An Availability-aware SFC placement Algorithm for Fat-Tree Data Centers

Ghada Moualla, Thierry Turletti, Damien Saucez

An Availability-aware SFC placement Algorithm for Fat-Tree Data Centers

Ghada Moualla, Thierry Turletti, Damien Saucez
Université Côte d'Azur, Inria, France

***Abstract*—Complex inter-connections of virtual functions form the so-called Service Function Chains (SFCs) deployed in the Cloud. Such service chains are used for critical services like e-health or autonomous transportation systems and thus require high availability. Respecting some availability level is hard in general, but it becomes even harder if the operator of the service is not aware of the physical infrastructure that will support the service, which is the case when SFCs are deployed in multi-tenant data centers. In this paper, we propose an algorithm to solve the placement of topology-oblivious SFC demands such that placed SFCs respect availability constraints imposed by the tenant. The algorithm leverages Fat-Tree properties to be computationally doable in an online manner. The simulation results show that it is able to satisfy as many demands as possible by spreading the load between the replicas and enhancing the network resources utilization.**

***Keywords*-SFC, Availability, Cloud, Data Center.**

I. Introduction

Network Function Virtualization (NFV) [1] virtualizes network functions and places them into commodity network hardware, such as a Data Center (DC). Since a single VNF cannot provide a full service, multiple VNFs are combined together in a specific order, called Service Function Chains (SFCs) [2]. SFCs determine the sequence of NFs that packets must follow and optimization techniques are used to map the SFCs in the network without overloading it and to provide availability guarantees.

Replication mechanisms have been proposed in the literature (e.g., [3], [4], [5]) to improve the required service availability based on VNF redundancy, which allow configurations in Active-Backup or Active-Active modes. However, some propositions [6] focus on replicating the SFCs in multi-tenant data centers where the tenant demands are oblivious to the actual physical infrastructure of the Data Center. Such an environment is particularly challenging as the demand is not known in advance and cannot be controlled. For the data center operator, it is therefore important to limit the number of replications to its minimum, yet respecting the level of service agreed with its tenants.

In this paper, we propose a placement algorithm for SFCs in Data Centers relying on Fat-Tree topologies. The algorithm is run by the network hypervisor and guarantees that Service Level Agreements (SLAs) with the tenants are respected, given the availability properties of the hardware deployed in the data center. Our proposition is based on an iterative linear program that solves the placement of SFCs in an online manner without prior knowledge on placement demand distribution. The algorithm is made computationally doable by leveraging symmetry properties of Fat-Tree topologies. Our evaluation on a very large simulated network topology (i.e., 27,648 servers and 2,880 switches) shows that the algorithm is fast enough for being used in production environments.

The rest of the paper is organized as follows. Section II presents the related work. Section III describes the problem statement. Section IV proposes an availability-aware placement algorithm. Finally, Section V evaluates the performance of our solution and Section VI concludes the paper.

II. Related Work

Multiple works tackle the problem of robust VM placement by deploying them on different physical nodes using specified availability constraints [7], [8], [9]. Zhang et al. [9] and Sampaio et al. [10] consider the MTBF of DC components to propose high availability placements of virtual functions in DCs. However, none of these works consider the benefits of using redundancy to ensure reliability. Rabbani et al. [11] solve the problem of availability-aware Virtual Data-Centers (VDC) embedding by taking into account components' failure rates when planning the number and the place of redundant virtual nodes but they do not consider the case of service chains.

In Herker et al. work [12], SFC requests are mapped to the physical network to build a primary chain, and backup chains are decided based on that primary chain. Engelmann et al. [13] propose to split service flows into multiple parallel smaller sub-flows sharing the load and providing only one backup flow for reliability guarantee. Our work follows the same principle as these two propositions but uses an active-active approach such that resources are not wasted for backup.

In this paper, we propose a stochastic approach for the case where SFCs are requested by tenants oblivious to the physical DC network and that only have to provide the SFC they want to place and the required availability.

III. Problem Statement

This section defines the problem of placing SFCs in Data Centers under availability constraints.

Without any loss of generality, and inspired by works ([12], [14]), we only consider server and switch failures and ignore link failures. We also consider that all equipment of a same type have the same level of availability. This work develops an availability-oriented algorithm for resilient placement of VNF service chains in Fat-Tree based DCs where component failures are common [12].

The Fat-Tree topology is modeled as a graph where the vertices represent switching nodes and servers, while the edges represent the network links between them. Furthermore, SFC provides a chain of network functions with a traffic flow that need to traverse them in a specific order. We only consider acyclic SFCs. As we are in a multi-tenant scenario, functions are deployed independently and cannot be aggregated (i.e., function instances are not shared between SFC instances or tenants).

Each function is considered as a single point of failure. Thus, to guarantee the availability of a chain we use *scaled replicas*: we replicate the chain multiple times and equally spread the load between the replicas.

Upon independent failures, the total ***availability*** for the whole placed SFC replicas will be computed using Eq. (1) (availability for parallel systems).

$$availability = 1 - \prod_{i \in n\_replicas} (1 - ava_{sc_i}), \quad (1)$$

where ava_{sc_i} is the availability of replica i of service chain sc and $n\_replicas$ is the number of scaled replicas for this service chain. The availability of each service chain replica ava_{sc_i} is defined by

$$ava_{sc_i} = \prod_{f \in F} A_f, \forall i \in n\_replicas \quad (2)$$

where A_f is the availability of a service chain function f, which corresponds to the availability of the physical node that hosts this function ([6], [4]). Details on how to compute A_f can be found in [15].

IV. SFC Placement Algorithm

We approximate the placement problem with the following algorithm that computes placements on pods instead of being on the whole topology. Our algorithm is called each time a request to install an SFC is received. Specifically, for a *required availability* R, the algorithm determines how many scaled replicas to create for that SFC and where to deploy them; taking into account the availability of network elements (servers and switches) without impairing the availability guarantees of the chains already deployed. To guarantee the isolation between scaled replicas, each replica of a chain is deployed in a different fault domain.

Algorithm 1 presents the pseudo-code of our algorithm where `scale_down(`C`, `n`)` is a function that computes the scaled replica scheme, i.e., an annotated graph representing the scaled down chain, for a chain C if it is equally distributed over n scaled replicas and where `solve_placement(`S`, `G`, `n`)` solves the problem of placing n replicas S on the network topology G. The solution of a placement is a set of mappings associating replica functions and the compute nodes on which they have to be deployed. The solution is empty if no placement can be found.

Our algorithm starts with one replica of a service request and first checks that no function is requesting more resources than what the pod can offer.

In the case it is not possible to find a placement with one replica, the algorithm scales down the chain S by adding one more replica and tries to find a placement for each one of these replicas in different fault domains. Otherwise, the algorithm tries to find a placement for it under one fault domain of the network (the fault domain is chosen randomly to spread the load over the entire DC) using `solve_placement(`S`, `G`, `n`)` function; if no placement is found in the current fault domain, we check the another fault domain, otherwise we compute the total availability for the current placement.

This strategy continues until a termination condition is met: (i) if the requested availability is reached then the service can be deployed with (`deploy` (*placement)*); (ii) if the maximum acceptable time for finding a placement is reached then no solution is found; (iii) if the number of created scaled replicas reached the maximum number of replicas (i.e., maximum number of fault domains), then no solution is found. The `compute_ava` function computes the availability of a chain placement according to Sec. III.

The `solve_placement(`S`, `G`, `n`)` function considers two graphs: the DC topology graph G and the scale replica graph S. The purpose of this function is to project the scaled replica graph S on the topology graph G with respect to the physical and chain constraints.

For each fault domain, `solve_placement(`S`, `G`, `n`)` tries to find a valid placement for the scale replica graph in one fault domain while maximizing the availability of the replica placement. The exact formulation of the problem can be found in [15].

V. Evaluation

In the following, we evaluate the Availability-aware placement algorithm introduced in the previous section.

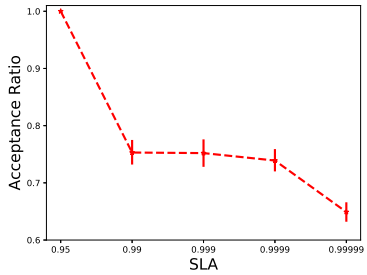

Figure 1. Comparing acceptance ratio for these different SLA values: 0.95, 0.99, 0.999, 0.9999, 0.99999.

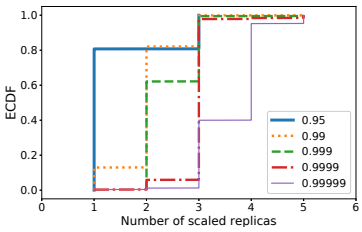

Figure 2. The ECDF for the number of created replicas with 5 different SLA with 48-Fat-Tree topology, $T_{IA} = 0.01$ and $S = 100$.

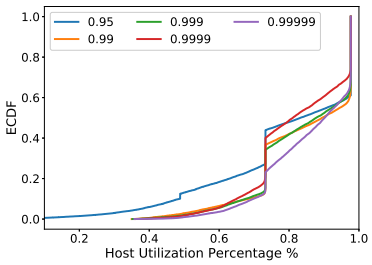

Figure 3. The ECDF for the host core utilization for different SLAs with 48-Fat-Tree topology, $T_{IA} = 0.01$ and $S = 100$.

Algorithm 1: Availability-aware placement

Input: Physical network Graph: G
chain $\in$ Chains
Scaled chain replica graph: *replica_scheme*
Required availability: R

T = *max_time*; $n = 1$; *tot_ava* = 0; *tot_time* = 0;
placement = ϕ; *replica_scheme* = *chain*
while *tot_ava*<*R* ***and*** *tot_time*<*T* ***and*** *n*<*max_n* **do**
  if *max_req* > *max_ava* **then**
    $n = n + 1$
    replica_scheme = `scale_down` (*chain, n*)
  else
    placement = `solve_placement` (*replica_scheme, G, n*)
    $n = n + 1$
    if ***not*** *placement* **then**
      replica_scheme = `scale_down` (*chain, n*)
    else
      tot_ava = `compute_ava` (*placement*)
  tot_time.update()
if *tot_ava* $\geq$ *R* **then**
  `deploy` (*placement*)

A. Simulation Environment

We implemented a simulator in Python interfaced with the Gurobi Optimizer 8.0 solver. All simulations have been run on a Intel i7-4800MQ CPU at 2.70GHz and 32GB of RAM running GNU/Linux Fedora core 21.[1]

In the evaluation, requests to deploy SFCs are independent and follow an exponential distribution of mean inter-arrival time T_{IA} (measured in arbitrary time units). SFCs have a service time of S time units, i.e., the time the SFC remains in the system is randomly selected following an exponential distribution of mean S. If an SFC cannot be deployed in the network, it will be rejected. In total, our synthetic workload contains 2,000 SFC request arrivals made of 20 random SFCs. All experiments presented here were repeated 5 times (5 different workloads of 2,000 SFC requests).

Furthermore, all SFCs are linear, i.e., they are formed of functions put in sequence between exactly one start point and one destination point. The number of NFs between the two endpoints is selected uniformly between 2 and 5, based on typical use cases of networks chains [16], and the requirements of each function in terms of cores is 1, 2, 4, or 8. Simulations are performed on a 48-Fat-Tree topology (i.e., 48 pods, 27.648 hosts). Every host has 4 cores. Similarly to [12] and [14] the availability is 0.99 for servers, 0.9999 for ToR and aggregation switches, 0.99999 for core switches, and 1.0 for links.

In the evaluation, each SFC requires the same SLA even though the algorithm does not enforce it. In practice, placements must be computed in reasonable time so we limited the computation time to at most 6s per request, above that requests are rejected.

B. Acceptance Ratio

The required availability level has an impact on the ability of a network to accept or not SFC requests. To study this impact, we consider the *acceptance ratio* defined as the number of accepted SFC requests over the total number of requests.

Figure 1 shows the evolution of the acceptance ratio w.r.t. the 5 different SLA levels. We can notice that the acceptance ratio decreases when the required availability level increases as each chain must reserve more resources than for lower availability levels as the physical topology is kept untouched. This can be explained by the fact that when increasing the required availability of a chain, it

[1] All the data and scripts used in this paper are available on https://team.inria.fr/diana/robstdc/.

is necessary to replicate it further and then to consume more resources as at least one core is attributed to each function, replicated or not.

C. Level of Replication

Figure 2 provides the Empirical Cumulative Distribution Function (ECDF) of the number of scaled replicas created for accepted SFCs for the different studied SLAs. It is clear that for the lowest required availability (0.95), 80% of SFCs were satisfied with exactly one replica as the availability of network elements are higher than this SLA level. However, when a SFC request needs more resources than the available resources in the network, it is split (20% of service were split for SLA=0.95) and, as the required availability increases, the required number of replicas are increased to satisfy the SFC SLA. Interestingly, as in practice the availability of the infrastructure is high, we can observe than even for an aggressive SLA of 0.9999, 90% of SFC requests can be satisfied with no more than 3 replicas. Figure 2 shows that the number of replicas tops to 5 even though in theory it would be possible to observe up to 48 replicas in a 48-Fat-Tree topology as there are 48 pods. We can explain this, as the computation time of our optimization is restricted to be less than 6 seconds.

Nevertheless, we can observe that a general increase of availability requirement increases the required number of replicas, which explains why the acceptance ratio decreases when the availability requirements increase.

D. Servers utilization

Figure 3 shows the ECDF of the server core utilization where the host utilization is the ratio between the total consumed CPU time and the total CPU time offered by the server. In all scenarios, more than 40% of the servers are fully occupied. When the required availability is as high as 0.99999, more than 80% of servers are more than 80% occupied. Nevertheless, even in highly loaded infrastructures, our algorithm can allocate resources in order to satisfy as much demands as possible.

VI. Conclusion

In this paper, we propose an online algorithm for SFC placement in data centers that leverages the Fat-Tree properties and respects the SFC availability constraints dictated by the tenant, taking into account the network components availability. The simulation results show that our algorithm is fast enough for being used in production environments and is able to satisfy as many demands as possible by spreading the load between the replicas while improving the network servers CPU utilization at the same time. For future work, we plan to extend our solution to consider other data center topologies, such as Leaf-and-Spine and BCube.

Acknowledgments

This work is funded by the French ANR through the Investments for the Future Program under grant ANR-11-LABX-0031-01 (LABEX UCN@Sophia).

References

[1] ETSI, "Network Function Virtualisation (NFV); Architectural Framework," *NFV 001*, 2013.

[2] J. M. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture," RFC 7665, Oct. 2015.

[3] F. Machida, M. Kawato, and Y. Maeno, "Redundant virtual machine placement for fault-tolerant consolidated server clusters," in *IEEE NOMS*, 2010.

[4] J. Fan, C. Guan, K. Ren, and C. Qiao, "Guaranteeing availability for network function virtualization with geographic redundancy deployment," Tech. Rep., 2015.

[5] F. Carpio *et al.*, "Replication of virtual network functions: Optimizing link utilization and resource costs," in *IEEE MIPRO*, 2017.

[6] F. Carpio and A. Jukan, "Improving reliability of service function chains with combined vnf migrations and replications," *arXiv preprint arXiv:1711.08965*, 2017.

[7] M. Mihailescu, A. Rodriguez, and C. Amza, "Enhancing application robustness in Infrastructure-as-a-Service clouds," in *IEEE/IFIP DSN Conference*, pp. 146–151.

[8] D. Jayasinghe, C. Pu *et al.*, "Improving performance and availability of services hosted on IAAS clouds with structural constraint-aware virtual machine placement," in *IEEE SCC Conference*, 2011.

[9] Q. Zhang *et al.*, "Venice: Reliable virtual data center embedding in clouds," in *IEEE INFOCOM*, 2014.

[10] A. M. Sampaio *et al.*, "Towards high-available and energy-efficient virtual computing environments in the cloud," *Future Gener Comput Syst*, 2014.

[11] M. G. Rabbani *et al.*, "On achieving high survivability in virtualized data centers," *IEICE T COMMUN*, 2014.

[12] S. Herker *et al.*, "Data-center architecture impacts on virtualized network functions service chain embedding with high availability requirements," in *IEEE Globecom Workshop*, 2015.

[13] A. Engelmann *et al.*, "A reliability study of parallelized vnf chaining," *arXiv preprint arXiv:1711.08417*, 2017.

[14] P. Gill *et al.*, "Understanding network failures in data centers: measurement, analysis, and implications," 2011.

[15] G. Moualla, T. Turletti, and D. Saucez, "An Availability-aware SFC placement Algorithm for Fat-Tree Data Centers," Inria, Research Report, Aug. 2018. [Online]. Available: https://hal.inria.fr/hal-01859599

[16] W. Liu *et al.*, "Service function chaining (SFC) general use cases," *IETF I-D draft-liu-sfc-use-cases-08*, 2014.