Bandwidth-optimal Failure Recovery Scheme for Robust Programmable Networks

Andrea Tomassilli, Giuseppe Di Lena, Frédéric Giroire, Issam Tahiri, Damien Saucez, Stéphane Pérennes, Thierry Turletti, Ruslan Sadykov, François Vanderbeck, Chidung Lac

Bandwidth-optimal Failure Recovery Scheme for Robust Programmable Networks

A. Tomassilli*, G. Di Lena*†, F. Giroire*, I. Tahiri‡, D. Saucez*,
S. Perennes*, T. Turletti*, R. Sadykov‡, F. Vanderbeck‡ and C. Lac†
*Université Côte d'Azur, CNRS, Inria Sophia Antipolis, France
‡Université de Bordeaux, Institut de Mathématiques, France
†Orange Labs, France

***Abstract*—With the emergence of Network Function Virtualization (NFV) and Software Defined Networking (SDN) efficient network algorithms considered too hard to be put in practice in the past now have a second chance to be considered again. In this context, we rethink the network dimensioning problem with protection against Shared Risk Link Group (SLRG) failures. In this paper, we consider a path-based protection scheme with a global rerouting strategy, in which, for each failure situation, there may be a new routing of all the demands. Our optimization task is to minimize the needed amount of bandwidth. After discussing the hardness of the problem, we develop a scalable mathematical model that we handle using the Column Generation technique. Through extensive simulations on real-world IP network topologies and on random generated instances, we show the effectiveness of our method. Finally, our implementation in OpenDaylight demonstrates the feasibility of the approach and its evaluation with Mininet shows that technical implementation choices may have a dramatic impact on the time needed to reestablish the flows after a failure takes place.**

I. Introduction

Network failures such as cable cuts, natural disasters, faulty interfaces, or human errors are the daily routines of a network [1].Faults in the IP and optical layer tend to be correlated between them [2]. Indeed, the failure of a component located in a common router, such as a linecard, or in the underlying optical infrastructure, such as a common fiber, may result in the consequential failure of multiple entities at the IP layer. To model this correlation, the concept of *Shared Risk Link Groups* (SRLGs) has been proposed [3]. SRLGs allow to easily express a risk relationship, and also, they can represent different types of failures, such as single and multiple, nodes and links failures.

We consider in this paper a protection technique called *unrestricted flow reconfiguration*, also known as *global rerouting* [4]. In each of the possible failure situations, a new set of backup paths is defined, one for each demand. This makes this protection method *bandwidth-optimal*. However, this also means that each failure may result in a completely different routing for the demands. In legacy networks, it is impractical to implement this technique due to the large number of rules to install on the network devices and hence signaling overhead. However, the introduction of SDN may change the game.

This work has been supported by the French government through the UCA JEDI (ANR-15-IDEX-01) and EUR DS4H (ANR-17-EURE-004) Investments in the Future projects, and by Inria associated team EfDyNet.

With SDN the network control is decoupled from the packet forwarding data plane. Network intelligence is centralized in the controller that maintains a global view of the network [5]. Routing decisions are taken in a single location, *the controller*, with a complete knowledge of the network state instead of resulting from a distributed algorithm. As a result, with SDN, the global rerouting protection scheme may be put in practice.

We address in this paper the problem of designing an SDN/NFV-enabled network that provides SRLG-failure survivability under the global rerouting protection scheme.

Our goal is to compute for each demand, *a primary and a backup path for each SRLG failure scenario*, while ensuring that the required network functions will be performed on the packets in the order specified by its Service Function Chain. The studied problem is a *dimensioning problem* for a network that needs to determine the minimal amount of resources (e.g., link capacity) it needs to deploy, while guaranteeing protection against SRLG failures. Even though, at first glance, the problem may appear easy due to the absence of link capacity constraints, we demonstrate that it is not the case. Indeed, we show that *even for a single demand* the problem is NP-Hard and inapproximable within $(1-\epsilon)\ln(|R|)$ for any $\epsilon > 0$ unless P=NP, where $|R|$ denotes the number of SRLG failure scenarios.

Our contributions can be summarized as follows.

– To the best of our knowledge, we are the first to provide a scalable exact method to solve the problem of global rerouting in SDN/NFV-enabled networks with SRLG constraints.

– We also propose a fast 2-phase polynomial method. The first phase consists in solving the fractional relaxation of the problem. The second one is building an integral solution from the fractional one. It leads to an optimization problem we named Min Overflow Problem. We show that the problem is NP-complete, but that there exists a $(1+\frac{1}{e}+\varepsilon)$–approximation algorithm to solve it.

– We demonstrate the applicability of our proposed protection method in Mininet and study metrics such as the burden on the network elements and time to recovery from a failure.

The rest of this paper is organized as follows. In Section II, we discuss related work. In Section III, we formally define the problem to be studied, as well as notations that will be used in this paper. Section IV develops the proposed optimization approaches. In Section V, we validate our models

by various numerical results on real world and randomly generated data instances, and in Section VI, we use Mininet to demonstrate the feasibility of our proposal. Finally, we draw our conclusions in Section VII.

II. Related Work

The problem of providing network protection against failures has been widely investigated in the last decades, see, e.g., [6]. With the advent of SDN/NFV, there are more opportunities to create, deploy, and manage networks more efficiently. Indeed, with SDN and its control–data planes decoupling, routing decisions can be done using a logically centralized approach. This paves the way for a broadening of perspective in terms of fault management [7].

Chu et al. [8] consider a hybrid SDN network and propose a method to design the network in such a way that fast failure recovery from any single link failure is achieved. Their proposal consists in redirecting the traffic on the failed link from the routers to SDN switches through pre-configured IP tunnels. Next hops are pre-configured before the failures take place, and the set of candidate recovery paths for different affected destinations is chosen by the SDN controller in such a way that the maximal link utilization after redirecting the recovery traffic through these paths is minimized.

Suchara et al. [9] propose a joint architecture for both failure recovery and traffic engineering. Their architecture uses multiple pre-configured paths between each pair of edge routers. In the event of a failure, the failover is made on the least congested path that ensures connectivity. Besides, Sgambelluri et al. [10] propose a controller–based fault recovery solution that uses OpenFlow's Fast Failover Group Tables to quickly select a pre-configured backup path in case of link failure.

Different from previous studies on failure recovery, we present a simple and bandwidth-optimal approach based on multiple backup paths to protect the network against SRLG failures where SDN switches are deployed. Our concept was previously introduced in [11].

The idea of using a set of pre-configured multiple backup network configurations is not new. For instance, in [12], [13] the authors propose a pre-configured proactive IP recovery scheme that makes use of multiple routing backup configurations as a method for fast recovery. The main idea is to create a small set of backup routing configurations to be used in the case of a single link or node failure.
Herein, we take to the extreme the idea of multiple routing configurations by allowing a completely different routing in response to an SRLG failure situation. Different from the above works, our aim is to provide a bandwidth-optimal mechanism to design a reliable network. Besides guaranteeing the recovery, our proposed approach also takes into consideration the Service Function Chain (SFC) requirement of the flows.

III. Problem Statement and Notations

We model the network as an undirected graph $G = (V, E)$, where V represents the set of nodes and E the set of links. We are given a set of SRLG events $\mathcal{R}$ that can incur link failures.

Each $r \in \mathcal{R}$ consists of a set of links that share a common physical resource. We denote by $\mathcal{D}$ the set of demands (e.g., traffic between two locations). A demand $d \in \mathcal{D}$ is modeled by a quadruple (s_d, t_d, bw_d, C_d) with s_d the source, t_d the destination, C_d the ordered sequence of network functions that need to be performed to all the packets belonging to the flow of the demand, and bw_d the required units of bandwidth. We denote by $\ell(d)$ the length of the SFC for a demand d.

Network functions need to be executed on the so called NFVI nodes. Not all the nodes are enabled to run virtual functions. We denote by $V^{\text{VNF}} \subseteq V$ the set of VNF-enabled nodes. Moreover, we assume that an NFVI-enabled node can only run a subset of the network functions, as there may be constraints on their location in the network (e.g., geography or regulatory constraints and anti-affinity rules).

Given the network topology and the traffic rate of the demands to be supported, the purpose of the design problem is to precompute a set of paths to guarantee the recovery of all the demands in the event of an SRLG failure, while satisfying their SFC requirements. The considered optimization task is to *minimize the required bandwidth in the network*. We refer to this problem as the Global Rerouting problem.

IV. Optimization Approaches

We begin the section by proving hardness and inapproximability results for the Global Rerouting problem. Then, we propose a scalable decomposition model which relies on the Column Generation technique.

Proposition 1. *The* Global Rerouting *problem is NP-hard even for a single demand, and cannot be approximated within* $(1-\epsilon)\ln(|R|)$ *for any* $\epsilon > 0$ *unless P=NP, where* $|R|$ *denotes the number of failing scenarios.*

See [14] for the proof.

A. A Column Generation Approach

A straightforward way to model our problem consists in using an ILP. The goal of the ILP is to find for each demand $d \in \mathcal{D}$ a Service path on the layered graph $G^L(d)$ for each SRLG event such that the total bandwidth required in the network is minimized. One can apply the Dantzig-Wolfe decomposition to the ILP formulation, to exploit its block structure per demand $d \in \mathcal{D}$. The resulting model takes the form of a path flow formulation. In order to model the ordered sequence of network functions by which the traffic associated to a demand must be processed, we use a layered graph, similarly as in [15]. Let $G = (V, E)$ be a graph. We associate to each demand $d \in \mathcal{D}$ a layered graph $G^L(d) = (V', E')$. $G^L(d)$ is defined as follows. For each $u \in V$, V' contains the vertices $(u, 0), (u, 1), ..., (u, \ell(d))$. An edge $((u, i), (v, j))$ belongs to E' if and only if (1) $(u, v) \in E$ and $i = j$, or (2) u is a NFVI-enabled node, $u = v$, $j = i + 1$, and the j^{th} function of C_d is installed on u. We refer to a path in $G^L(d) = (V', E')$ as a Service Function Path (SFP).
We denote by Π_d^r, the set of service function paths for a demand d in the SRLG failure situation r. Each service path π

is associated with an integer value $a^{\pi}_{uv} \geq 0$ telling the number of times link (u, v) is used in the service path π.
Variables:
• $y^{d,r}_{\pi} \geq 0$, where $y^{d,r}_{\pi} = 1$ if demand d uses path π as a service path in the SRLG failure event $r \in \mathcal{R}$.
• $x_{uv} \geq 0$, is the bandwidth allocated on link $(u, v) \in E$.
Objective: minimization of the required bandwidth

$$\min \sum_{(u,v)\in E} x_{uv} \tag{1}$$

One service path for each demand and SRLG failure event: for all $d \in \mathcal{D}$, $r \in \mathcal{R}$

$$\sum_{\pi\in\Pi^r_d} y^{d,r}_{\pi} \geq 1. \tag{2}$$

Bandwidth utilization: for all $(u, v) \in E$, $r \in \mathcal{R}$

$$x_{uv} \geq \sum_{d\in\mathcal{D}} \sum_{\pi\in\Pi^r_d} bw_d \cdot a^{\pi}_{uv} \cdot y^{d,r}_{\pi}. \tag{3}$$

Given its very large number of variables, column generation is an efficient technique to handle the above linear integer programming model. One starts with a limited set of variables in a so-called restricted master program (RMP). At each iteration, the RMP is solved. The dual values associated to the constraints are used to generate new paths with negative reduced cost and the associated variables are added to the RMP that may enable to improve the current solution. This process is repeated until no more columns can be added to the RMP, i.e., no more columns with negative reduced cost exist. We refer to [16] for more details regarding this technique.

The pricing subproblem is solved independently for each demand d and SRLG failure event r and it returns a service path π. It consists in finding a minimum cost service path in the layered graph where the weight of a link is defined according to the dual values of the associated constraint.
Variables:
• $\varphi_{(ui,vj)} \in \{0, 1\}$, where $\varphi^{d,r}_{(ui,vj)} = 1$ if the flow is forwarded on link $((u, i), (v, j))$ of $G^L(d)$.
Let $\alpha^{sd}_{\omega} \geq 0$ and $\beta^{r}_{uv} \geq 0$ be the dual values relative to constraints (2) and (3), respectively. The service path reduced cost for a given demand d and an SRLG r can be written as:

$$\min -\alpha^{d}_{r} + bw_d \cdot \sum_{(u,v)\in E} \beta^{r}_{uv} \cdot \sum_{k=0}^{\ell(d)} \varphi_{(uk,vk)} \tag{4}$$

The first term is a constant for each request, and the second term corresponds to a summation over the links of the network. Therefore, the objective function becomes:

$$\min \sum_{(u,v)\in E} \beta^{r}_{uv} \cdot \sum_{k=0}^{\ell(d)} \varphi_{(uk,vk)}. \tag{5}$$

Thus, for each request and for each failure situation, the pricing subproblem corresponds to a weighted shortest-path problem in the layered graph. In a given SRLG failure situation r and for all the demands $d \in D$, the weight of a link $((u, i), (v, j))$ of $G^L(d)$ is defined to be β^{r}_{uv} if $i = j$, 0 otherwise. Either one of these paths leads to a negative reduced cost column, or the current master solution is optimal for the unrestricted program. In the former case, the new configurations found are then added iteratively to the RMP. In the second case, the solution of the linear relaxation of the RMP z^*_{LP} is optimal. Convergence of the basic column generation procedure suffers from dual oscillations as the number of constraints (3) is large. To improve the convergence and reduce the fluctuations in the dual variables, we use a piecewise linear penalty function stabilization described in [17]. Associated to the optimal solution of the linear relaxation of the RMP, for each demand d and SRLG failure situation r, there is a set of service paths identified by all the variables $y^{d,r}_{\pi}$ with value greater than 0. These service paths guarantee the minimum cost in terms of required bandwidth to deploy to guarantee the recovery in the splittable flow case. However, if we restrict our attention to the unsplittable flow case, we have to select only one service path for each demand and SRLG failure situation. The problem now consists in making this choice by reducing the *overflow* introduced in the network. One possible way consists in changing the domain of the variables in the last RMP from continuous to integer and use an ILP solver. We refer to this strategy as MASTERILP.

B. The Min-Overflow problem

As it is costly to solve (exactly) the integer version of the master program, to obtain a "good" integer solution, we could use another approach. That is, we may start by efficiently computing a fractional solution to the linear relaxation of the problem (i.e., when flows are splittable) using the Column Generation algorithm and then we try to obtain a *good* integer solution to the problem (i.e., when flows are unsplittable) by minimizing the cost to pay in terms of additional capacity (i.e., the *overflow*) over all the scenarios.

We define overflow as the total amount of additional bandwidth to be allocated in the network in order to satisfy all the demands. One possible strategy to do that may consist in considering each scenario one at a time, and formulating a multicommodity flow problem as an ILP. The objective function consists in minimizing the overflow to be allocated in the network. We refer to this strategy as ITERILP.
If on one hand, this strategy leads to good results, on the other hand, it may not scale well, since we have to solve an ILP for each SRLG failure scenario.

Another strategy consists in using an algorithm to route the demands while minimizing the overflow. The problem to be solved for an SRLG failure scenario which we refer to as MIN OVERFLOW PROBLEM can be stated as follows.
Input: A graph $G = (V, E)$, a collection $\mathcal{D}$ of demands, each associated with a source, a destination and the units of flows to be routed. Also, each demand is associated with a set of paths, corresponding to the fractional solution of the splittable flow version of the problem. Lastly, a capacity function $c^* : (u, v) \rightarrow c^*_{uv}$, according to the optimal capacities found solving the linear relaxation of the general problem.
Output: A path for each demand.
Objective: Minimize the overflow, i.e., minimize $\sum_{(u,v)\in E} \frac{\tilde{c}(u,v)}{c^*_{uv}}$ with $\tilde{c}(u, v)$ defined as the maximum

between c^*_{uv} and the capacity of the link (u, v) after having selected one path per demand.

Note that, contrary to the classical version of the problem, we do not have hard capacity constraints to respect while computing an integer routing. Herein, the goal is to route all the demands reducing the increase in terms of capacity over each of the links (i.e., the overflow) with respect to the *free given capacities* already available in the network.

Proposition 2. *The* MIN OVERFLOW PROBLEM *is APX-hard (and so is NP-Hard) and cannot be approximated within a factor of* $1 + \frac{3}{320}$*, unless P=NP.*

See [14] for the proof.

Proposition 3. *The* MIN OVERFLOW PROBLEM *can be approximated with high probability within a factor of* $(1 + \frac{1}{e}) + \epsilon$*, for any* $\epsilon > 0$.

Let c^*_{uv} be the optimal capacity of an edge (u, v) in the splittable flow case. After having computed a fractional flow, we have associated to each demand $d \in \mathcal{D}$ a set consisting of $n(d) \geq 1$ paths $\mathcal{P}_d = \{P_{d,i} : i = 1, ..., n(d)\}$. Each path $P_{d,i}$ is associated to a multiplier $0 \leq \lambda_{d,i} \leq 1$ such that $\sum_{i=1}^{n(d)} \lambda_{d,i} = 1$ which gives the amount of flow $\lambda_{d,i} \cdot bw_d$ routed on $P_{d,i}$. In order to find an unsplittable solution, we use a rounding–based heuristic referred to as RANDOMIZED ROUNDING, which assigns to a demand d a path $P_{d,i}$ with probability $\lambda_{d,i}$. The expected cost of the solution provided is $1.37 + \epsilon$ times the optimal one. See [14] for a detailed proof. We may extend RANDOMIZED ROUNDING to the case of multiple scenarios by simply solving the scenarios in an iterative fashion. At each iteration, an SRLG $r \in \mathcal{R}$ is considered. First, a fractional capacitated multicommodity flow is solved. Then, a $(1 + \frac{1}{e} + \epsilon)$–approximated integer solution is found using the RANDOMIZEDROUNDING procedure. The overflow introduced (if any) by the procedure is then added. We refer to this method as ITERATIVE RANDOMIZED ROUNDING.

V. NUMERICAL RESULTS

In this section, we evaluate the performances of our proposed algorithms. The compared methods are MasterILP, in which the last RMP is solved as an ILP by setting the domain of the paths variables from fractional to binary. IterILP, in which each scenario is solved independently with an ILP that has, as a goal, the minimization of the overflow and IterRR, in which instead of using an ILP to minimize the overflow, we use a $(1 + \frac{1}{e} + \epsilon)$–approximation algorithm. We show the effectiveness of our algorithms in terms of *scalability* and of GLOBAL REROUTING in terms of *bandwidth usage*.

Data sets. We conduct experiments on three real-world topologies from SNDlib [18]: `polska`, (12 nodes, 18 links, and 66 demands), `pdh` (11 nodes, 34 links, and 24 demands) and `nobel-germany` (17 nodes, 26 links, and 121 demands). For these networks, we use the given traffic matrices. No information is available about the SRLGs for these networks. Thus, the collection of network failures $\mathcal{R}$ for these instances contains single edge failures. We also conduct experiments

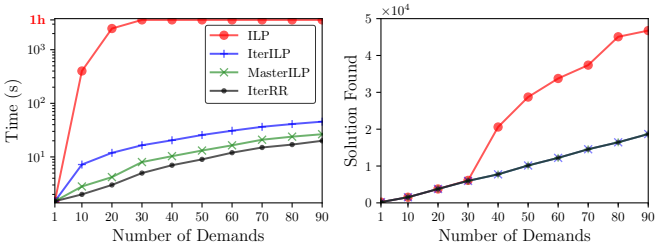

Fig. 1: Time and value of the solution found by the ILP and by our proposed methods as a function of the number of demands.

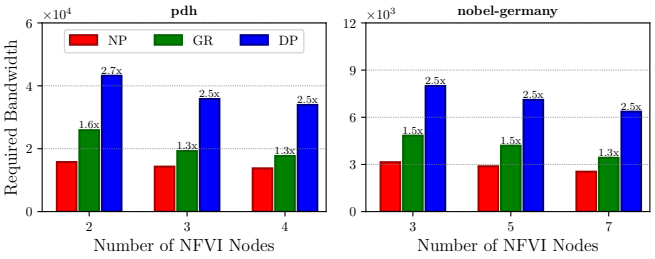

Fig. 2: Bandwidth overhead comparison of the Global Rerouting (GR) and Dedicated Path Protection (DP) schemes with respect to the No-Protection scenario (NP) for `pdh` and `nobel-germany` networks. Labels on top of the bars indicate the overhead with respect to the unprotected case.

on randomly generated instances of different sizes and with different SRLGs. Due to space constraints, random instances are presented and studied in [14], but the obtained results are similar. All the IP links using the same physical link are associated to an SRLG. In addition, we add an SRLG for each undirected link. Demands are generated using the model described in [19]. The model considers the distance factor $\exp^{\frac{-\text{dist}(u,v)}{2L}}$ between two nodes u and v, where L is the maximum distance between two nodes. As a result, the load of the demands between close pairs of nodes is higher with respect to pairs of nodes far apart. Finally, the chain of each demand is composed of 3 to 6 functions uniformly chosen at random from a set of 10 functions. Each VNF-enabled node can run up to 6 network functions. Indeed, a node may not be allowed to run all the network functions. Similarly as in [15], locations are chosen according to their betweenness centrality, an index of the importance of a node in the network: it is the fraction of all shortest paths between any two nodes that pass through a given node. Experiments have been conducted on an Intel Xeon E5520 with 24GB of RAM.

Limits of an ILP-based approach. To study the limits in terms of computing time of an ILP-based approach (the detailed model can be found in [14]), we tested our optimization models on a small random topology with 10 nodes, 16 links, and 26 SRLGs. In Fig. 1, we show the impact of the number of demands on the execution time. We compare the time necessary to find an optimal solution (on the left) and the value of the solution found (on the right) by the ILP and by our proposed methods. For each experiment, we set a maximum time limit of one hour. If the time limit is exceeded, the

solution reported represents the best solution found so far. For just 30 demands, the time needed by IBM ILOG Cplex 12.8 to find an exact solution exceeds 1 hour. For large instances, an optimal solution cannot be found using an ILP approach in a reasonable amount of time. On the other hand, the proposed algorithms can compute solutions for larger instances fairly efficiently. Indeed, these algorithms only take 1 minute to solve the problem for 90 demands. As the considered network is small, the computed values tend to be close between them. We compare our optimization models in more detail in [14].

Varying the number of NFVI-enabled nodes. In Fig. 2, we compare the overhead in terms of bandwidth needed in the network by the global rerouting scheme and Dedicated Path Protection with respect to the bandwidth needed in the unprotected case. For Dedicated Path Protection we compute, for each demand, two SRLG-disjoint paths, i.e., two paths such that no link on one path has a common risk with any link on the other path. In doing this, we set the bandwidth minimization as an optimization task. With an increasing number of NFVI nodes in the network, the required bandwidth decreases. However, the overhead with respect to the unprotected case tends to remain constant. Indeed, if with global rerouting we only need from 30 to 60% more bandwidth, with dedicated path protection we may need almost 3 times more bandwidth to guarantee the recovery. In [14], we additionally study the impact of the number of NFVI nodes on the paths' latencies distribution and compare them with the ones calculated using shortest paths on the layered network. We show that the length of the paths computed using our method are almost as good (in terms of number of hops) as the shortest paths.

VI. Experimental evaluation

In this section, we discuss how to implement our proposition with OpenFlow and we evaluate it with Mininet. Our evaluation shows that implementation choices have a significant impact on the recovery time of protection mechanisms.

A. Implementation options

A first option to implement the protection scheme in OpenFlow is to let the OpenFlow controller fully update the flow tables on the switches upon failure. When the controller detects a failure, it sends the new flow tables to the impacted switches. This approach minimizes the memory usage on the switches but incurs high signaling overhead between the controller and the switches, and imposes the latter to install a full flow table at every network change. We refer to this option as *full*. A variation of this option is to only send the changes to be performed on the flow tables to the switches to reduce the signaling load and the number of flow table updates on the switches. We name this option *delta*. Another option is to leverage the Multiple Flow Tables capability introduced in OpenFlow 1.3 to pre-install the flow tables for each SRLG failure scenario in the switches. When the controller sends a failure notification to a switch, the switch activates the appropriate flow table in only one operation (using goto). This approach minimizes the signaling load and flow table changes but consumes more memory on the switches than the other options. This option is referred to as *notification*.In the rest of the paper, we study the impact of the technical choices on the recovery time in realistic operational scenarios.

B. Experimental setup

Our experimental platform is a dual Intel Xeon E5-2630 CPU server with 128GB of RAM running Mininet 2.2.2 and the controller *OpenDaylight* Oxygen with OpenFlow 1.3.
The routing logic is implemented as a network application orchestrator that communicates with the controller with the HTTP OpenDaylight Northbound API. This approach is recommended as it decouples the implementation of the logic from the implementation of the controller.

We also made an *ideal* implementation to assess the best possible performance one could have. It is equivalent to the *notification* option but is implemented directly in Mininet with Open vSwitch commands. Mininet emulation is centralized, so we are able to synchronize all failure notifications to the switches just after the failure occurs bypassing thus the controller.

Due to the limited number of CPU cores on our emulation server, we could only evaluate the `wxm10` and the `polska` networks. Due to space limitations, we only discuss the `polska` network in this paper. Details about `wxm10`, that behaves the same as `polska`, can be found in [14].

C. Recovery time

The *recovery time* is the span of time between a failure event and the moment in which all switches are updated to be in a state that circumvents the failure. To measure the recovery time, we continuously probe the end-to-end paths with UDP datagrams. Fig. 3 shows the recovery time for our three OpenDaylight implementation options and the ideal one. It compares our Global Rerouting (GR) protection scheme to the Dedicated Path Protection (DP) scheme. The figure highlights the importance of implementation choices on the recovery time: the notification option significantly outperforms the other options. The ideal implementation also shows that the tools used to implement the protection scheme have a significant impact on the recovery time as, all things considered, our ideal is just a way of implementing the notification option without a controller. Actually, a significant fraction of the recovery time in OpenDaylight implementations is caused by the usage of the Northbound API. All implementation options offer sub-second recovery time for the considered network. Figures 3 and 4 show that there is a direct link between the number of changes to be performed on the switches and the recovery time. Fig. 4 reports, for each switch, the maximum number of flow table changes observed expressed in number of flow entries for the three OpenDaylight implementation options. Dedicated path protection has longer recovery time than global rerouting when the full implementation is used. This is because with DP two SRLG-disjoints paths are always provided while GR only provides the paths of the current scenario. On the contrary, DP converges faster than GR with

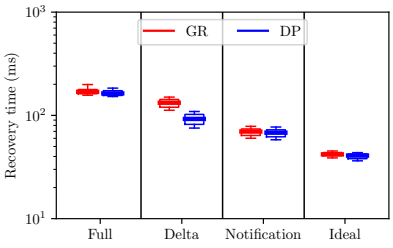

Fig. 3: Recovery time comparison of various implementation options for Global Rerouting (GR) and Dedicated Path Protection (DP) for the `polska` network.

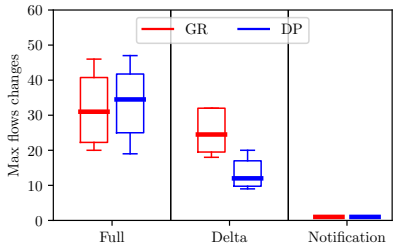

Fig. 4: Comparison of the number of flow table changes of various implementation options for Global Rerouting (GR) and Dedicated Path Protection (DP) for the `polska` network.

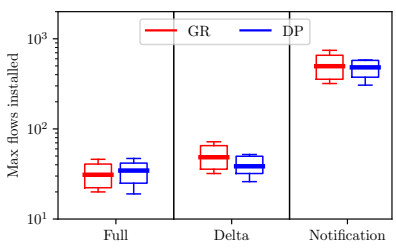

Fig. 5: Comparison of the flow table sizes of various implementation options for Global Rerouting (GR) and Dedicated Path Protection (DP) for the `polska` network.

the delta implementation as less path changes are needed for DP than for GR. When notifications are used, GR and DP reach the same performance.

D. Operational trade-offs

Based on the recovery time, one would recommend to deploy the notification option. However, the reduction of the recovery time comes at the cost of increasing flow table sizes on switches. Fig. 5 reports, for each switch, the maximum observed flow table size expressed in number of flow entries for the three OpenDaylight implementation options. The full option minimizes the number of entries as it only requires to have the flow table for the current routing case. The delta option consumes slightly more space than the full one as the flow table always contains the "no-failure" scenario flow table and the additional flow entries needed to circumvent the current failure. Finally, the notification option has significantly larger flow tables (one order of magnitude more) as flow tables always contain all the potential failure scenarios.

As the robustness of the controller is an orthogonal problem that must be treated by all SDN solutions and because it is already largely studied [20], it was not considered here.

VII. Conclusion

In this paper, we studied the network dimensioning problem with protection against a Shared Risk Link Group failure in the light of network virtualisation. We considered a path-protection method based on a global rerouting strategy, which makes the protection method optimal in terms of bandwidth. We proposed algorithms to compute the backup paths for the demands which rely on the Column Generation technique. We validated them with simulations on real-world instances. Finally, we showed the applicability of the global rerouting protection method thanks to SDN with a real implementation using OpenDaylight.

References

[1] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot, "Characterization of failures in an ip backbone," in *Proceedings of IEEE INFOCOM, 2004*.

[2] S. Kandula, D. Katabi, and J.-P. Vasseur, "Shrink: A tool for failure diagnosis in ip networks," in *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*. ACM, 2005, pp. 173–178.

[3] D. Papadimitriou, "Inference of shared risk link groups," *Internet-draft: draft-many-inference-srlg-02. txt*, 2001.

[4] M. Pióro and D. Medhi, *Routing, flow, and capacity design in communication and computer networks*. Elsevier, 2004.

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

[6] A. Fumagalli and L. Valcarenghi, "Ip restoration vs. wdm protection: Is there an optimal choice?" *IEEE network*, vol. 14, no. 6, 2000.

[7] P. Fonseca and E. Mota, "A survey on fault management in software-defined networks," *IEEE Communications Surveys & Tutorials*, 2017.

[8] C.-Y. Chu, K. Xi, M. Luo, and H. J. Chao, "Congestion-aware single link failure recovery in hybrid sdn networks," in *Proceedings of IEEE INFOCOM, 2015*.

[9] M. Suchara, D. Xu, R. Doverspike, D. Johnson, and J. Rexford, "Network architecture for joint failure recovery and traffic engineering," in *Proceedings of ACM SIGMETRICS 2011*. ACM.

[10] 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, 2013.

[11] A. Tomassilli, G. Di Lena, F. Giroire, I. Tahiri, D. Saucez, S. Pérennes, T. Turletti, R. Sadykov, F. Vanderbeck, and C. Lac, "Poster: Design of Survivable SDN/NFV-enabled Networks with Bandwidth-optimal Failure Recovery," *Proc. of IFIP Networking 2019*.

[12] A. Kvalbein, A. F. Hansen, S. Gjessing, and O. Lysne, "Fast ip network recovery using multiple routing configurations," in *Proceedings of IEEE INFOCOM, 2006*.

[13] A. Kvalbein, T. Cicic, and S. Gjessing, "Post-failure routing performance with multiple routing configurations," in *Proceedings of IEEE INFOCOM, 2007*.

[14] A. Tomassilli, G. Di Lena, F. Giroire, I. Tahiri, D. Saucez, S. Pérennes, T. Turletti, R. Sadykov, F. Vanderbeck, and C. Lac, "Bandwidth-optimal Failure Recovery Scheme for Robust Programmable Networks," Research Report, Apr. 2019. [Online]. Available: https://hal.inria.fr/hal-02112282/

[15] N. Huin, B. Jaumard, and F. Giroire, "Optimal network service chain provisioning," *IEEE/ACM Transactions on Networking*, 2018.

[16] G. Desaulniers, J. Desrosiers, and M. M. Solomon, *Column generation*. Springer Science & Business Media, 2006, vol. 5.

[17] A. Pessoa, R. Sadykov, E. Uchoa, and F. Vanderbeck, "Automation and combination of linear-programming based stabilization techniques in column generation," *INFORMS Journal on Computing*, 2018.

[18] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "Sndlib 1.0—survivable network design library," *Networks*, vol. 55, no. 3, 2010.

[19] B. Fortz and M. Thorup, "Optimizing ospf/is-is weights in a changing world," *IEEE journal on selected areas in communications*, vol. 20, no. 4, pp. 756–767, 2002.

[20] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, "A survey on software defined networking with multiple controllers," *J. Netw. Comput. Appl.*, vol. 103, no. C, pp. 101–118, Feb. 2018.