SIGSAC conference on Computer & communications security*, 2013, pp. 413–424.

[11] U. C. Kozat, G. Liang, and K. Kokten, "On diagnosis of forwarding plane via static forwarding rules in software defined networks," in *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, 2014.

[12] M. L. Cing-Yu Chu, Kang Xi and H. J. Chao, "Congestion-aware single link failure recovery in hybrid sdn networks," in *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, April 2015.

[13] A. Bianco, P. Giaccone, A. Mahmood, M. Ullio, and V. Vercellone, "Evaluating the sdn control traffic in large isp networks," in *Proc. IEEE International Conference on Communications (ICC)*, June 2015, pp. 5248–5253.

[14] R. Ahmed and R. Boutaba, "Design considerations for managing wide area software defined networks," *IEEE Communications Magazine*, vol. 52, no. 7, pp. 116–123, 2014.

[15] A. Sgambelluri, A. Giorgetti, F. Cugini, F. Paolucci, and P. Castoldi, "Openflow-based segment protection in ethernet networks," *Journal of Optical Communications and Networking*, vol. 5, no. 9, pp. 1066–1075, 2013.

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

[17] S. S. Lee, K.-Y. Li, K. Y. Chan, J.-H. YwiChi, T.-W. Lee, W.-K. Liu, and Y.-J. Lin, "Design of sdn based large multi-tenant data center networks," in *Proc. IEEE 4th International Conference on Cloud Networking (CloudNet)*, 2015, pp. 44–50.

[18] N. Beheshti and Y. Zhang, "Fast failover for control traffic in software-defined networks," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, 2012, pp. 2665–2670.

[19] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "Fast failure recovery for in-band openflow networks," in *Proc. 9th international conference on the Design of reliable communication networks (DRCN)*, 2013, pp. 52–59.

[20] Y. Hu, W. Wendong, G. Xiangyang, C. H. Liu, X. Que, and S. Cheng, "Control traffic protection in software-defined networks," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, 2014, pp. 1878–1883.

[21] A. Detti, C. Pisa, S. Salsano, and N. Blefari-Melazzi, "Wireless mesh software defined networks (wmsdn)," in *Proc. IEEE 9th international conference on wireless and mobile computing, networking and communications (WiMob)*, 2013, pp. 89–95.

[22] S. Salsano, G. Siracusano, A. Detti, C. Pisa, P. L. Ventre, and N. Blefari-Melazzi, "Controller selection in a wireless mesh sdn under network partitioning and merging scenarios," *arXiv preprint arXiv:1406.2470*, 2014.

[23] K. Seppänen, J. Kilpi, and T. Suihko, "Integrating wmn based mobile backhaul with sdn control," *Mobile Networks and Applications*, vol. 20, no. 1, pp. 32–39, 2015.

[24] H. Huang, P. Li, S. Guo, and W. Zhuang, "Software-defined wireless mesh networks: architecture and traffic orchestration," *IEEE Network*, vol. 29, no. 4, pp. 24–30, 2015.

[25] D. Turner, K. Levchenko, A. C. Snoeren, and S. Savage, "California fault lines: understanding the causes and impact of network failures," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 315–326, 2011.

[26] S. S. Lee, P.-K. Tseng, A. Chen, and C.-S. Wu, "Non-weighted interface specific routing for load-balanced fast local protection in ip networks," in *Proc. IEEE International Conference on Communications (ICC)*, 2011, pp. 1–6.

[27] C. Cascone, L. Pollini, D. Sanvito, and A. Capone, "Traffic management applications for stateful sdn data plane," in *Proc. Fourth European Workshop on Software Defined Networks (EWSDN)*, 2015, pp. 85–90.

[28] A. Capone, C. Cascone, A. Q. Nguyen, and B. Sanso, "Detour planning for fast and reliable failure recovery in sdn with openstate," in *Proc. 11th International Conference on the Design of Reliable Communication Networks (DRCN)*, 2015, pp. 25–32.

[29] P. M. Mohan, T. Truong-Huu, and M. Gurusamy, "Tcam-aware local rerouting for fast and efficient failure recovery in software defined networks," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, 2015.

[30] T. Mizrahi and Y. Moses, "On the necessity of time-based updates in sdn," in *Open Networking Summit (ONS)*, 2014.

[31] ——, "Software defined networks: It's about time," in *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, April 2016, pp. 1–9.

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

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

[34] B. Niven-Jenkins, D. Brungard, M. Betts, N. Sprecher, and S. Ueno, "Requirements of an mpls transport profile," Tech. Rep. IETF RFC 5654, 2009.

[35] M. Mendonca, K. Obraczka, and T. Turletti, "The case for software-defined networking in heterogeneous networked environments," in *Proc. ACM conference on CoNEXT student workshop*, 2012, pp. 59–60.

[36] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the internet-of-things," in *Proc. IEEE Network Operations and Management Symposium (NOMS)*, 2014, pp. 1–9.

[37] R. J. Edell, N. McKeown, and P. P. Varaiya, "Billing users and pricing for tcp," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1162–1175, 1995.

[38] Z. Wang and J. Crowcroft, "Qos routing for supporting resource reservation," *IEEE Journal on Selected Areas in Communications, September*, 1996.

[39] X. Yuan, "Heuristic algorithms for multi-constrained quality-of-service routing," *IEEE/ACM Transactions on Networking*, vol. 10, no. 2, pp. 244–256, 2002.

[40] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6301–6327, 2013.

[41] K. T. K. Cheung, S. Yang, and L. Hanzo, "Achieving maximum energy-efficiency in multi-relay ofdma cellular networks: a fractional programming approach," *IEEE Transactions on Communications*, vol. 61, no. 7, pp. 2746–2757, 2013.

[42] A. BenMimoune, F. A. Khasawneh, M. Kadoch, S. Sun, and B. Rong, "Inter-cell handoff performance improvement in lte-a multi-hop relay networks," in *Proc. 12th ACM international symposium on mobility management and wireless access*, 2014, pp. 9–13.

[43] A. BenMimoune, F. A. Khasawneh, M. Kadoch, and B. Rong, "Resource allocation framework in 5g multi-hop relay system," in *Proc. IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.

[44] N. Sapountzis, T. Spyropoulos, N. Nikaein, and U. Salim, "Optimal downlink and uplink user association in backhaul-limited hetnets," in *Proc. IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2016.

[45] M. Borokhovich, L. Schiff, and S. Schmid, "Provable data plane connectivity with local fast failover: Introducing openflow graph algorithms," in *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, 2014, pp. 121–126.

[46] D. Adami, S. Giordano, M. Pagano, and N. Santinelli, "Class-based traffic recovery with load balancing in software-defined networks," in *Proc. IEEE Global Communications Conference Workshops*, 2014, pp. 161–165.

[47] S. Ramamurthy and B. Mukherjee, "Survivable wdm mesh networks. part i-protection," in *Proc. Eighteenth Proc. IEEE International Conference on Computer Communications (INFOCOM)*, vol. 2, 1999, pp. 744–751.

[48] K. Walkowiak, M. Klinkowski, B. Rabiega, and R. Goścień, "Routing and spectrum allocation algorithms for elastic optical networks with dedicated path protection," *Optical Switching and Networking*, vol. 13, pp. 63–75, 2014.

[49] C. Huang, V. Sharma, K. Owens, and S. Makam, "Building reliable mpls networks using a path protection mechanism," *IEEE Communications Magazine*, vol. 40, no. 3, pp. 156–162, 2002.

[50] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[51] P. Diaconis and D. Stroock, "Geometric bounds for eigenvalues of markov chains," *The Annals of Applied Probability*, pp. 36–61, 1991.

[52] D. M. Topkis, "A k shortest path algorithm for adaptive routing in communications networks," *IEEE Transactions on Communications*, vol. 36, no. 7, pp. 855–859, 1988.

[53] M. R. Rahman and R. Boutaba, "Svne: Survivable virtual network embedding algorithms for network virtualization," *IEEE Transactions on Network and Service Management*, vol. 10, no. 2, pp. 105–118, 2013.

[54] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint vm placement and routing for data center traffic engineering," in *Proc. IEEE International Conference on Computer Communications. (INFOCOM)*, 2012, pp. 2876–2880.

[55] G. Optimization, "Gurobi optimizer reference manual," 2016.

[56] F. P. Kelly, *Reversibility and stochastic networks*. Cambridge University Press, 2011.

[57] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov chains and mixing times*. American Mathematical Soc., 2009.

[58] K. L. Judd, "The law of large numbers with a continuum of iid random variables," *Journal of Economic theory*, vol. 35, no. 1, pp. 19–25, 1985.

APPENDIX A
PROOF OF THEOREM 1

Proof: By the two conditions for state space of constructing the designed Markov chain, we see that all configurations can reach each other within a finite number of transitions, where a single in-use path is replaced in each transition. Therefore, the constructed Markov chain is irreducible, i.e., an ergodic Markov chain. Now we show that the stationary distribution of the constructed Markov chain is exactly (4).

In the proposed *Sojourn-and-Transit* algorithm, the sojourn time of each configuration is exponentially distributed and the transition probability between different configurations is independent of time, so the state process constitutes a homogeneous continuous-time Markov chain. Let $Pr_{f\to f'}$ denote the probability that system will enter state $f^{'}$ when it leaves state f due to expiration of a count-down timer. We also introduce N_f indicating the set of state which are directly connected to a state f. In *Sojourn-and-Transit* algorithm, the next state of f has equal probability to be any state $f^{'}$, $\forall f^{'} \in N_f$. It is not hard to know the size of state space N_f is $|N_f| = \sum_{s\in\mathbb{S}} |D_s|(|J_s| - |D_s|)$, where s is the critical session which induces the transition from f to $f^{'}$. Then, we have

$$Pr_{f\to f'} = \frac{1}{|N_f|} = \frac{1}{|\mathbb{S}||D_s|(|J_s| - |D_s|)}, \forall f \in \mathcal{F}, \forall f^{'} \in N_f.$$

In the following, we show that the state transition rate from f to $f^{'}$ in the implemented *Sojourn-and-Transit* algorithm satisfies (5).

- Firstly, all the transition rates of the paths selection process are finite;
- Secondly, it is straightforward to see that all path configurations can reach each other via a finite number of transitions. Therefore, the constructed Markov chain is irreducible.
- Finally, the detailed balance equation holds between any two adjacent states. Because according to (6), given a current state f, each session $s \in \mathbb{S}$ counts down with a rate

$$\lambda_s = \frac{1}{\exp(\tau)} \cdot \frac{|D_s|(|J_s| - |D_s|)}{\exp(\frac{1}{2}\beta \sum_{s\in\mathbb{S}}(u_s(f') - u_s(f)))},$$

then, the system leaves state f with a rate $\sum_{s\in\mathbb{S}} \lambda_s$. With probability $Pr_{f\to f'}$, system jumps to an adjacent state $f^{'}$ when leaving the current state f. Therefore, we calculate

the transition rate from f to $f^{'}$ as follows:

$$\begin{aligned} q_{f,f'} &= \sum_{s\in\mathbb{S}} \lambda_s \times Pr_{f\to f'} \\ &= \frac{1}{\exp(\tau)} \cdot \frac{\sum_{s\in\mathbb{S}} |D_s|(|J_s|-|D_s|)}{\exp(\frac{1}{2}\beta \sum_{s\in\mathbb{S}}(u_s(f')-u_s(f)))} \\ &\quad \times \frac{1}{\sum_{s\in\mathbb{S}} |D_s|(|J_s|-|D_s|)} \\ &= \frac{1}{\exp(\tau)} \cdot \exp(\frac{1}{2}\beta \sum_{s\in\mathbb{S}}(u_s(f)-u_s(f^{'}))). \end{aligned} \tag{14}$$

Finally, using (14) and (4), we can see that $p^*_f q_{f,f'} = p^*_{f'} q_{f',f}, \forall f, f^{'} \in \mathcal{F}$, i.e., the detailed balance equations hold. According to the Theorem 1.3 and Theorem 1.14 in [56], the constructed Markov chain is time-reversible and its stationary distribution is (4). ∎

Appendix B
Proof of Theorem 2

Proof: Following the framework proposed in [40], and based on the analysis method [51], [57], we conduct the proof for the lower bound and upper bound of mixing time.

At the first, the constructed MC is a continuous-time Markov chain with a stationary distribution given by (4). Because $\sum_{f'\in\mathcal{F}} \exp(\beta u_{f'}) \le |\mathcal{F}| \exp(\beta u_{\max})$ and $|\mathcal{F}| \le \prod_{s\in\mathbb{S}} \binom{|J_s|}{|D_s|}$, the minimal probability in the stationary distribution should be:

$$\begin{aligned} p_{\min} := \min_{f\in\mathcal{F}} p^*_f &\ge \frac{\exp(\beta u_{\min})}{|\mathcal{F}| \exp(\beta u_{\max})} \\ &\ge \frac{1}{\varpi} \exp(-\beta(u_{\max}-u_{\min})) \end{aligned} \tag{15}$$

We conduct the remaining proof utilizing the *uniformization* technique [40]. Denote $Q = \{q_{f,f'}\}$ as the transition rate matrix of the MC. We then construct a discrete-time Markov chain $Z(n)$ with a probability transition matrix $P = I + \frac{Q}{\theta}$, where I is the *unit matrix* and θ is the *uniform rate parameter*. Then let's consider a system that the successive states visited form a Markov chain $Z(n)$, where the times when system changes its state, constitute an independent Poisson process $N(t)$ with rate θ. Let $Z(N(t))$ denote the state of this system at discrete time t.

Note that, since $\forall f, f' \in \mathcal{F}$, $q_{f,f'} \le \exp(\frac{1}{2}\beta(u_{f'}-u_f)-\tau) \le \exp(\frac{1}{2}\beta(u_{\max}-u_{\min})-\tau)$, and f can at most transit to $\zeta = \sum_{s\in\mathbb{S}} |D_s| \cdot (|J_s|-|D_s|)$ other states, thus $\sum_{f\ne f'} q_{f,f'} \le \zeta \exp(\frac{1}{2}\beta(u_{\max}-u_{\min})-\tau)$. Therefore, θ is given as:

$$\theta = \zeta \exp(\frac{1}{2}\beta(u_{\max}-u_{\min})-\tau). \tag{16}$$

Next, due to the uniformization theorem [57], the MC and its discrete-time counterpart $Z(N(t))$ has the same distribution. Furthermore, they also share the same stationary distribution. Let ρ_2 denotes the second largest eigenvalue of transition matrix P for $Z(n)$. Then, applying the spectral gap inequality [51], [57], we have:

$$\begin{aligned} \frac{\exp(-\theta(1-\rho_2)t)}{2} &\le \max_{f\in\mathcal{F}} \|\boldsymbol{H}_t(f) - \boldsymbol{p}^*\|_{TV} \\ &\le \frac{\exp(-\theta(1-\rho_2)t)}{2(p_{\min})^{\frac{1}{2}}}. \end{aligned}$$

Therefore,

$$\begin{aligned} \frac{1}{\theta(1-\rho_2)} \ln\frac{1}{2\epsilon} &\le t_{mix}(\epsilon) \\ &\le \frac{1}{\theta(1-\rho_2)}[\ln\frac{1}{2\epsilon} + \frac{1}{2}\ln\frac{1}{p_{\min}}]. \end{aligned} \tag{17}$$

Particularly, ρ_2 can be bounded by Cheeger's inequality [51], [57] as:

$$1 - 2\Phi \le \rho_2 \le 1 - \frac{1}{2}\Phi^2. \tag{18}$$

where Φ is the "conductance" of P, and defined as follows:

$$\Phi := \min_{N\subset\mathcal{F}, \pi_N\in(0,1/2]} \frac{F(N,N^c)}{\pi_N}. \tag{19}$$

Here $\pi_N = \sum_{f\in N} p^*_f$ and $F(N,N^c) = \sum_{f\in N, f'\in N^c} p^*_f P(f,f')$.

The combination of (17) and (18) yields

$$\frac{1}{2\theta\Phi}\ln\frac{1}{2\epsilon} \le t_{mix}(\epsilon) \le \frac{2}{\theta\Phi^2}[\ln\frac{1}{2\epsilon} + \frac{1}{2}\ln\frac{1}{p_{\min}}]. \tag{20}$$

Then, we derive the upper bound of Φ: For any $N' \subset \mathcal{F}, \pi(N') \in (0, 1/2]$,

$$\begin{aligned} \Phi &= \min_{N\subset\mathcal{F}, \pi_N\in(0,1/2]} \frac{F(N,N^c)}{\pi_N} \\ &\le \frac{1}{\pi_{N'}} \sum_{f\in N, f'\in N'^c} p^*_f P(f,f') \\ &= \frac{1}{\pi_{N'}} \sum_{f\in N'} p^*_f \cdot (\sum_{f'\in N'^c} P(f,f')) \\ &\le \frac{1}{\pi_{N'}} \sum_{f\in N'} p^*_f \\ &= 1. \end{aligned} \tag{21}$$

Now we have the lower bound of $t_{mix}(\epsilon)$ by combining (16), (20) and (21):

$$\begin{aligned} t_{mix}(\epsilon) &\ge \frac{1}{2\theta}\ln\frac{1}{2\epsilon} \\ &= \frac{\exp[\tau - \frac{1}{2}\beta(u_{\max}-u_{\min})]}{2\zeta}\ln\frac{1}{2\epsilon}. \end{aligned} \tag{22}$$

Next, we derive the lower bound of Φ. When $q_{f,f'} \ne 0, \forall f, f' \in \mathcal{F}$, via the equation (5) in original manuscript we know that

$$q_{f,f'} = \exp(\frac{1}{2}\beta(u_{f'}-u_f)-\tau) \ge \exp(\frac{1}{2}\beta(u_{\min}-u_{\max})-\tau). \tag{23}$$

The combination of (19) and (23) outputs:

$$\begin{aligned}\Phi &\geq \min_{N\subset\mathcal{F},\pi_N\in(0,1/2]} F(N,N^c)\\ &\geq \min_{f\neq f',P(f,f')>0} F(f,f')\\ &= \min_{f\neq f',P(f,f')>0} p_f^* P(f,f')\\ &= \min_{f\neq f',P(f,f')>0} p_f^*\cdot\frac{q_{f,f'}}{\theta}\\ &\geq \frac{p_{\min}}{\theta}\cdot\exp(\frac{1}{2}\beta(u_{\min}-u_{\max})-\tau).\end{aligned} \tag{24}$$

Finally, through combining (20), (15), (16) and (24), we have the upper bound of mixing time:

$$\begin{aligned}&t_{mix}(\epsilon)\leq\frac{2}{\theta\Phi^2}[\ln\frac{1}{2\epsilon}+\frac{1}{2}\ln\frac{1}{p_{\min}}]\\ &\leq\frac{2\theta\exp(2\tau-\beta(u_{\max}-u_{\min}))}{p_{\min}^2}[\ln\frac{1}{2\epsilon}+\frac{1}{2}\ln\frac{1}{p_{\min}}]\\ &\leq 2\zeta\varpi^2\exp[\frac{3}{2}\beta(u_{\max}-u_{\min})+\tau]\cdot\\ &\quad[\ln\frac{1}{2\epsilon}+\frac{1}{2}\ln\varpi+\frac{1}{2}\beta(u_{\max}-u_{\min})].\end{aligned} \tag{25}$$

This concludes the proof. ∎

Appendix C
Proof of Lemma 1

Proof: By referring the derivation of (4), we know the stationary distribution of the *configurations* in $\hat{M}$ is shown as:

$$q_g^*=\frac{\exp(-\beta u_g)}{\sum_{g'\in\mathcal{G}}\exp(-\beta u_{g'})},\forall g\in\mathcal{G}. \tag{26}$$

Next, we analyze the distribution of configurations $g\in\mathcal{G}$ in M when link failure just occurs. As we see in the **Step 2** of the Alg. 4 when link failure occurs, if there is no any session losing an in-use path, before **Step 3** the distribution of $g\in\mathcal{G}$ in M is already known as:

$$p_g^*=\frac{\exp(-\beta u_g)}{\sum_{f'\in\mathcal{F}}\exp(-\beta u_{f'})},\forall g\in\mathcal{G}; \tag{27}$$

otherwise, when any in-use path is replaced, the distribution of $g\in\mathcal{G}$ in M will become bigger than p_g^*. That is, we have:

$$\hat{q}_g\geq p_g^*,\forall g\in\mathcal{G}. \tag{28}$$

By (26) and (28), we know:

$$\begin{aligned}q_g^*-\hat{q}_g&\leq\frac{\exp(-\beta u_g)}{\sum_{g'\in\mathcal{G}}\exp(-\beta u_{g'})}-\frac{\exp(-\beta u_g)}{\sum_{f'\in\mathcal{F}}\exp(-\beta u_{f'})}\\ &=\frac{\exp(-\beta u_g)}{\sum_{g'\in\mathcal{G}}\exp(-\beta u_{g'})}-\frac{\exp(-\beta u_g)}{\sum_{f'\in\mathcal{G}}\exp(-\beta u_{f'})+\sigma},\forall g\in\mathcal{G},\end{aligned} \tag{29}$$

where $\sigma=\sum_{\check{g}\in\mathcal{F}\backslash\mathcal{G}}\exp(-\beta u_{\check{g}})$.

Then, we calculate the $d_{TV}(\mathbf{q}^*,\hat{\mathbf{q}})$ as follows.

$$d_{TV}(\mathbf{q}^*,\hat{\mathbf{q}})=\frac{1}{2}\sum_{g\in\mathcal{G}}|q_g^*-\hat{q}_g|=\sum_{g\in\mathbf{A^o}}(q_g^*-\hat{q}_g), \tag{30}$$

where $\mathbf{A^o}\triangleq\{g\in\mathcal{G}:q_g^*\geq\hat{q}_g\}$, and $\mathbf{A^o}\subset\mathcal{G}$.

On the other hand, the system costs $u_{f\in\mathcal{F}}$ are independent to each other, and follow normal distribution. That is, $u_{f\in\mathcal{F}}$ are independent and identically distributed (i.i.d.) discrete random values and the expectation of system cost exists within the finite configuration space $\mathcal{F}$. Letting this expectation denote by μ, and according to the law of large numbers [58], we have $\sum_{f\in\mathcal{F}\backslash\mathcal{G}}\exp(-\beta u_f)=|\mathcal{F}\backslash\mathcal{G}|\exp(-\beta\mu)$ with probability 1 and $\sum_{f\in\mathcal{F}}\exp(-\beta u_f)=|\mathcal{F}|\exp(-\beta\mu)$ with probability 1. Thus, this yields

$$\begin{aligned}d_{TV}(\mathbf{q}^*,\hat{\mathbf{q}})&=\sum_{g\in\mathbf{A^o}}(q_g^*-\hat{q}_g)\\ &\leq\frac{\sum_{g\in\mathbf{A^o}}\exp(-\beta u_g)}{\sum_{g'\in\mathcal{G}}\exp(-\beta u_{g'})}-\frac{\sum_{g\in\mathbf{A^o}}\exp(-\beta u_g)}{\sum_{f'\in\mathcal{G}}\exp(-\beta u_{f'})+\sigma}\\ &\leq\frac{\sum_{g\in\mathcal{G}}\exp(-\beta u_g)}{\sum_{g'\in\mathcal{G}}\exp(-\beta u_{g'})}-\frac{\sum_{g\in\mathcal{G}}\exp(-\beta u_g)}{\sum_{f'\in\mathcal{G}}\exp(-\beta u_{f'})+\sigma}\\ &=1-\frac{\sum_{g\in\mathcal{G}}\exp(-\beta u_g)}{\sum_{f'\in\mathcal{G}}\exp(-\beta u_{f'})+\sum_{f\in\mathcal{F}\backslash\mathcal{G}}\exp(-\beta u_f)}\\ &=\frac{\sum_{f\in\mathcal{F}\backslash\mathcal{G}}\exp(-\beta u_f)}{\sum_{f'\in\mathcal{F}}\exp(-\beta u_{f'})}=\frac{|\mathcal{F}\backslash\mathcal{G}|\exp(-\beta\mu)}{|\mathcal{F}|\exp(-\beta\mu)}\\ &=\frac{|\mathcal{F}\backslash\mathcal{G}|}{|\mathcal{F}|}.\end{aligned}$$

This concludes Lemma 1(a). Now we prove Lemma 1(b). Since we only consider single-link failure and each session is provided with *non link-shared* candidate paths, the number of disappeared candidate path is at most one. That is, to each session $s\in S_1$, the size of its candidate path space turns from $|J_s|$ to $|J_s|-1$. Now, we compute the total number of configurations disappeared due to the single-link failure. By the definition of S_{im}, we know in the disappeared configurations, each session in S_{im} selects the disappeared candidate path and other $|D_s|-1$ candidate paths as its in-use paths. Consequently, the number of the path selection conditions for sessions in S_{im} is calculated as $c_1=\prod_{s\in S_{im}}\binom{|J_s|-1}{|D_s|-1}$. On the other hand, it is not hard to get the number of path selection conditions for sessions from $S_1\backslash S_{im}$ is calculated as $c_2=\prod_{s\in S_\perp}\binom{|J_s|-1}{|D_s|}$. Recall that, there is no influence to any session in $\mathbb{S}\backslash S_1$, because their candidate paths are not affected by the failed link. Similarly, the number of path selection conditions of all sessions in $\mathbb{S}\backslash S_1$ is $c_3=\prod_{s\in\mathbb{S}\backslash S_1}\binom{|J_s|}{|D_s|}$. In addition, it should be noticed that any session $s\in S_1$ can be a participant of S_{im}. Therefore, the S_{im} should be varied as $S_{im}\subseteq S_1$ in the calculation to include all possible solutions.

Finally, we can compute the $\frac{|\mathcal{F}\backslash\mathcal{G}|}{|\mathcal{F}|}$ as follows.

$$\begin{aligned}\frac{|\mathcal{F}\backslash\mathcal{G}|}{|\mathcal{F}|}&=\frac{c_3\times\sum_{\forall S_{im}\subseteq S_1}\{c_1\times c_2\}}{\prod_{s\in\mathbb{S}}\binom{|J_s|}{|D_s|}}\\ &=\frac{c_3\times\sum_{\forall S_{im}\subseteq S_1}\{c_1\times c_2\}}{\prod_{s\in\mathbb{S}\backslash S_1}\binom{|J_s|}{|D_s|}\times\prod_{s\in S_1}\binom{|J_s|}{|D_s|}}\\ &=\frac{\prod_{s\in\mathbb{S}\backslash S_1}\binom{|J_s|}{|D_s|}\times\sum_{\forall S_{im}\subseteq S_1}\{\prod_{s\in S_{im}}\binom{|J_s|-1}{|D_s|-1}\times\prod_{s\in S_\perp}\binom{|J_s|-1}{|D_s|}\}}{\prod_{s\in\mathbb{S}\backslash S_1}\binom{|J_s|}{|D_s|}\times\prod_{s\in S_1}\binom{|J_s|}{|D_s|}}\\ &=\frac{\sum_{\forall S_{im}\subseteq S_1}\{\prod_{s\in S_{im}}\binom{|J_s|-1}{|D_s|-1}\times\prod_{s\in S_\perp}\binom{|J_s|-1}{|D_s|}\}}{\prod_{s\in S_1}\binom{|J_s|}{|D_s|}}\end{aligned}$$

This concludes the proof. ■

APPENDIX D
PROOF OF THEOREM 3

Proof: In the context of this special case, referring to Lemma 1 and (12), $\frac{|\mathcal{F}\backslash\mathcal{G}|}{|\mathcal{F}|}$ can be extracted as:

$$\begin{aligned}
\frac{|\mathcal{F}\backslash\mathcal{G}|}{|\mathcal{F}|} &= \frac{\sum_{k=1}^{|S_1|}\{\binom{|S_1|}{k}\binom{|J_s|-1}{1}^k \times \binom{|J_s|-1}{2}^{|S_1|-k}\}}{\binom{|J_s|}{2}^{|S_1|}} \\
&= \frac{[\binom{|J_s|-1}{1}+\binom{|J_s|-1}{2}]^{|S_1|} - \binom{|J_s|-1}{2}^{|S_1|}}{\binom{|J_s|}{2}^{|S_1|}} \\
&= \frac{[\frac{|J_s|(|J_s|-1)}{2}]^{|S_1|} - [\frac{(|J_s|-1)(|J_s|-2)}{2}]^{|S_1|}}{[\frac{|J_s|(|J_s|-1)}{2}]^{|S_1|}} \\
&= \frac{|J_s|^{|S_1|} - (|J_s|-2)^{|S_1|}}{|J_s|^{|S_1|}} \\
&= 1 - (\frac{|J_s|-2}{|J_s|})^{|S_1|} \\
&= 1 - (1-\frac{2}{|J_s|})^{|S_1|} \\
&\leq 1 - (1-\frac{2}{|J_s|})^{|\mathbb{S}|},
\end{aligned}$$

where $|J_s| \geq 4$, and is same for $\forall s \in \mathbb{S}$.

Now, we calculate the fluctuation of system cost:

$$\begin{aligned}
\|\mathbf{q}^*\mathbf{u}^T - \hat{\mathbf{q}}\mathbf{u}^T\| &= \|\sum_{g\in\mathcal{G}}(q_g^* - \hat{q}_g)\cdot u_g\| \\
&\leq u_{\max}\cdot\sum_{g\in\mathcal{G}}|q_g^* - \hat{q}_g| \\
&= u_{\max}\cdot 2d_{TV}(\mathbf{q}^*, \hat{\mathbf{q}}) \\
&\leq 2u_{\max}[1-(1-\frac{2}{|J_s|})^{|\mathbb{S}|}].
\end{aligned}$$

On the other hand, it is not hard to find that the performance perturbation should be no greater than the cost of the worst configuration in $\mathcal{G}$. Thus the upper bound will never expire $u_{\max}$. Finally, we have the final result shown in (13). This concludes the proof. ■