No Interruption When Reconfiguring my SFCs

Adrien Gausseran, Andrea Tomassilli, Frédéric Giroire, Joanna Moulierac

No Interruption When Reconfiguring my SFCs

Adrien Gausseran, Andrea Tomassilli, Frédéric Giroire, Joanna Moulierac
Université Côte d'Azur, CNRS, Inria Sophia Antipolis, France

***Abstract*—Software Defined Networking (SDN) and Network Function Virtualization (NFV) are complementary and core components of modernized networks. In this paper, we consider the problem of reconfiguring Service Function Chains (SFC) with the goal of bringing the network from a sub-optimal to an optimal operational state. We propose optimization models based on the *make-before-break* mechanism, in which a new path is set up before the old one is torn down. Our method takes into consideration the chaining requirements of the flows and scales well with the number of nodes in the network. We show that, with our approach, the network operational cost defined in terms of both bandwidth and installed network function costs can be reduced and a higher acceptance rate can be achieved, while not interrupting the flows.**

I. Introduction

The last decade has seen the development of new paradigms to pave the way for a more flexible, open, and economical networking. In this context, Software Defined Networking (SDN) and Network Function Virtualization (NFV) are two of the most promising technologies for the Next-Generation Network. SDN aims at simplifying network management by decoupling the control plane from the data plane. Network intelligence is logically centralized in an SDN controller that maintains a global view of the network state. As a consequence, the network becomes programmable and can be coupled to users' business applications [1]. With the NFV paradigm, network functions (e.g., a firewall, a load balancer, and a content filtering) can be implemented in software and executed on generic-purpose servers located in small cloud nodes. Virtual Network Functions (VNFs) can be instantiated and scaled on–demand without the need of installing new equipment. Besides, network flows are often required to be processed by an ordered sequence of network functions. For example, an Intrusion Detection System may need to inspect the packet before compression or encryption are performed. This notion is known as Service Function Chaining (SFC) [2]. SDN has the potential to make the chaining of the network functions much easier.

In this context, a fundamental problem that arises is how to map these VNFs to nodes (servers) in the network to satisfy all demands, while routing them through the right sequence of functions and meeting service level agreements. In doing this, the capacity constraints on both nodes and links must be respected. Our goal is to minimize the network operational cost, defined in terms of both bandwidth cost to route the demands and the cost for all the VNFs running in the network. Moreover, the network state changes continually due to the arrival and departure of flows. An optimal or near-optimal resource allocation may result after a lapse of time in over-provisioning or in an inefficient resource usage. Also, it may lead to a higher blocking probability even though there are enough resources to serve new demands. Indeed, as reported by [3], 99% of rejections were caused by bandwidth shortage even though there were enough resources to satisfy the request. Therefore, operators must take it into consideration and adjust network configurations in response to changing network conditions to fully exploit the benefits of the SDN and NFV paradigms, and to avoid undue extra cost (e.g., software licenses, energy consumption, and Service Level Agreement (SLA) violation). Thus, another problem is how to reroute traffic flows through the network and how to improve the mapping of network functions to nodes in the presence of dynamic traffic, with the goal being to bring the network closer to an optimal operating state, in terms of resource usage.
Rerouting demands and migrating VNFs may take several time steps. If during this time, traffic is interrupted, it may have a non-negligible impact on the QoS experienced by the users. To tackle this issue, our strategy performs the reconfiguration by using a two–phase approach. First, a new route for the transmission is established while keeping the initial one enabled (i.e., two redundant data streams are both active in parallel), and after the network has been updated to the new state, the transmission moves on the new route and the resources used by the initial one are released. This strategy is often referred to as *make-before-break*. Our contributions can be summarized as follows.

- We provide the first method, `Break-Free`, to reconfigure, with a *make-before-break mechanism*, the routing and provisioning of a set of service function chains. This mechanism allows to reach closer to optimal resource allocation while *not interrupting the demands* which have to be rerouted.
- We show that `Break-Free` allows to *lower the network cost* and *increase the acceptance rate*. It can achieve, in most considered cases, a gain close to the one of a reconfiguration algorithm that interrupts the requests (referred to as `Breaking-Bad` in the following).

We additionally exhibit that the percentage of demands which have to be rerouted to achieve a significant gain in terms of network cost or acceptance rate is very high. This shows the importance of considering mechanisms limiting the impact on the demands. Network reconfiguration has to be done frequently to achieve a significant gain, however this reconfiguration can be quickly computed and carried out, making it possible to be put into practice in real time.

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.

II. Related Work

The problem of how to deploy and manage network services conceived as a chain of VNFs has received a significant interest in the research and industrial community. We refer to [4] and [5] for comprehensive surveys on the relevant state of the art. Although a lot of effort has been made to develop efficient strategies to route demands and satisfy their chaining requirements, not enough has been made to improve resources usage during network operation. Recently, some research work has started to explore SDN capabilities for a more efficient usage of the network resources by dynamically adapting the routing configuration over time. For instance, Paris *et al.* [6] study the problem of online SDN controllers to decide when to perform flow reconfigurations for efficient network updating such that the flow reallocation cost is minimized. However, the network function requirements are not considered in their work. Indeed, the traffic of a request may need to be steered to traverse middleboxes implementing the required network functions. Ayoubi *et al.* [7] propose an availability-aware resource allocation and reconfiguration framework for elastic services in failure-prone data center networks. Their work is limited to the case of Virtual Network scale-up requests such as resource demand increase, new network component arrival, and/or service class upgrade. The goal is to provide the highest availability improvement minimizing the overall reconfiguration cost.

Ghaznavi *et al.* [8] propose a consolidation algorithm that optimizes the placement of the VNFs in response to on-demand workload. The algorithm decides the VNF Instances to be migrated on the basis of the reconfiguration costs implied by the migration. However, they assume only one type of VNF and do not consider chaining requirements.

In [9], Eramo *et al.* study the problem of migrating VNFs in a dynamic scenario. The considered objective is to minimize the network operation cost which is the sum of the energy consumption costs and the revenue loss due to the bit loss occurring during the downtime. However, their model does not consider the bandwidth resources.

The closest study to our work is from Liu *et al.* [10]. They consider the problem of optimizing VNF deployment and readjustment to efficiently orchestrate dynamic demands. When a new request arrives, the service provider can serve it or change the provisioning schemes of the already deployed ones at time instances with a fixed interval in between. They consider the maximization of the service provider's profit which is the total profit from the served requests minus the total deployment cost as an optimization task. For this purpose, they formulate an Integer Linear Programming (ILP) model. Then, to reduce the time complexity, they design a column generation model. An important unaddressed issue concerns the revenue loss of an operator due to the QoS degradation occurring when demands are reconfigured [9]. Indeed, in their model, transmissions may need to be interrupted in order to be moved to the new computed state.

Different from the above mentioned works, our aim is to provide efficient mechanisms to dynamically reallocate the demands *without the consequential QoS deterioration due to the traffic interruption, but instead using make-before-break strategy*.

III. Problem Statement and Notations

We model the network as a directed capacitated graph and the set of demands D as a set of quadruples as shown in Table I, which defines the notation used throughout the paper.

We consider a setting with splittable flows as it is frequent to have load balancing in networks [11] and as it makes the model quicker to solve [12]. Following the model of [13], a demand can follow different paths and the network functions of its chain can be processed in different cloud nodes.

The optimization task consists in routing each demand while *minimizing the network operational cost* defined in terms of bandwidth and VNF cost (licenses, energy consumption, etc). Also, as the dynamics related to the arrival and departure of demands may leave the network in a sub-optimal operational state, we want to reconfigure the network to improve resource usage and to be able to accommodate new incoming traffic. In doing this, we use the *make-before-break* mechanism to avoid network service disruption due to traffic rerouting resulting from the re-optimization process.

An Example. Figure 1 illustrates an example for the reconfiguration of a request using a *make-before-break* process. When the request from A to F arrives, two requests have already been routed (step (b)). To avoid the cost of installing new VNFs, the route from A to F with minimum cost is a long 5–hops route (step (c)). When requests from B to C and from F to C leave, the request is routed on a non-optimal path (step (d)) which uses more resources than necessary. We compute one optimal 3-hops path and reroute the request to it (step (f)) with an intermediate make-before-break step (step (e)) in which both routes co-exist.

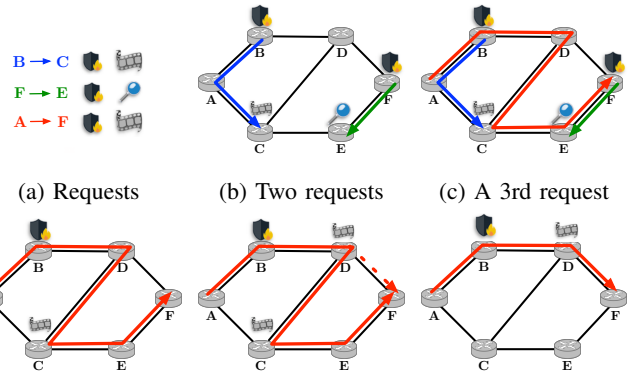
IV. Modeling

In the considered setting, demands arrive and leave the network. To route them, we consider them one by one, and find the route which minimizes the *additional network operational cost* to be paid. Indeed, in an SDN network, even if multiple

[1]A node u with a strictly positive number of cores (i.e., $C_u \in \mathbb{N}^+ = \{1, 2, \cdots\}$) represents a cloud location with the capability to execute VNFs, while a node with $C_u = 0$ is a node that serves only as an SDN router.

$G = (V, E)$	the network where V represents the set of nodes and E the set of links.
C_{uv}	capacity of a link $(u, v) \in E$ expressed as its total bandwidth available.
C_u	available resources[1] such as CPU, memory, and disk of a node $u \in V$.
Δ_f	number of units of bandwidth required by the function $f \in F$.
$c_{u,f}$	installation cost of the function $f \in F$ which also depends on the node u.
(v_s, v_d, c_d, bw_d)	each demand $d \in D$ is modeled by a quadruple with v_s the source, v_d the destination, c_d the ordered sequence of network functions that need to be performed, and bw_d the required units of bandwidth.

TABLE I: Notation used throughout the paper



(a) Requests (b) Two requests (c) A 3rd request

(d) The first two requests leave (e) Reconfiguration phase (f) Optimal routing for the third request

Fig. 1: An example of the reconfiguration of a request using the *make-before-break* procedure.

flows arrive simultaneously, they will be processed one by one by the SDN controller [14]. We then reconfigure the network to improve the network operational cost when one of the following conditions holds:

- Periodically, after a given period of time;
- When the set of requests has changed significantly (after a given number of SFC arrivals and departures);
- When a request arrives and cannot be accepted with the current provisioning and routing solution.

The solution we propose, called `Break-Free` (for Break-Free Reconfiguration algorithm), implements a *make-before-break* mechanism to avoid the interruption of the flows. In our experiments, we compare its results with `Breaking-Bad` (for Breaking-Bad Reconfiguration algorithm). This algorithm finds an optimal Routing & Provisioning solution (R&P) without considering the current setting. Then, it breaks the flows before rerouting them, implying packet losses and QoS degradation for these flows. Our models are based on the concept of a layered graph presented in Section IV-A.

A. Layered graph

Similarly as in [15], in order to model the chaining constraint of a demand, we associate to each demand d a layered graph $G^L(d)$. We denote by $u_{i,l}$ the copy of node u_i in layer l. The path for demand d starts from node $v_{s,0}$ in layer 0 and ends at node $v_{d,|c_d|}$ in layer $|c_d|$ where $|c_d|$ denotes the number of VNFs in the chain of the demand.
Given a link (u_i, v_j), each layer l has a link $(u_{i,l}, v_{j,l})$ defined. This property does not hold for links of the kind $(u_{i,l}, u_{i,l+1})$. Indeed, a node may be enabled to run only a subset of the virtual functions. To model this constraint, given a demand d we add a link $(u_{i,l}, u_{i,l+1})$ only if Node u is enabled to run the $(l+1)-th$ function of the chain of d. The $l-th$ function of the chain of d will be denoted by $f_l^{c_d}$.
A path on the layered graph corresponds to an assignment to a demand of both a path and the locations where functions are being run. Using a link $(u_{i,l}, v_{j,l})$ on G^L, implies using link (u, v) on G. On the other hand, using link $(u_{i,l}, u_{i,l+1})$ implies using the $(l+1)-th$ function of the chain at node u. Capacities of both nodes and links are shared among layers. See [16] for an example.

B. `Break-Free` *Reconfiguration (Make-before-break)*

The goal of the optimization is to find a better solution than the given one which can be reached using T reconfiguration steps. This can be done using an ILP computing the T transitions from the current routing. Each intermediate reconfiguration step corresponds to a solution of the R&P problem. The objective is to minimize the network operational cost (i.e., bandwidth cost and network function activation cost) of the R&P of the final state.

At time 0, the R&P is set to the current one. Then, at each step of reconfiguration, a set of demands can be rerouted as long as there are enough link and node capacities to satisfy the intermediate make-before-break reconfiguration steps. When the path of a demand has to be moved at a time step, the two paths (the new one at t_i and the new one at t_{i+1}) coexist during the intermediate step. This can be modeled linearly by defining a variable which is equal to 1 if a resource is used by a request either at time $t-1$ or at time t.

The value of T is an important parameter. Indeed, a value too small may lead to models with no solution, while a value too large to models with prohibitive execution times. This is why we tested different values in our experiments (between 1 and 4). We observe that when the network is not congested, corresponding to the low-traffic scenarios of Section V-B, a single reconfiguration step is enough to provide optimal (or close to optimal) solutions. This value of T leads to solutions almost as bad as without reconfiguration in the high-traffic scenarios of Section V-C. A good way to find the right value is to start with $T = 1$, which is the fastest model, and then to increase progressively the value of T until either the solution does not improve any more or the solving time is too long.
Model. The ILP takes as an input both the current configuration (i.e., paths and function locations for all the demands) and the number of time steps T to be used in the reconfiguration process. The output corresponds to both the final SFC-R&P at time T after the reconfiguration process and the intermediate SFC-R&P to be used to reach the final state. Between two consecutive time steps $t_0 < ... < t_i < t_{i+1} < ... < T$, a subset of the demands may be moved to a new route. In doing this, resources of both nodes and links must not be exceeded.
Variables:
- $\varphi_{uv,i}^{d,t} \geq 0$ is the amount of flow on Link (u, v) in Layer i at time step t for Demand d.
- $\alpha_{u,i}^{d,t} \geq 0$ is the fraction of flow of Demand d using Node u in Layer i at time step t.
- $x_{uv,i}^{d,t} \geq 0$ is the maximum amount of flow on Link (u, v) in Layer i at time steps t and $t-1$ for Demand d.
- $y_{u,i}^{d,t} \geq 0$ is the maximum fraction of flow of demand d using Node u in Layer i at time steps t or $t-1$.
- $z_{u,f}^{T} \in \{0, 1\}$, where $z_u^f = 1$ if function f is activated on Node u at time step T in the final routing.

The optimization model starts with the initial configuration as an input. Thus, for each demand $d \in D$ the variables $\varphi_{uv,i}^{d,0}$

(for each node $u \in V$, layer $i \in \{0, ..., |c_d|\}$) and $\alpha_{u,i}^{d,0}$ (for each link $(u, v) \in E$, layer $i \in \{0, ..., |c_d|\}$) are known. The ILP is based on the layered graph described in Section IV-A. Objective: minimize the amount of network resources consumed during the last reconfiguration time step T. The parameter $\beta \geq 0$, specified by the network administrator, can be defined as the ratio between the cost of sending 1 TB of traffic on a link and the cost of installing a function which can handle 1 TB of data.

$$\min \quad \sum_{d \in D} \sum_{(u,v) \in E} \sum_{i=0}^{|c_d|} bw_d \cdot \varphi_{uv,i}^{d,T} + \beta \cdot \sum_{u \in V} \sum_{f \in F} c_{u,f} \cdot z_{u,f}^{T}$$

Flow conservation constraints. For each Demand $d \in D$, Node $u \in V$, time step $t \in \{1, ..., T\}$.

$$\sum_{\substack{(u,v) \in \\ \omega^+(u)}} \varphi_{uv,0}^{d,t} - \sum_{\substack{(v,u) \in \\ \omega^-(u)}} \varphi_{vu,0}^{d,t} + \alpha_{u,0}^{d,t} = \begin{cases} 1 \text{ if } u = v_s \\ 0 \text{ else} \end{cases} \tag{1}$$

$$\sum_{\substack{(u,v) \in \\ \omega^+(v)}} \varphi_{uv,|c_d|}^{d,t} - \sum_{\substack{(v,u) \in \\ \omega^-(v)}} \varphi_{vu,|c_d|}^{d,t} - \alpha_{u,|c_d|-1}^{d,t} = \begin{cases} -1 \text{ if } v = v_d \\ 0 \text{ else} \end{cases} \tag{2}$$

$$\sum_{\substack{(u,v) \in \\ \omega^+(u)}} \varphi_{uv,i}^{d,t} - \sum_{\substack{(v,u) \in \\ \omega^-(u)}} \varphi_{vu,i}^{d,t} + \alpha_{u,i-1}^{d,t} - \alpha_{u,i-1}^{d,t} = 0 \quad 0 < i < |c_d| \tag{3}$$

Node usage over two consecutive time periods. For each Demand $d \in D$, Node $u \in V$, Layer $i \in \{0, ..., |c_d| - 1\}$ time step $t \in \{1, ..., T\}$.

$$\alpha_{u,i}^{d,t} \leq y_{u,i}^{d,t} \text{ and } \alpha_{u,i}^{d,t-1} \leq y_{u,i}^{d,t} \text{ and } \alpha_{u,i}^{d,t} + \alpha_{u,i}^{d,t-1} \geq y_{u,i}^{d,t} \tag{4}$$

Link usage over two consecutive time periods. For each Demand $d \in D$, Link $(u, v) \in E$, Layer $i \in \{0, ..., |c_d|\}$ time step $t \in \{1, ..., T\}$.

$$\varphi_{uv,i}^{d,t} \leq x_{uv,i}^{d,t} \text{ and } \varphi_{uv,i}^{d,t-1} \leq x_{uv,i}^{d,t} \text{ and } \varphi_{uv,i}^{d,t} + \varphi_{uv,i}^{d,t-1} \geq x_{uv,i}^{d,t} \tag{5}$$

Make Before Break - Node capacity constraints. The capacity of a node u in V is shared between each layer and cannot exceed C_u considering the resources used over two consecutive time periods. For each Node $u \in V$, time step $t \in \{1, ..., T\}$.

$$\sum_{d \in D} bw_d \sum_{i=0}^{|c_d|-1} \Delta_{f_i^{c_d}} \cdot y_{u,i}^{d,t} \leq C_u \tag{6}$$

Make Before Break - Link capacity constraints. The capacity of a link $(u, v) \in E$ is shared between each layer and cannot exceed C_{uv} considering the resources used over two consecutive time periods. For each Link $(u, v) \in E$, time step $t \in \{1, ..., T\}$.

$$\sum_{d \in D} bw_d \sum_{i=0}^{|c_d|} x_{uv,i}^{d,t} \leq C_{uv} \tag{7}$$

Location constraints. A node may be enabled to run only a subset of the virtual network functions. For each Demand $d \in D$, Node $u \in V$, layer $i \in \{0, ..., |c_d| - 1\}$, if the $(i+1) - th$ function of c_d cannot be installed on Node u, we add the following constraint for each time step $t \in \{1, ..., T\}$.

$$\alpha_{u,i}^{d,t} = 0 \tag{8}$$

Function activation. To know which functions are activated on which nodes in the final routing. For each Node $u \in V$, Function $f \in F$, Demand $d \in D$, Layer $i \in \{0, ..., |c_d| - 1\}$,

$$\alpha_{u,i}^{d,T} \leq z_{u,f_i^{c_d}}^{T}. \tag{9}$$

Note that we do not consider the cost of potential activations of VNFs during the reconfiguration process. Indeed, our goal is to minimize the network operational cost over time and the reconfiguration duration is very small in comparison in an SDN network [17].

C. *Static* R&P*:* `Breaking-Bad`

To solve the static R&P problem, we can use the previous ILP with zero reconfiguration step: $T = 0$. We also have to remove the make-before-break constraints, Constraints 6 and 7. The complete ILP is written and described in [16].

D. R&P *for a single demand*

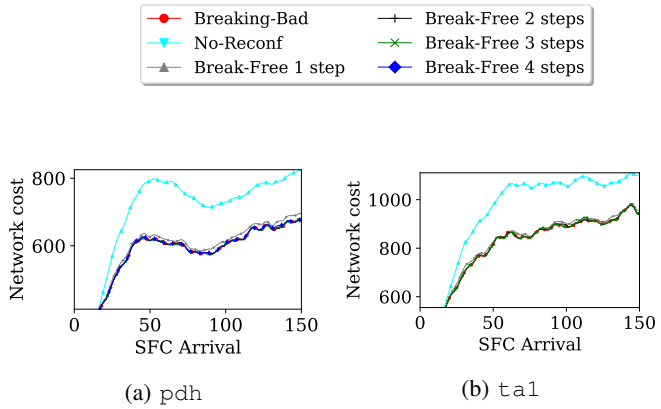
Note first, that even routing a single demand is NP-hard as it is equivalent to find a shortest Weight-Constrained Path [18] in the layered graph as link and node capacities are shared between layers [15]. A solution is to use the ILP for static R&P in which all the demands routed in the past are fixed. The ILP routes the demand (if possible) with the goal of minimizing the additional needed cost without exceeding the available network resources. The complete ILP could be found in [16]. Another possibility is to adapt the Label-setting algorithm based on dynamic programming [19] for the the Weight-Constrained Shortest Path Problem.

V. Numerical Results

In this section, we evaluate the performance of our proposed solution, `Break-Free`. We study the impact of the reconfiguration on different metrics such as cost savings, acceptance rate, and resource usage. We compare the results with the ones of `Breaking-Bad`, which computes an optimal R&P for the whole set of requests for each SFC arrival and with `No-Reconf` which computes the R&P problem for a single demand, the newly arrived SFC. We consider two scenarios, one with low traffic in which basically all demands can be accepted and one with high traffic in which some have to be rejected. In the low traffic scenario, we can fairly compare resource usage using the different algorithms `Break-Free`, `Breaking-Bad`, and `No-Reconf`, as they are accepting the same demands. In the high traffic scenarios, we can compare them in terms of acceptance rate of demands.

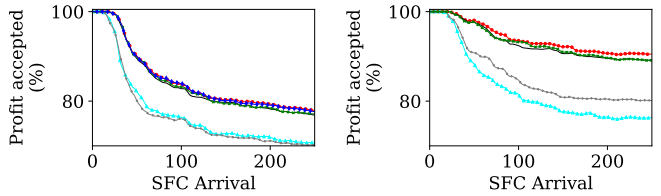
A. Data sets

We conduct experiments on two real-world topologies from SNDlib [20] of different sizes: `pdh` (11 nodes, 34 links), and `ta1` (24 nodes, 55 links). We generate our problem instances as follows. We considered 250 demands for each network. The source and destination of each demand are chosen uniformly at random among the nodes. Following [21], the lifetime of a demand is exponentially distributed with mean $\mu = 25$. We then round this lifetime to an integral number of time steps. The volume of the demands is chosen randomly and is on average 2 times higher in the high-traffic scenario than in the low-traffic one. Also, each demand is associated with an ordered sequence of 2 to 3 functions uniformly chosen at random from a set of 5 different functions. Experiments have been conducted on an Intel Xeon E5520 with 24GB of RAM.



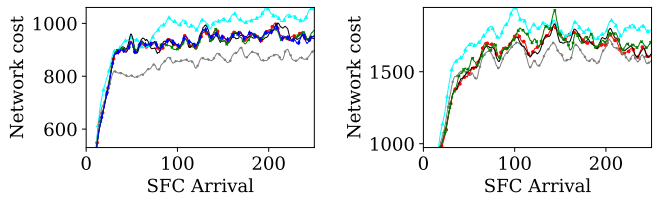
(a) pdh (b) ta1

Fig. 2: Low-Traffic scenario - Network cost.



(a) pdh (b) ta1

Fig. 3: High-Traffic scenario - Percentage of accepted profit across time.



(a) pdh (b) ta1

Fig. 4: High-Traffic scenario - Cost gain across time.

B. Low-traffic scenario - Resource usage

In Figure 2, we show the network cost for our *low-traffic scenario*. We first see that Break-Free has similar network operational cost as Breaking-Bad, which interrupts the requests at each SFC reconfiguration. This means that our solution is as efficient as possible as Breaking-Bad provides a lower bound of the best our algorithm can achieve. Moreover, Break-Free achieves this performance for any number of time steps (even 1). This leads to a very fast algorithm as discussed below. Indeed, when the network is not congested, there is enough capacity to host both the old and new routes.

Reconfiguration leads to a better resource utilization and reduces the network operational cost compared to No-Reconf, and this given a same volume of traffic. Indeed, reconfiguring the network regularly permits a reduction of 19% of network operational cost while using 22.5% fewer VNFs and 18.5% less link bandwidth compared to No-Reconf.

C. High-Traffic scenario - Acceptance Rate

In our *high-traffic scenario*, there are not enough resources to satisfy all the demands. As a consequence, some requests cannot be accepted. We show, in Figure 3, the profit achieved by Break-Free, Breaking-Bad, and No-Reconf. We define the profit associated with an accepted demand as the asked volume of bandwidth multiplied by its duration. The global profit is defined as the sum of all the accepted requests' profits. We show the profit as a percentage in terms of maximum achievable profit. It can be seen that No-Reconf and Break-Free (with 1-step) lead to equivalent profit, around 70% for pdh (and between 78 and 81% for ta1), while Break-Free (with 2, 3, and 4 steps) and Breaking-Bad have similar performances (around 80% for pdh and 90% for ta1). For this congested scenario, one step of reconfiguration is not enough as there is not enough place to move the requests. Therefore, some requests are rejected. Allowing to use more steps in our *make-before-break* reconfiguration process, without interrupting the requests, we can reach the same performances as Breaking-Bad.

In Figure 4, we show the network operational cost as a function of the number of demands arrived. The first observation is that Break-Free (with more than 2-steps) leads to a smaller network operational cost than No-Reconf. It accepts more, with less cost. The second observation is that even if Break-Free (with 1-step) has a similar profit to No-Reconf, it has substantially less network operational cost than all the other algorithms.

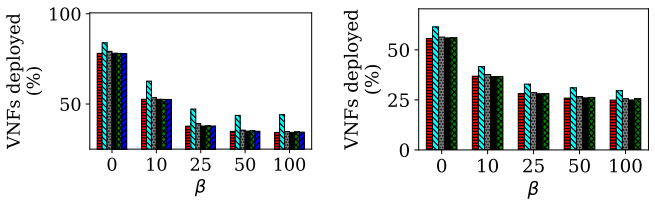
D. Impact of Parameter β

In Figures 5, we study the impact of β on the number of deployed VNFs. We focus on the low-traffic scenario as we can compare the algorithms for the same global volume of traffic. As β increases, the impact of the VNF cost on the total cost is greater. As a consequence, the number of deployed VNFs decreases, leading to longer routes and thus, to an increased amount of bandwidth necessary in the network to route the demands in order to satisfy their chaining constraints. Note also that, for all values of β, reconfiguration using Break-Free (for any number of steps) leads to similar gains to reconfiguration using Breaking-Bad. This shows that the conclusion discussed in Section V-B for a specific value of $\beta = 25$ (our default value) is valid in more general settings for a wide range of β.

Another important observation is that the gain of reconfiguration is higher for larger values of β. The reason is that, when β is large, the requests tend to use longer routes as the cost of bandwidth is less important compared to the one of VNFs. On the contrary, when β is small, the routes try to always use close to shortest path solutions, leading to lower gains. There is still a gain as a shortest path is not always available (due to node and link capacities).

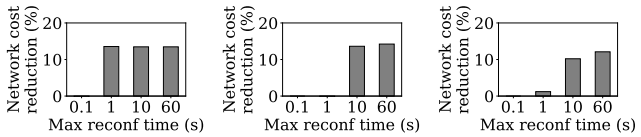
E. Time limits for the reconfiguration

Figure 6 shows the gains of network cost (compared to No-Reconf) in percentage for Break-Free (1 to 3 steps) when limiting the time spent for the reconfiguration.



(a) pdh (b) ta1

Fig. 5: Low-Traffic scenario - Impact of parameter β - VNFs deployed as a function of β.



(a) B-Free (1 step) (b) B-Free (2 steps) (c) B-Free (3 steps)

Fig. 6: Low-Traffic scenario - Gains of network operational costs for different time limits for the optimization process.

Break-Free with 1 step needs only 1 second to reach its best solution. This variant of the algorithm is almost as fast as Breaking-Bad (which does not compute an intermediate make-before-break step). 10 seconds are needed to reach a close to optimal solution for the 2–step variant, and a good solution for the 3–step variant. The best solution is attained after 1 minute. We remind the reader that in the low-traffic scenario, the 1–step variant is enough to achieve solutions close to optimal, while in the high-traffic scenario, this is the case of the 2 step variant. It is thus possible to reconfigure a network without interruption and with significant gain in a few seconds.

F. Percentage of rerouted requests

To see the importance of implementing a make-before-break process, we study the percentage of rerouted requests during the reconfiguration process. We report in Figure 7 (left) the percentage of reconfigured SFCs for Break-Free (1 to 4 steps) and Breaking-Bad for the high-traffic scenario. Firstly, Breaking-Bad has to interrupt, on average, 78% of the requests (between 45% and 100%) to maintain an optimal solution. This is thus of *crucial importance to avoid impacting this large number of requests when reconfiguring.* Break-Free changes the routing of approximately the same number of requests but without any interruption of traffic. For the 1-step case, the reconfiguration is not always possible as the network is almost saturated. Therefore, when reconfiguration is achieved, it impacts slightly fewer requests: 65% on average for pdh and 70% for ta1.

VI. Conclusion

In this work, we provide a solution, Break-Free, to reconfigure a set of requests which have to go through service function chains. The requests are routed greedily when they arrive, leading to a sub-optimal use of network resources,

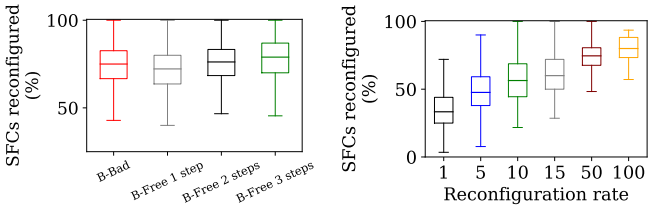


Fig. 7: Percentage of rerouted requests for ta1, considering (left) different intermediate reconfiguration steps and (right) different reconfiguration rates.

bandwidth, and virtual network functions. Break-Free reroutes the requests to an optimal or close to optimal solution while providing a make-before-break mechanism to avoid impacting the rerouted requests. Our algorithm is fast and provides a close to optimal reconfiguration in a few seconds. As future work, we will propose algorithms to handle larger instances.

References

[1] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, 2013.

[2] P. Quinn and T. Nadeau, "Problem statement for service function chaining," Tech. Rep., 2015.

[3] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, "VNR algorithm: A greedy approach for virtual networks reconfigurations," in *IEEE GLOBECOM*, 2011.

[4] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE TNSM*, 2016.

[5] R. Mijumbi *et al.*, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, 2016.

[6] S. Paris, A. Destounis, L. Maggi, G. S. Paschos, and J. Leguay, "Controlling flow reconfigurations in sdn," in *IEEE INFOCOM*, 2016.

[7] S. Ayoubi, Y. Zhang, and C. Assi, "A reliable embedding framework for elastic virtualized services in the cloud," *IEEE TNSM*, 2016.

[8] M. Ghaznavi *et al.*, "Elastic virtual network function placement," in *IEEE CloudNet*. IEEE, 2015.

[9] V. Eramo *et al.*, "An approach for SFC routing and virtual function network instance migration in NFV architectures," *IEEE/ACM Transaction in Networking*, 2017.

[10] J. Liu, W. Lu, F. Zhou, P. Lu, and Z. Zhu, "On dynamic service function chain deployment and readjustment," *IEEE TNSM*, 2017.

[11] B. Augustin, T. Friedman, and R. Teixeira, "Measuring load-balanced paths in the internet," in *ACM SIGCOMM conference on Internet measurement*, 2007.

[12] N. Garg and J. Koenemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," *SIAM Journal on Computing*, 2007.

[13] Y. Sang, B. Ji, G. R. Gupta, X. Du, and L. Ye, "Provably efficient algorithms for joint placement and allocation of virtual network functions," in *IEEE INFOCOM*, 2017.

[14] W. Ma *et al.*, "Traffic aware placement of interdependent NFV middleboxes," in *IEEE INFOCOM*, 2017.

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

[16] A. Gausseran, A. Tomassilli, F. Giroire, and J. Moulierac, "Don't interrupt me when you reconfigure my sfcs," Inria, Tech. Rep., 2018.

[17] P. Berde *et al.*, "Onos: towards an open, distributed sdn os," in *Workshop on Hot topics in software defined networking*. ACM, 2014.

[18] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[19] S. Irnich and G. Desaulniers, "Shortest path problems with resource constraints," in *Column generation*. Springer, 2005, pp. 33–65.

[20] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "Sndlib 1.0survivable network design library," *Wiley Networks*, 2010.

[21] S. Sahhaf *et al.*, "Network service chaining with optimized network function embedding supporting service decompositions," *Computer Networks*, 2015.