Poster: Don't Interrupt Me When You Reconfigure my Service Function Chains

Adrien Gausseran, Andrea Tomassilli, Frédéric Giroire, Joanna Moulierac

Poster: Don't Interrupt Me When You Reconfigure my Service Function Chains

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

***Abstract*—Network Functions Virtualization (NFV) enables the complete decoupling of network functions from proprietary appliances and runs them as software applications on general–purpose servers. Service Function Chains (SFC) are paths with an ordered sequence of network functions that have to be processed. In this paper, we consider the problem of reconfiguring SFCs 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 SFC 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.**

I. Introduction & State of the art

Network flows are often required to be processed by an ordered sequence of network functions. This notion is known as Service Function Chaining (SFC) [8]. The application of SFCs often requires the usage of two different independent technologies, Software Defined Networking (SDN) and Network Function Virtualization (NFV), which are complementary.

SDN aims at simplifying network management by decoupling the control plane from the data plane, making it easier to manipulate network devices through a software program [4]. As a consequence, the network becomes programmable. With the NFV paradigm, network functions can be implemented in software, becoming Virtual Network Functions (VNFs) that can be instantiated and scaled on–demand without having to install new equipment and executed on generic-purpose servers located in small cloud nodes. Since traffic is dynamic, the allocation of a demand is performed individually without having full knowledge of the incoming traffic. This may lead to a fragmented network which uses more resources than necessary. Also, it may lead to a higher blocking probability even though there are enough resources to serve new demands. Therefore, we must take it into consideration and adjust network configurations in response to changing network conditions. Thus, our problem is to reroute traffic flows through the network and to improve the mapping of network functions to nodes in the presence of dynamic traffic. Our objective is to minimize the network operational cost, defined as the sum of the link bandwidth cost to route the demands and the cost for all the installed VNFs in the network.

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.
ISBN 978-3-903176-16-4 2019 IFIP

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. 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). When the network is at the new state, the transmission moves to the new route and the resources used by the initial one are released. This strategy is often referred to as *make-before-break*.

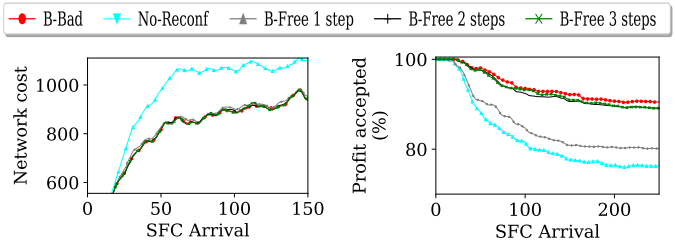
a) State of the art: Paris *et al.* [7] study the problem of online SDN controllers to decide when to perform flow reconfigurations, but network function requirements are not considered in their work. The closest study to our work is from Liu *et al.* [5]. They consider the problem of optimizing VNF deployment and readjustment to efficiently orchestrate dynamic demands. But an important unaddressed issue concerns the revenue loss of an operator due to the QoS degradation occurring when demands are reconfigured. Duong *et al.* [2] apply this reconfiguration strategy in optical networks using column generation, but they only allow one request to be moved at a given time.

II. Problem Statement and Modeling

We consider that the flows are splittable as it is frequent to have load balancing in networks [1]. We consider the SFCs one by one, and find a route which minimizes the *additional network operational cost* to be paid. We then reconfigure the network with a *make-before-break* mechanism 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.

We compare the results of `Break-Free` with `Breaking-Bad` that breaks the flows before rerouting them, implying packet losses and QoS degradation. When a reconfiguration has to be carried out, `Breaking-Bad` considers basically a static setting with the requests present in the network and finds an optimal Routing & Provisioning solution without considering the current setting.

a) Presentation of `Break-Free`*: reconfiguration without interruption:* Our idea consists in improving the current solution, getting as close as possible to the optimal one while being achievable by a *make-before-break* reconfiguration. To this end, we set a number of intermediate reconfiguration steps,



(a) Low-Traffic scenario - Network operational cost.

(b) High-Traffic scenario - Percentage of accepted profit.

Fig. 1: Results for `ta1` for two different cases.

T, and the goal of the optimization is to find a routing with minimal cost, which can be reached from the current routing using T reconfiguration steps. At time 0, the configuration is the current one (i.e., paths and function locations for all the SFCs). Then, at each step of reconfiguration, a valid solution (respecting the link and node capacities) is computed, and some SFCs may have two active configurations. As a single step of reconfiguration may not be enough, the ILP has several intermediate reconfiguration steps. The objective function is to minimize the network operational cost of the final state. The value of T is an important parameter. Indeed, a value too small may lead to models with poor solutions, while a value too large to models with prohibitive execution times. The model is based on the notion of layered graph with $|c_d|$ layers (where $|c_d|$ denotes the number of VNFs in the chain). The model is not detailed here due to lack of space but may be found in [3].

III. Numerical Results

We study the impact of the reconfiguration by comparing the results of `Break-Free` (with reconfiguration steps from 1 to 3) with the ones of `Breaking-Bad`, and `No-Reconf` which never reconfigures the SFC. We consider two scenarios: the first one with low traffic in which we plot the network operational cost, and the second one, with high traffic scenario, all the demands cannot be accepted, and we study the acceptance rate of demands. The experiments are conducted on a real-world topology from SNDlib [6]: `ta1` (24 nodes, 55 links). We considered 250 demands for each network. The source and destination of each demand are chosen uniformly at random among the nodes. Following [9], the lifetime of a demand is exponentially distributed (rounded to an integral number of time steps). When the lifetime is reached, the demand leaves the network. The volumes of the demands are chosen randomly and are 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. The demands are routed one by one, greedily, when they arrive.

a) Low-traffic scenario - Network cost: In Figure 1, we first see that `Break-Free` has similar performances to `Breaking-Bad`. Recall that `Breaking-Bad` provides a lower bound for our algorithm as it interrupts the SFCs. Moreover, `Break-Free` achieves this performance for any number of time steps (even 1). Indeed, when the network is not congested, there is enough capacity to host both the old and new routes and to perform efficient reconfiguration. Reconfiguration leads to a better resource utilization compared to `No-Reconf`. Reconfiguring regularly permits a reduction of 19% of network operational cost while using 22.5% fewer VNFs and 18.5% less link bandwidth.

b) High-Traffic scenario - Acceptance profit rate: We show the profit achieved by the three algorithms in Figure 1b. We define the profit associated with an accepted demand as the asked volume of bandwidth multiplied by its duration. 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 81%), while `Break-Free` (with 2 and 3 steps) and `Breaking-bad` have similar performances (around 90%). For this congested scenario, one step of reconfiguration is not enough as there is not enough space to move the requests, thus the requests are rejected. If we increase the number of steps of reconfiguration for `Break-Free`, we can reach the same performances as `Breaking-Bad`.

As a conclusion of these results, `Break-Free` allows to lower the network cost and to increase the acceptance rate almost as much as `Breaking-Bad`. Reconfiguration has to be done often to attain a significant gain showing that the make-before-break mechanism is crucial to avoid an impact on the QoS. Further simulation results are given in [3], such as the impact of the frequency of reconfiguration or the impact of setting a time limit for the resolution of the linear program.

IV. 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, 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. However, as a future work, we plan to propose algorithms to handle larger instances.

References

[1] B. Augustin, T. Friedman, and R. Teixeira. Measuring load-balanced paths in the internet. In *ACM SIGCOMM IMC*, 2007.
[2] H. Duong, B. Jaumard, D. Coudert, and R. Armolavicius. Efficient make before break capacity defragmentation. *IEEE HPSR*, 2018.
[3] A. Gausseran, A. Tomassilli, F. Giroire, and J. Moulierac. Don't interrupt me when you reconfigure my sfcs. Technical report, Inria, 2018.
[4] H. Kim and N. Feamster. Improving network management with software defined networking. *IEEE Communications Magazine*, 2013.
[5] J. Liu, W. Lu, F. Zhou, P. Lu, and Z. Zhu. On dynamic service function chain deployment and readjustment. *IEEE TNSM*, 2017.
[6] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski. Sndlib 1.0–survivable network design library. *Wiley Networks*, 2010.
[7] S. Paris, A. Destounis, L. Maggi, G. S. Paschos, and J. Leguay. Controlling flow reconfigurations in sdn. In *IEEE INFOCOM*, 2016.
[8] P. Quinn and T. Nadeau. Problem statement for service function chaining. Technical report, 2015.
[9] S. Sahhaf and et al. Network service chaining with optimized network function embedding supporting service decompositions. *Computer Networks*, 2015.