Scheduling Service Function Chains with Dependencies in the Cloud

Tarek Menouer  
Umanis Research & Innovation  
7 Rue Paul Vaillant Couturier,  
92300 Levallois-Perret, France  
tmenouer@umanis.com

Amina Khedimi, Christophe Cérin  
Université Sorbonne Paris Nord  
LIPN/CNRS UMR 7030,  
93430 Villetaneuse, France  
{khedimi,cerin}@lipn.univ-paris13.fr

Mohammed Chahbar  
Université Sorbonne Paris Nord  
L2TI - Institut Galilée  
93430 Villetaneuse, France  
mohammed.chahbar@univ-paris13.fr

Abstract—Cloud services are now well established, thanks to some providers’ pioneering work that currently offer on-premise the advantage of the predictability, continuity, and quality of service delivered by virtualization technologies. In this context, SDN (Software Defined Networking) aims to provide tenant-controlled management of forwarding and different abstractions of the underlying network infrastructure to the applications. The scheduling and placement of network functions in the cloud is a challenging task. One reason is that it also requires tedious provisioning and configuration steps. Even if we consider in this paper only the placement of network functions and not their configurations, we are faced with the general problem of defining, in an ‘optimal’ way, the placement of network functions to be executed so that some criteria are preserved. In this paper, we formulate an approach to schedule network functions according to their dependencies.

Index Terms—Cloud network and virtualization, Virtualization of Network Equipment, Network Function Virtualization, Service Function Chain, Scheduling, Optimization.

I. INTRODUCTION

Service Function Chaining (SFC) enables composite services constructed from one or many Service Functions (SFs). These functions include traditional network SFs such as firewalls and IP Network Address Translators (NATs), as well as application-specific functions. These also include WAN and application acceleration, deep packet inspection, HOST_ID injection, HTTP header enrichment functions and TCP optimizer.

The definition and instantiation of an ordered set of SFs (the literature also uses the network functions name) and subsequent network traffic passing through them is precisely termed Service Function Chaining (SFC) by IETF according to RFC 7665 [5] and RFC 7498 [12] that serve as guidelines in our paper. Note that these two RFCs are informational and correspond to one level in the non-standard track maturity levels of the IETF [2]. An 'Informational' specification is published for the general information of the Internet community, and does not yet represent an Internet community consensus or recommendation.

In Figure 1, we present a SFC composed of three SFs between the traffic source and its destination.

As an example, one fundamental building block of the SFC architecture is the Service Function Forwarder (SFF) aiming to forward packets (according to classification) and frames received from the network to one or more SFs associated with a given SFF using information conveyed in the SFC encapsulation as it is presented in Figure 2. Traffic from SFs eventually returns to the same SFF, which is responsible for injecting traffic back onto the network. The collection of SFFs and their associated SFs creates a service-plane overlay.

In this paper, conceptually, we consider that both SFF and SF can be virtualized and, more precisely, that SFFs are instances of SFs. In other words, for the sake of simplicity, we consider that an SFC is made only of SFs, i.e., SF is a superset of SFFs and SFs in the traditional sense.

The objective of this paper is to provide a methodology to schedule SFCs with dependencies on a cloud architecture. The novelty of the article concerns the management of dependencies.

The organization of the paper is as follows. Section II introduces the related works regarding scheduling and placement of SFs. In Section III, we explain how to solve this problem according to our modeling. Section IV concludes the paper.
II. RELATED WORK

The problem of Virtual Network Functions (VNF) scheduling and placement is catching the interest of the scientific literature. Some works deal with placement and scheduling of network functions; others deal only with function placement. All have different approaches and priorities and also propose a separate resolution methods for different objectives. Most of them consider the static setting of VNF placement and offer an integer linear programming (ILP) model to obtain the optimal solution.

In [15], authors addressed the problem of allocating Service Functions Chains (SFC) on an NFV Infrastructure composed of Points of Presence (PoPs) with VNF hosting capabilities. They proposed a general cost-driven Integer Linear Programming (ILP) for the SFC allocation problem, and a graph-based heuristic, for large scale instances, inspired by the Viterbi algorithm to select the minimum set of PoPs and then allocate the SFC requests to them.

An ILP formulation and a heuristic approach based on the relaxation of the optimal ILP formulation are also proposed in [1] for the VNF Placement and Routing (VNF-PR) problems for Service Function Chaining. The authors considered the partial orders of VNFs within each SFC and the anti-affinity rules between functions where two VNFs cannot handle the same service chain on the same node.

In [6], the authors considered the placement of SFs and corresponding routing to maximize the acceptable flow rate and minimize the energy cost in an SFC-enabled network. They propose a multi-objective optimization problem for the placement of SFs and flow distribution. The problem is then transformed on single-objective mixed-integer linear programming (MILP) problem using the ε-constraint method and a rounding-based heuristic are proposed. To join the two objectives, they determine the number of SF instances and their locations, and the SC (Service Chain) flows routes.

Authors in [10] mixed the VNF placement and chaining in networks, and cloud environments. They propose an analytical eigendecomposition based approach and a heuristic based on a custom greedy algorithm for both physical and virtual network function placement and chaining.

The dynamic placement of VNFs is less studied. Authors in [13] propose an integer Linear Program based heuristic for an on-line VNFs placement and chaining to minimize physical resource usage taking into account the ability to share functions between multiple requests. This work has been extended in [14]. They proposed an ILP for an on-line and batch mode VNFs placement and chaining. According to the stakeholders’ preference, the model can be tuned to several objectives such as power consumption and cost minimization, load balancing, and physical resource usage minimization. To reduce the problem’s complexity, they started by selecting the potential candidate’s nodes from the infrastructure to host the VNF using a bipartite graph matching algorithm. The ILP is then solved by the Dijkstra algorithm using CPLEX solver. The batch mode consists of selecting the more pertinent requests to be served in priority to generate higher revenues for the provider.

Authors in [8] proposed mixed-integer linear programming (MILP) for the VNF mapping and scheduling problem. To deal with NP-hardness of the MILP, they proposed a two-stage online algorithm. The first stage aims to map and schedule the VNFs on a service function chain (SFC) by minimizing the waiting time of VNFs. A rescheduling scheme taking into account the delay is then triggered, in which the selected existing VNFs are remapped and rescheduled in the second stage. This dynamic approach performs flexible function placement.

Authors in [3] addressed the problem of the availability of SFC with a minimum number of off-site backup VNFs service. They proposed an online algorithm to minimize resource consumption for service providers while an SFC request acceptance ratio is maximized.

The network slicing concept has been introduced in [9]. The authors dealt with the problem of resource requirements of Service Function Chains belonging to a slice network. They proposed a slice resource provisioning approach to ensure SFC resource allocation. The problem is formulated as a mixed-integer linear programming problem taking into account the radio coverage constraints part of the SLA with the tenant, and other constraints related to the network infrastructure.

In [4, 7], authors focused on the NFV resource allocation in their state of the art and proposed a classification of the existing approaches to solve this problem based on their metrics, objectives, and the nature of the placement (static or dynamic).

In summary, most of the presented works proposed a static approach based on a MILP (or an ILP) to get the optimal solution and an heuristic algorithms to deal with large instances, for the VNF placements.

III. SCHEDULING SERVICE FUNCTION CHAINS

Once the order is established, we traverse the SFs to check if the dependency relation is satisfied or not, to allocate the SF. At this point, our algorithm verifies the existence of edges in the graph representing the dependencies between the SFs to be placed on the node returned by the bin-packing algorithm and all the existing functions already set on this node. If so, the algorithm goes to the next node (see Label in Algorithm 1). If all the nodes have been visited, then we reject the SFC. This statement means that we also have a system to control starvation. This system is based on the management of priorities. It has been put in place in a previous work [11]. For the sake of simplicity, we do not put all the details in this paper.

As a discussion about the method, we would like to remind that this paper aims to manage dependency between SFs. An optimization of this family of algorithms, in terms of the best compromise between the response time and the
quality of the allocation will be considered in our future works.

Algorithm 1 SFC placement based on FIFO management

Require: $S_{f}$: set of Service Functions Chain requests
Require: $N$: set of nodes
Ensure: $S$: set of tuples $(f_{i}, N_{0<j<=n})$

1. $S \leftarrow \emptyset$
2. for $s = G(F,L) \in \text{orderSFCRequestsByFIFO}$ do
3. first $\leftarrow$ orderByFIFO($F$).
4. $n \leftarrow N$
5. for $f \in \text{first}$ do
6. Label: $N^* \leftarrow \text{BINPACK}(n)$
7. if $(\_, N^*) \notin S$ then
8. $S \leftarrow S \cup (f,N^*)$
9. else
10. $ft \leftarrow \func_{of L on N^*}(L,N^*)$
11. if $\text{Independence}(f,ft) == \text{true}$ then
12. $S \leftarrow S \cup (f,N^*)$
13. else
14. $n \leftarrow n \backslash N^*$
15. if $n$ not empty then
16. goto Label
17. else
18. reject $s$
19. end if
20. end if
21. end if
22. end for
23. end for

IV. Conclusion

In this paper, we proposed the algorithm of scheduling Service Function Chains with dependencies in the Cloud. Due to the ongoing migration and hosting in the Cloud of software-defined networking technologies, it is essential to control the deployment of Service Functions Chains to optimize cloud resources. We added one difficulty in our modeling in considering dependencies between functions. The placement is more challenging than in considering independent functions, i.e., namely, bag-of-tasks in the scheduling terminology.

As future works, we propose to experiment with our approach in a cloud infrastructure with realistic benchmarks. The objective will be to study the impact of the number of dependencies and the placement strategy on the performance.

REFERENCES