VNF Placement and Sharing in NFV-based Cellular Networks

VNF Placement and Sharing in NFV-Based Cellular Networks

Francesco Malandrino[1] *and Carla Fabiana Chiasserini*[2]

[1] *CNR-IEIIT, Torino, Italy*
[2] *Politecnico di Torino, CNR-IEIIT, Torino, Italy*

Introduction

Network function virtualization (NFV) has revolutionized the concept of service, conceiving it as a set of software, namely, virtual, interconnected functions, referred to as virtual network functions (VNFs): the service data traffic then flows through the sequence of VNFs composing the service, and is processed by each them. In parallel, new generation cellular networks have become capable not only to efficiently transfer traffic but also to process and store data, thus they can effectively implement a large variety of services that may be requested by the so-called verticals, i.e. content providers, automotive, or e-health industries. Deploying services within the cellular network instead of the cloud can indeed bring significant advantages, among which, lower service latencies, local data processing (hence, lower bandwidth consumption due to data transfer through the network infrastructure), and lower energy consumption.

Network providers and vertical industries have therefore built a new relation, regulated by Service Level Agreements (SLA), which define the level of quality of service that a network provider has to ensure in order to match the fee paid by a vertical. As a consequence, upon receiving a service request, a network provider has to put into place service deployment strategies that allow the fulfillment of the target key performance indicators (KPIs), e.g. throughput, delay, or reliability, while minimizing the cost, i.e. the amount of resources necessary to run the service. In the following paragraphs, we describe in more detail these two important issues and briefly discuss some relevant works that have dealt with them.

Target KPI. To effectively address the target KPIs, network providers can resort to network slices (NGMN Alliance 2016), i.e. reserve a suitable amount of computing, network, and memory resources for a service, or a set of services with no isolation requirements and similar target KPIs. As an example, such safety services as forward collision warning or intersection collision avoidance (Malinverno et al. 2019) by the same automotive industry, can be implemented using virtual machines (VMs) in servers, along with network links for data transfer, that are sufficiently capable to match millisecond-order latencies. Creating a network slice thus implies that the network provider needs to identify the VMs where to place the VNFs composing the service,

Wiley 5G Ref. Edited by Rahim Tafazolli, Chin-Liang Wang and Periklis Chatzimisios.

DOI: 10.1002/9781119471509.w5GRef264

how much CPU and memory to assign to each of such VMs, and which links to use to transfer data traffic from one VNF to the next one, if they are located at different servers. This problem, often referred to as VNF placement, is typically formulated as a Generalized Assignment Problem (GAP) (Cohen et al. 2015), which minimizes the cost assignment of VNFs to VMs subject to capacity and KPIs constraints. Optimization is indeed an especially popular approach, aiming at minimizing load imbalance (Hirwe and Kataoka 2016), network utilization (Kuo et al. 2016), or energy consumption (Pham et al. 2017). Other works have built more complex cost functions, accounting for several network-related aspects (Mechtri et al. 2016; Gu et al. 2016) and/or energy consumption (Marotta and Kassler 2016; Khoury et al. 2016).

Optimization approaches usually result in a mixed-integer linear programming (MILP) formulation; since MILP problems are impractical to solve in real-world scenarios, the aforementioned works have envisioned finding near-optimal solutions through heuristic approaches. An alternative approach is represented by works tying VNF placement to more general problems, e.g. shortest path with limited resources (Martini et al. 2015), and set covering (Tomassilli et al. 2018). These works prove competitive ratio properties for the algorithms they propose, and such algorithms are also valid for problems other than VNF placement. The recent work (Bega et al. 2019) aims at simplifying the problem of VNF placement by foreseeing the evolution of the load to be served; specifically, the authors observe that the needed capacity is easier to foresee than the actual traffic demand, and employ deep neural networks to that end.

Cost minimization. To minimize deployment costs, instead, one may exploit the fact that several services may have one or more VNFs in common, i.e. services may be composed of a same VNF sub-graph, corresponding to a child service (Rost et al. 2017). This implies that several services, as well as several slices, may contain common sub-slices (5G PPP Architecture Working Group 2017; IETF 2017), and that such sub-slices can be shared instead of being replicated for each single service. For instance, the aforementioned safety services may share the same LTE eNB or 5G gNB when their service coverage overlap; similarly, they can share the same database where the vehicles' information, like position, speed, or heading, can be stored before data are processed (Rost et al. 2017). Importantly, VNF sharing is a multi-faced problem, which requires that the network operator not only identifies which VNFs (sub-slices) can be shared and among which services, but also how the amount of CPU and memory assigned to the VMs implementing them should be set so as to still fulfil the target KPIs of all involved services. VNF sharing is a new aspect that has been scarcely addressed so far. In 2017, the brief contribution in (Yi et al. 2017) proposed a scheduling scheme for VNF-based networks where the same VNF instance may be used for multiple services. In the same year, Soualah et al. (2017) mentioned VNF sharing as one of several viable strategies to improve the energy efficiency of networks. More recently, Malandrino et al. (2019) has identified VNF sharing as a distinct problem from VNF placement, arising within datacenters (points of presence, PoPs) and thus calling for different decisions on (i) whether a given VNF instance should be shared, (ii) how much capacity it shall be assigned, and (iii) the priority to give to different services sharing the same VNF instance.

In this article, we first summarize in the section titled "Related Issues" some important issues related to VNF placement and sharing. Then in the section titled "System Model and Decisions to Make" presents the 5G-PPP 5G reference architecture, highlighting the role of the orchestrator, as well as possible multi-access edge computing (MEC)-based

architectures, where, due to resource scarcity, effective service deployment is utmost important. In the section titled "System Model and Decisions to Make" also introduces the main quantities that need to be taken into account, the decision to be made in terms of resource scaling and service/traffic priority setting, and the objective one seeks to optimize. In the section titled "The FlexShare Algorithm" describes the state-of-the-art solution strategy FlexShare (Malandrino et al. 2019), which, as mentioned, can make effective decisions in polynomial time, thus permitting a swift and efficient network and system management. In the section titled "Numerical Results" discusses some numerical results, obtained through FlexShare, considering real-world services and realistic network scenarios. Finally, we draw our conclusions and highlight possible directions for future work in the section titled "Conclusions".

Related Issues

It is worth mentioning that both VNF placement and VNF sharing are related to the network (NGMN Alliance 2016) concept and are actually part of a network slice creation process. Indeed, network (NGMN Alliance 2016) is a network paradigm whereby the same physical infrastructure – including networking and computing equipment – is concurrently used by multiple services, each of which is guaranteed isolation from the others. Earlier works like Zhang et al. (2017) and Rost et al. (2017) focus on architectural aspects, including the entities in charge of making decisions on infrastructure usage.

Specifically, Zhang et al. (2017) identifies mobility management as one of the main challenges arising in NGMN Alliance (2016)-based 5G networks, due to their multi-RAT (radio-access technology) nature. Indeed, 5G networks embed several different radio access technologies, from Wi-Fi to mmWave, with significantly different characteristics, e.g. availability and reliability. This poses two challenges when users move from a network to another: first, ensuring that the new network is consistent with the user's quality-of-service (QoS) needs; second, ensuring that the end-to-end, service performance experienced by the user is not jeopardized under the new network. The latter becomes especially complex when slices have not only data transfer capabilities but also processing ones, e.g. VNFs deployed on edge servers: the edge servers may be too far away from the new RAT the user is connected to, and a VNF migration may be triggered. The result is that handover procedures in 5G are much more complex than their counterparts in 4G/LTE, and require deeper coordination among the involved actors.

Focusing on the same multi-RAT scenario, Rost et al. (2017) focuses on scheduling issues arising from the integration of different network technologies. Among the challenges to tackle, the authors identify the need to decide at which level RATs shall be shared, e.g. whether MAC-level decisions for different RATs shall be made jointly by centralized entities or locally at the individual RATs. Furthermore, the authors raise the issue of network (NGMN Alliance 2016) request brokerage: since spectrum is a finite resource and overprovisioning is impossible, an admittance strategy for new slice requests must be defined. Upon denying a slice request, the authors envision that the network may reply with a counter-offer, suggesting to offer a different RAT that is more plentiful and offer comparable performance.

Other works, including Samdanis et al. (2017) and Vassilaras et al. (2017), tackle instead the problem of how those decisions should be made, including algorithmic and complexity challenges. Indeed, Samdanis et al. (2017) identifies software-defined networking (SDN) and NFV as two of the main enabling technologies network (NGMN Alliance 2016), which itself one of the main innovations of 5G networks. The authors also point out the potential challenges due to the increased complexity of the network control plane, and identify two main ways to tackle such a complexity: on the one hand, using network slice templates to reduce the number of decisions to make; on the other, crafting efficient and effective decision-making algorithms. Relatedly, Vassilaras et al. (2017) identifies two main decisions to be made concerning network (NGMN Alliance 2016), namely (i) which resources should be used to build the slice and (ii) how to connect them. The authors map both decisions to solving a virtual network embedding (VNE) problem, whereby the physical infrastructure at the operator's disposal is mapped to the virtualized slice to build. Such a problem is NP-hard, as proven via a reduction from the multi-way separator problem; indeed, even deciding the connection between resources is NP-hard, as proved via a reduction from the unsplittable multicommodity flow problem.

The process of creating, updating, and deleting network slices is known as orchestration. The work (Li et al. 2018), supported by the European project 5G-TRANSFORMER, identifies service orchestration as one of the main tasks 5G networks have to perform. Orchestration decisions need to account for a variety of different factors, including service requirements, the available infrastructure, and inter-operator agreements concerning resource sharing (multi-domain federation). To make all these decisions, the authors envision a three-layer architecture, whereby a vertical slicer (VS) translates business-related service goals into technical slice requirements, a service orchestrator (SO) selects the resources to use, possibly from different domains, and a mobile transport platform (MTP) manages the virtual and physical hardware. The entity in charge of most orchestration decisions is the SO, implements and extend the functionality standardized by ETSI in standard NFV-MAN 001 ("Management and Orchestration").

The authors of Santos et al. (2017) consider multi-domain federation scenarios, and address the unique security challenges therein. A first one concern information: on the one hand, operators need to exchange information in order to make effective orchestration decisions; on the other, they do not want to disclose critical information on the (mis)configuration of their own infrastructure, and the global resources at their disposal. A second one is represented by isolation: while all services should be transparent to each other, some may require stronger isolation, e.g. by avoiding to share VNF instances with other services. Such stronger isolation increases security – in the example, a malformed or malicious input at one service cannot compromise the other – but increase the resource consumption and therefore the cost.

As discussed earlier, services can be decomposed into virtual network functions (VNFs), which can then be placed at different locations throughout the infrastructure. In many relevant scenarios, services can be described as sequences or chains of VNFs that incoming flows have to traverse; in this case, the problem of placing VNFs across the available hosts takes the name of VNF chaining. Works talking VNF chaining usually aim at optimizing a cost function, subject to constraints concerning the available resources and the target delays (Pham et al. 2017). Liu et al. (2017) follows a similar approach but also accounts for the coexistence between already-deployed and newly

requested services, which may share some of their VNFs; the resulting optimization problem is solved via a column generation-based approximation. Other works aim at bringing into the picture additional real-world limitations, including memory access issues in multi-core servers, e.g. Zheng et al. (2019). The authors formulate the problem as a non-linear, integer optimization problem; then, in view of its complexity, follow a heuristic approach whereby a performance drop index is defined, measured, and used to efficiently make near-optimal decisions.

More recent works have accounted for additional metrics beyond sheer performance, the most relevant of which is reliability. As an example, Zhang et al. (2019a,b) pursues the goal of resilience to server failures, and tackles the problems of (i) how many additional VNF instances shall be created, (ii) where they shall be placed, and (iii) how many resources they should be assigned to. The resulting problem is NP-hard and is solved through a greedy, iterative algorithm. Chemodanov et al. (2019) considers instead the physical location of servers (especially those at the network edge), and makes VNF placement decisions able to guarantee the geographical availability of services. Making such decisions requires to solve a multi-commodity-chain flow (MCCF) problem, which is tackled through a metapath composite variable approach, reaching near-optimal performance in the average case.

Several recent works tie network (NGMN Alliance 2016) and resource allocation with other problems, hitherto considered separate or orthogonal to it. As an example, Mandelli et al. (2019) MAC-level scheduling, with the objective of ensuring that network slices are able to deliver the data they process to the users needing them. In a similar spirit, Zhang et al. (2019a,b) jointly addresses the problems of choosing (i) the best location within the network to process the data at, and (ii) the best radio technology to deliver the results to the users, in order to reduce interference and increase the throughput.

Jošilo and Dán (2019) addresses a hybrid scenario, where individual devices can choose between performing their computation tasks themselves or leveraging the resources available at the network edge. The authors model the resulting interaction between devices and network operators a Stackelberg game, and propose several decentralized algorithms converging to the equilibrium. Liu and Han (2019) addresses cross-domain (also known as federation) scenarios, whereby different mobile operators decide to pool their resources in order to provide their services with a lower cost. In this context, the authors propose a distributed decision-making algorithm based (i) decomposing the original problem into subproblems via the alternating direction method of multipliers (ADMM) method, and (ii) solving the individual subproblems via learning-assisted optimization (LAO).

System Model and Decisions to Make

Several NFV-architecture for 5G networks have been proposed; examples include the one envisioned by the EU 5GPPP (and further interpreted by the EU 5G-PPP 5G-TRANSFORMER project), those proposed by the NGMN alliance and the IETF, and those considered in the works addressing VNF placement such as Cao et al. (2016), Cohen et al. (2015), Agarwal et al. (2018), Einziger et al. (2019). These architectures foresee that all decisions on VNF placement and resource allocation are made by a

centralized entity, namely the Network Function Virtualization Orchestrator (NFVO) (see the ETSI Management and Orchestration (MANO) framework (ETSI 2014; IETF 2017). Thus, it is the NFVO that makes fine-grained decisions on the allocation and usage of individual hosts and links.

With reference to VNF placement in real-world implementations, it is important to underline that ETSI (2016) specifies four hierarchical levels, namely,

- individual host;
- zone, defined as a set of hosts with similar features;
- zone group, defined as a set of zones;
- point of presence (PoP), e.g. a datacenter.

Real-world 5G networks, also demonstrated through testbeds (Antevski et al. 2018; Sayadi et al. 2016; De la Oliva et al. 2018), consider that the NFVO makes VNF placement decisions at the PoP level, while different entities are in charge of placement and sharing decisions within each PoP (such entities are referred to using different names depending on the standardization body or association, e.g. ETF (IETF 2017), NGMN alliance (NGMN Alliance 2017), or 5G-PPP (5G PPP Architecture Working Group 2017).

In the following, we focus on the architecture foreseen by ETSI and reported in the 5GPPP document entitled "View on 5G Architecture" (2019). With reference to such proposal, the pictorial representation of the architecture in Figure 1 underlines the relationship of Software Defined Networking (SDN) and MANO controllers and their relation with the network, storage, and computing resources, abstracted through the Virtual Infrastructure Manager (VIM). The VIM summarizes the details of such deployed units as containers and VMs. Importantly, the VIM also abstracts the edge computing or multi-access edge computing (MEC) resources. The SDN controller's task is to correctly configure the network, while the MANO's task is to control storage and computing resources, both acting on behalf of the NFVO. Slice management can be part of the

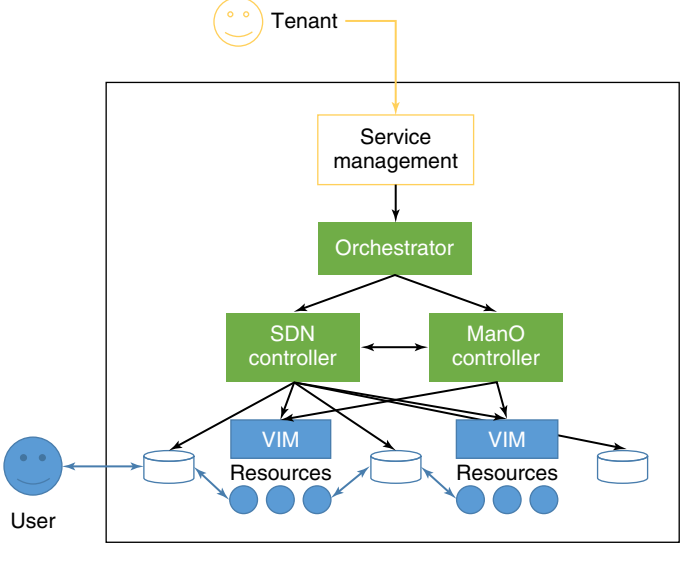

Figure 1 ETSI architecture as reported in 5G PPP Architecture Working Group (2017).

NFVO's tasks, or another entity can be included with this specific purpose. Finally, the service management block should interact with tenants, and the same or similar entities with vertical industries requiring a service to be deployed.

In this scenario, it is worth emphasizing how MEC resources can be managed and allocated. We again refer to the ETSI specifications (ETSI 2018). Possible scenarios all foresee one or more edge PoPs, each hosting a server and leveraging radio access functionalities. Such servers can be directly connected to points of access of the Radio Access Network (RAN), or to entities such as the Evolved Packet Core (EPC) in LTE, i.e. dealing with IP packets. Importantly, the ETSI NFV MANO considers applications at the mobile edge as regular VNFs. It follows that the MEC computing, storage, and network resources have to be orchestrated as well as the deployment of VNFs composing services that require to be in the MEC must be carefully instantiated so as to meet the service KPIs (e.g. ultra-low latency) and maximize the resource efficiency utilization. The latter is indeed particularly critical at the MEC, due to the limited available resources therein.

Turning our attention back to the problem of VNF placement and sharing, given the above architecture and upon receiving a new service request, the NFVO should make decisions about:

- whether any of the VNFs composing the newly requested service shall be provided through existing instances of such VNFs;
- if existing instances can be reused, how to set the priorities for the traffic flows belonging to the services sharing the same VNF instance;
- else, which VM to exploit to deploy the VNF instance;
- how to scale up or scale down the computing resources assigned to the VMs within the PoP.

Below, we will focus on a single PoP and consider that the VNFs composing a service should be instantiated or be co-located within the PoP. This implies that network latencies due to traffic flows transiting from one VNF to another can be neglected, thanks to the high-speed switching that is possible between VMs within the same datacenter (Xia et al. 2017). As a consequence, the only contribution that is relevant to the latency KPI is the data processing time within the VNFs composing the service.

Relevant Quantities for VNF Placement and Sharing

Without loss of generality, we consider that VNFs instances are deployed within VMs and that each VM can host exactly one VNF instance; also, VNF are assumed not to require isolation.

As considered in many recent works (Cohen et al. 2015; Agarwal et al. 2018; Bhamare et al. 2017), each VNF instance running within a VM is represented as an M/M/1 queue with FIFO queueing and preemption. For simplicity, we focus on adapting computing capabilities of VMs and neglect instead memory and storage. In order to reflect real-world conditions, we account for the fact that the computing resources assigned to a VNF instance can be varied (i.e. scaled up/down). For instance, a VNF requiring 1 computational unit and running on a VM with capability equal to 1 unit, it takes 1 time unit to process the traffic associated to the service that includes that VNF. Using instead that VM for a VNF requiring 2 computational units leads to a processing time of 2 time units. Importantly, the requirement values do not depend on the service that

includes that VNF, but only on the VNF itself. Crucially, varying the amount of computing resources per VM influences the processing time at flows at the VNF hosted by that VM. As in real-world implementation, the amount of computational resources used by a VM cannot exceed a given maximum value.

As mentioned, services may be composed of a number of VNF; in the following we consider as target KPI the maximum average delay of a service, although the model and discussion can be extended to other KPIs as well.

Objective and Constraints

Deployment cost is one of the main concerns for the mobile network providers as well as for the vertical industries. Such a cost typically consists of two components: the cost for a VM instantiation, which is a fixed contribution, and a variable cost, which depends on the computing resources consumed by the VM hosting a given VNF instance (a proportional dependency is commonly assumed).

The main constraints instead to account for when making placement and scaling decisions are:

- at most one instance of at most one VNF can run at each VM;
- VMs cannot be scaled beyond their maximum capability;
- all traffic of all services must be served;
- per-service processing time targets must be honored.

The last constraint is especially important to honor and complex to formulate: indeed, the service time experienced by flows of a given service at a VNF depend upon (i) the computational capability available to the VNF; (ii) the arrival rate of flows (of any service) to the VNF; and (iii) the priority of the service, relative to other services sharing the same VNF.

Due to the many factors to account for, as well as the sheer number of available options, making optimal decisions about VM scaling and VNF placement is exceedingly complex, and impractical in most real-world scenarios. To this end, in the section titled "System Model and Decisions to Make" we present a simpler, efficient and effective solution strategy called FlexShare, able to perform near-optimal decisions in a short – namely, polynomial – time.

The FlexShare Algorithm

The problem formulated above is too complex to be directly solved with off-the-shelf solvers like CPLEX or Gurobi. Therefore, Malandrino et al. (2019) proposes an efficient and effective solution methodology named FlexShare, whose high-level approach is summarized in Figure 2. FlexShare considers service requests one by one, and previously made priority decisions – though not placement ones – can be adjusted as new services are deployed.

The first step of FlexShare is described in the section titled "Relevant Quantities for VNF Placement and Sharing", and deals with placement decisions, i.e. which VNF deploy at which VM. To this end, FlexShare builds a bipartite graph, whose nodes correspond to VNFs to deploy and VMs the latter can be deployed to. Edges of the

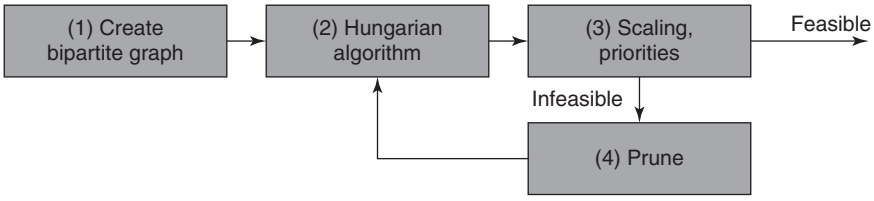

Figure 2 The FlexShare strategy. Step 1 builds a bipartite graph showing which VMs *could* run each VNF. Step 2 runs the Hungarian algorithm on such a graph to obtain the placement decisions. Step 3 solves a convex optimization problem to make scaling and priority assignment decisions. If such a problem is infeasible, the bipartite graph is *pruned* (step 4) and the procedure restarts from step 2.

bipartite graph connect (VNF, VM) pairs such that the newly-requested service can use the VNF instance deployed at the VM. The weights of edges correspond to the cost of providing the VNF with that VM, including the proportional component (which is influenced by the service traffic) and the fixed component (which is not incurred if the VM is already active, that is, if the VNF instance can be shared).

Given the bipartite graph, step 2 leverages the Hungarian algorithm, Kuhn (1955) to obtain the minimum-weight (hence, cheapest) assignment to VNFs to VM. Such an assignment is solely based on the weights on the bipartite graph and, critically, is not guaranteed to be feasible.

In step 3, the decisions made in step 2 are used as constraints of a convex (hence, simple to solve efficiently) optimization problem, detailed in the section titled "Priority, Scaling, and Pruning". The purpose of the problem is to (i) setting the priorities of each service within every VNF, (ii) set the capabilities of each VM, and, crucially, (iii) verify if the placement decisions made in step 2 are feasible. If the optimization succeeds, then FlexShare terminates successfully.

If step 3 fails, i.e. the problem solved therein is infeasible, then FlexShare moves to step 4 and tries to *prune* the bipartite graph. Indeed, an infeasible problem in step 3 can be due to too much sharing in step 2, i.e. too many VNF-to-VM edges in step 1. To correct this, FlexShare removes one of the edges from the bipartite graph, and restarts from step 2. This implies that placement decisions will foresee less VNF sharing and thus have a higher cost, but also a higher likelihood to result in a feasible problem in step 3.

Placement Decisions

On the left-hand side of the bipartite graph created in step 1 of FlexShare, we find the VNFs to place; on the right-hand one, the VMs they can be placed at. Edges are drawn between (VM, VNF) pairs such that the VNF can be provided by the VM. This happens in two cases:

- the VM is currently unused, therefore, a new instance of the VNF can be deployed therein;
- the VM already hosts an instance of the VNF, and such an instance can be shared between already-deployed services and the newly requested one.

The weight of the edge represents the cost of providing the VNF through a given VM; if the VM is currently inactive, the edge weight also includes the fixed activation cost.

Importantly, edges are not drawn if the VM cannot be scaled up to a capacity sufficient to serve the newly requested service while keeping stability. This, however, does not imply that service time requirements are met; indeed, such a condition is checked in step 3 as discussed in the section titled "Priority, Scaling, and Pruning."

Once the bipartite graph is ready, the Hungarian algorithm (Kuhn 1955) is employed to find a minimum-cost matching between VNFs and VMs. Specifically, the Hungarian algorithms selects a set of edges such that (i) each VNF is connected (hence, is deployed) in exactly one VM, and (ii) the total weight of the selected edges is as low as possible. Importantly, the Hungarian algorithm has polynomial (namely, cubic) complexity in the size of the graph. The edges selected by the Hungarian algorithm correspond to placement decisions, including:

- activation of currently inactive VMs, if need be;
- sharing of already-deployed VNF instances, if warranted.

These decisions are fed to the optimization problem in step 3, as described in the section titled "Priority, Scaling, and Pruning."

Priority, Scaling, and Pruning

Step 3 of FlexShare takes as an input the deployment decisions made by the Hungarian algorithm in step 2, and then solves a convex optimization problem where:

- the objective is to minimize the total cost;
- the constraints concern VM capability and per-service, end-to-end target delays;
- the decision variables are the capability to assign to each VM and the priorities to give to each service sharing every VM.

Importantly, all decision variables are real, hence, the problem can be solved in polynomial (cubic) time through commercial solvers (Boyd and Vandenberghe 2004); indeed, embedded convex optimization is routinely used in real-time applications. If the optimization succeeds, i.e. if the problem is feasible, then FlexShare terminates and the placement decisions made in step 2, along with the scaling and priority decisions made in step 3, can be applied.

If the optimization fails, then FlexShare proceeds to step 4, i.e. pruning the bipartite graph. The intuition behind the pruning procedure is that one cause for infeasibility is too aggressive sharing of VNF instances. To fix that, one edge of the bipartite graph is removed; to select such an edge, FlexShare resorts to the irreducible infeasible set (IIS) (Chinneck 2007). The IIS contains all constraints that, if removed, would render the problem feasible; intuitively, it provides an explanation as to why the optimization failed. Among constraints in the IIS, step 4 of FlexShare identifies the one that:

- concerns VM capability;
- involves a VNF used by the newly deployed service;
- involves the VM closest to its maximum capability,

The intuition behind the latter item is that VMs close to their maximum capability are more likely to introduce long delays, hence, lead to infeasible problem instances. By removing the corresponding edge from the bipartite graph, we ensure that such a placement decision is not made in subsequent iterations of FlexShare.

Note that it is possible to prove that the IIS contains at least one VM-capability constraint, hence, it is always possible to perform the procedure in step 4.

Reference Scenarios and Benchmark Strategies

We evaluate the performance of FlexShare using two reference scenarios, namely, a synthetic, small-scale scenario allowing us to perform a comparison against the optimum, and a large-scale scenario including real-world services.

Synthetic Scenario

In order to understand how FlexShare operates and to compare its performance with alternative approaches, we first leverage a simple, synthetic scenario. The scenario includes three services and five VNFs, with a many-to-many relationship among them. Services have different request rates and target delays, as reported in Table 1. The available infrastructure is composed of 10 VMs, with capability varying between 5 and 10 units; all VMs have an activation cost of 8 units and a proportional cost of 0.5 units.

Thanks to its small size, in the synthetic scenario it is possible to compare the performance of FlexShare against the optimum; specifically, optimal decisions are found through brute force.

Real-World Scenario

The FlexShare performance is also studied in a real-world, large-scale scenario, including five services belonging to the domains of smart city and smart factory. The VNFs composing each service, as well as the traffic each of them has to process, are based on Casetti et al. (2018) and Taleb et al. (2014, 2019), as reported in Table 2. The reference topology is the Luxembourg City center (Codeca et al. 2015).

Three services, namely, Intersection Collision Avoidance (ICA), vehicular see-through (CT), and entertainment (CDN) concern vehicles and their drivers/passengers. In ICA, the cooperative awareness messages (CAMs) broadcasted by vehicles are processed by a collision detector in order to check whether some vehicles are set on a collision course and, if so, alert them. All vehicles within 50 m of an intersection send a CAM message every 100 ms.

In the CT service, vehicles can display on their on-board screen a video feed coming from preceding vehicles (e.g. a truck blocking the view), so that their driver can be aware

Table 1 Services in the synthetic scenario.

Service	Arrival rate (flows/ms)	Max. delay (ms)
s_1	2	10
s_2	1.5	7.5
s_3	1	5

Table 2 Services and arrival rate in the realistic scenario.

VNF	Flow arrival rate
Intersection collision avoidance (ICA)	
eNB	117.69
EPC PGW	117.69
EPC SGW	117.69
EPC HSS	11.77
EPC MME	11.77
Car information management (CIM)	117.69
Collision detector	117.69
Car manufacturer database	117.69
Alarm generator	11.77
See through (CT)	
eNB	179.82
EPC PGW	179.82
EPC SGW	179.82
EPC HSS	17.98
EPC MME	17.98
Car information management (CIM)	179.82
CT server	179.82
CT database	17.98
Sensing (IoT)	
eNB	50
EPC PGW	50
EPC SGW	50
EPC HSS	5
EPC MME	5
IoT authentication	20
IoT application server	20
Smart factory (SF)	
eNB	50
EPC PGW	50
EPC SGW	50
EPC HSS	5
EPC MME	5
Robotics control	50
Video feed from robots	5

Table 2 (Continued)

VNF	Flow arrival rate
Entertainment (EN)	
eNB	179.82
EPC PGW	179.82
EPC SGW	179.82
EPC HSS	17.98
EPC MME	17.98
Video origin server	17.9
Video CDN	179.82

of impeding obstacles and/or hazards. Messages are sent every 200 ms by any vehicle within 100 m of an intersection.

The CDN service is used by 10% of all vehicles on the topology, randomly chosen with uniform probability; such vehicles consume a 25-fps video.

For the smart-city domain, an Internet-of-Things (IoT) service is considered, including a total of 200 sensors deployed throughout the topology; as per the 3GPP standard (Taleb et al. 2014), each sensor transmits 10 packets per second.

Finally, smart-factory applications are represented by a smart-robot service, controlling 50 robots in real time (which requires one packet per millisecond per robot); furthermore, 10% of all robots also need to transmit a 25-fps video feed.

As for the operator infrastructure, we assume it contains a total of 10 VMs, whose capability can be scaled up to 1000 units, and whose fixed and proportional cost are (respectively) 1000 units and 1 unit.

Benchmark Strategies

The performance evaluation includes several priority assignment strategies, with different levels of flexibility.

The lowest-flexibility option is represented by service-level priorities (**service** in plots), whereby priority levels are associated to whole services (the lower the delay target, the higher the priority); all requests of a given service have the same priority.

An intermediate option is represented by VNF-level priorities, where different services can have different priority levels at different VNFs (but all requests of the same service in the same VNF have the same priority). Priority levels can be decided through FlexShare (*VNF/FS* in plots) or through brute-force (*VNF/brute*).

Finally, at the highest level of flexibility, there are per-request priorities, assigned via FlexShare (*req./FS* in plots).

All solutions are implemented in Python, and a Xeon E5-2640 server with 16 GByte of RAM is used to run all tests.

Numerical Results

We begin from the synthetic scenario described in the section titled "Synthetic Scenario." Figure 3a shows the cost sustained by the MNO as the traffic changes; such costs become, consistently with our intuition, higher as the traffic grows. It is more interesting to remark that, for a given quantity of traffic, more flexibility always means smaller costs.

In Figure 3b, we turn our attention to sharing, and display the number of services using, on average, a given VNF instance. Again, more flexibility results in more sharing; intuitively, operators are able to more fully utilize their VNF instances, hence, need to deploy fewer of them.

Accordingly, Figure 3c shows that the used VM capability, as well as the maximum capability to which used VMs could be scaled to, decrease with more flexible priority assignments. This is consistent with Figure 3b: how higher flexibility is associated with more sharing, hence a better usage of the deployed VMs, hence the need to deploy fewer VMs.

Moving to the realistic scenario – where, due to its size, comparison with the optimum is impossible – it is possible to observe the same trends. Figure 4a highlights how more flexibility consistently results in smaller costs. Also notice how, when n becomes very high, the costs associated with all strategy overlap; this is because, in very high traffic conditions, no VNF can be shared for any priority assignment.

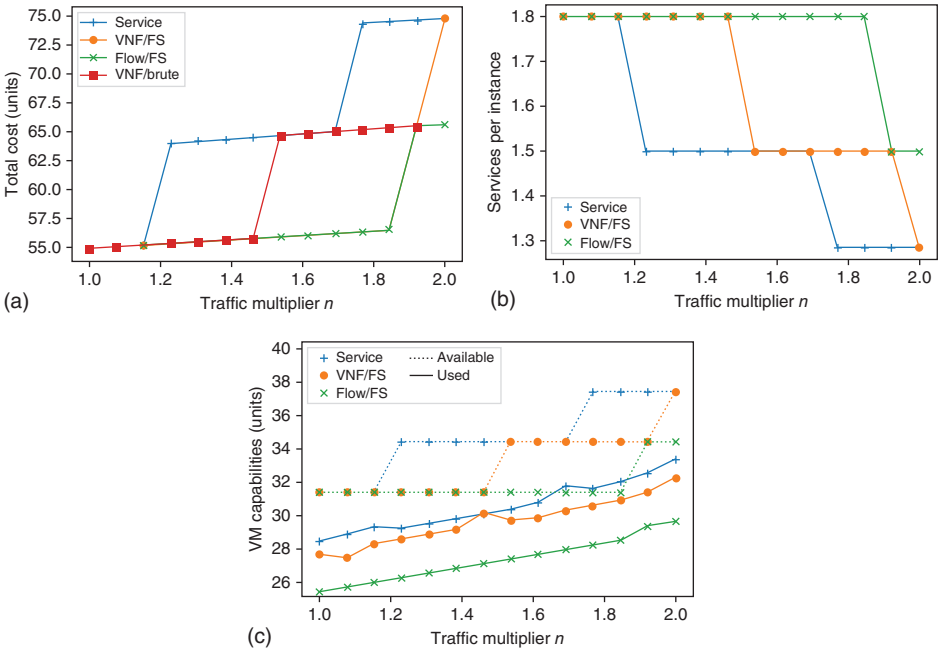

Figure 3 (a) Synthetic scenario: total cost; (b) average number of services sharing a VNF instance; (c) used and maximum VM capability. Per-VNF and per-flow priorities are assigned via FlexShare; per-service priorities are assigned by giving higher priorities to lower-delay services.

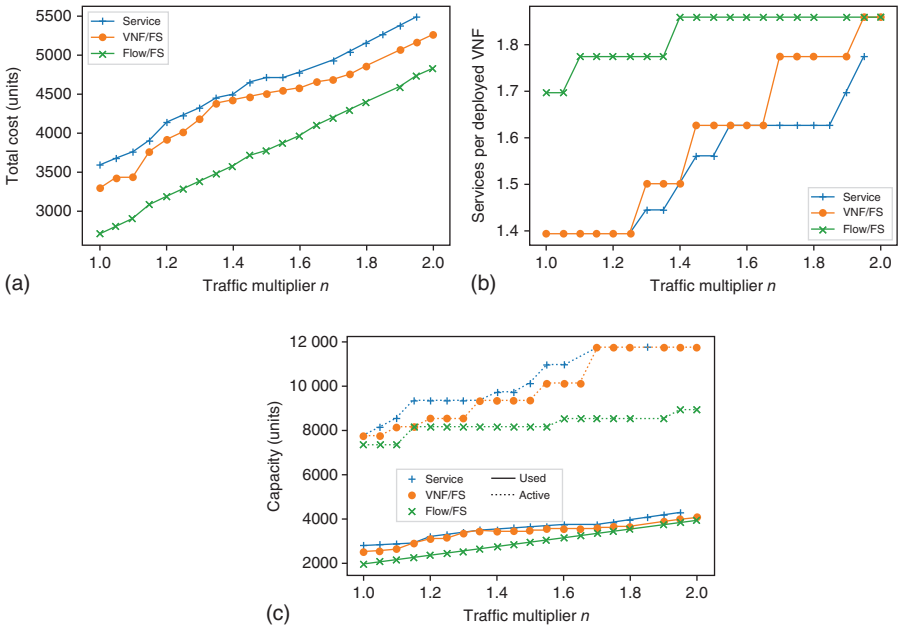

Figure 4 Realistic scenario: (a) total cost; (b) average number of services sharing a VNF instance; (c) used and maximum VM capability. Per-VNF and per-flow priorities are assigned via FlexShare; per-service priorities are assigned by giving higher priorities to lower-delay services.

Consistently with Figure 3b, Figure 4b confirms that more flexibility results in more sharing. Furthermore, the intermediate approach of per-VNF priorities tends to perform better than it does in Figure 3b, which suggests that such an approach can be a viable option in those cases where per-request priorities are too complex to implement.

Moving to VM capabilities, Figure 4c shows a very significant difference between used and potential VM capabilities; in other words, VMs are used *less* efficiently than in the synthetic scenario, for all priority assignments. This may seem surprising; however, recall that, as seen in Table 2, the real-world scenario has fewer common VNFs between services, hence, it presents fewer sharing opportunities in the first place.

We now ask a different question, namely, how efficient FlexShare is, that is, how long it takes to make its decisions. As summarized in Table 3, FlexShare takes at most a few minutes to run, even for the complex, real-world scenario summarized in Table 2. Consistently with our intuitive expectations, FlexShare takes longer to run in the realistic scenario; the main reason is that such a scenario has more alternatives to explore and compare.

It is also interesting to observe how, in general, more traffic tends to correspond to longer run times, but such a trend is by no means monotonic. The explanation for this effect lies in Figure 2. Indeed, the runtime of FlexShare is associated with the number of times the cycle in Figure 2 is repeated, and such a cycle is repeated every time a deployment is found to be infeasible. How often this happens does not depend upon the traffic *per se*, but rather upon how close VMs operate to their maximum capacity.

Table 3 Running time (in minutes) of our FlexShare implementation, in the synthetic and realistic scenarios.

Traffic multiplier	Synthetic scenario	Realistic scenario
1	4	6
1.2	5	7
1.4	6	6
1.6	5	10
1.8	7	9
2	7	12

Finally, it is important to stress that all runtimes can be substantially reduced if need be, by replacing the current Python implementation of FlexShare, which leverages the venerable but old optimization routines of the SciPy library, with more optimized code leveraging more modern, state-of-the-art solvers like CPLEX and Gurobi.

Conclusions

The problem of VNF placement and sharing in NFV-based networks is one of the main issues to overcome in 5G-and-beyond network systems. The support of the services target KPIs, along with the efficiency of resource utilization, are main concerns of both network providers and vertical industries wishing to make their services available to mobile users.

In this article, we first introduced the most widely accepted system architecture and highlighted the main challenges that it poses, along with some solutions that have been proposed to address them. Among the most relevant existing approaches, the FlexShare scheme promises to make swift decisions on resource allocation and sharing among different services, achieving a performance close to the optimum.

Such an approach, however, considered a single PoP, and neglected the network latency due to traffic transfer from one PoP to another. Interesting directions for future research thus include the definition of strategies for making the same VNF instances be reused by several services even when such instances are not deployed within the same datacenter. Furthermore, different target KPIs can be considered, beside the maximum average latency. To tackle this last point and introduce stricter delay guarantees, new modeling strategies are needed. Finally, the envisioned algorithmic solutions should be fully implemented in real-world networks to actually verify their ability to cope with practical issues and requirements.

Acknowledgment

This work was partially supported by the EU 5GROWTH project (Grant No. 856709).

Related Articles

5G Development: Autonomic Network Management and Orchestration
5G Development: Autonomic Networking in the Core Network and Beyond
5G Development: Autonomic Networking in RAN (from C-RAN to small cells)
5G Enabling Technologies and Autonomic Networking (SDN, MEC, NFV, SFC)
Mobility Prediction based Resource Management
Edge/Fog Computing Networks
NFV: Performance Considerations for 5G Networks
Network Management in Programmable Networks
Slicing Across Multiple Operator Domains
Slicing and Radio Resource Management
Slice Deployment and Management for Vertical Industries
Service Chaining using Software Defined Networks
Slicing the Radio Access Network
Vehicular Networks

References

5G PPP Architecture Working Group (2017). View on 5G Architecture.
Agarwal, S., Malandrino, F., Chiasserini, C.-F., and De, S. (2018). Joint VNF placement and CPU allocation in 5G. IEEE INFOCOM.
Antevski, K., Jorge, M.-P., Nuria, M. et al. (2018). Resource orchestration of 5G transport networks for vertical industries. IEEE PIMRC.
Bega, D., Gramaglia, M., Fiore, M. et al. (2019). DeepCog: cognitive network management in sliced 5G networks with deep learning. IEEE INFOCOM.
Bhamare, D., Samaka, M., Erbad, A. et al. (2017). Optimal virtual network function placement in multi-cloud service function chaining architecture. *Computer Communications* 102: 1–16.
Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.
Cao, J., Zhang, Y., An, W. et al. (2016). VNF placement in hybrid NFV environment: modeling and genetic algorithms. IEEE ICPADS.
Casetti, C., Chiasserini, C.F., Molner, N. et al. (2018). Arbitration among vertical services. IEEE PIMRC.
Chemodanov, D., Calyam, P., and Esposito, F. (2019). A near optimal reliable composition approach for geo-distributed latency-sensitive service chains. IEEE INFOCOM.
Chinneck, J.W. (2007). *Feasibility and Infeasibility in Optimization: Algorithms and Computational Methods*. Springer.
Codeca, L., Frank, R., and Engel, T. (2015). Luxembourg SUMO Traffic (LuST) Scenario: 24 hours of mobility for vehicular networking research. IEEE VNC.
Cohen, R., Lewin-Eytan, L., Naor, J.S., and Raz, D. (2015). Near optimal placement of virtual network functions. IEEE INFOCOM.
De la Oliva, A., Li, X., Costa-Perez, X. et al. (2018). 5G-transformer: slicing and orchestrating transport networks for industry verticals. *IEEE Communications Magazine* 56 (8): 78–84.
ETSI (2014). Network functions virtualisation (NFV); management and orchestration.

ETSI (2016). Network functions virtualisation (NFV); management and orchestration; Or-VNFM reference point – interface and information model specification.
ETSI (2018). MEC deployments in 4G and evolution towards 5G.
Einziger, G., Goldstein, M., and Sa'ar, Y. (2019). Faster placement of virtual machines through adaptive caching. IEEE INFOCOM.
Gu, L., Tao, S., Zeng, D., and Jin, H. (2016). Communication cost efficient virtualized network function placement for big data processing. IEEE INFOCOM Workshops.
Hirwe, A., and Kataoka, K. (2016). LightChain: a lightweight optimization of VNF placement for service chaining in NFV. IEEE NetSoft.
IETF (2017). Network slicing management and orchestration.
Jošilo, S., and Dán, G. (2019). Wireless and computing resource allocation for selfish computation offloading in edge computing. IEEE INFOCOM.
Khoury, N.E., Ayoubi, S., and Assi, C. (2016). Energy-aware placement and scheduling of network traffic flows with deadlines on virtual network functions. IEEE Cloudnet.
Kuhn, H.W. (1955). The Hungarian method for the assignment problem. Wiley Naval Research Logistics.
Kuo, T.W., Liou, B.H., Lin, K.C.J., and Tsai, M.J. (2016). Deploying chains of virtual network functions: on the relation between link and server usage. IEEE INFOCOM.
Li, X., Mangues-Bafalluy, J., Pascual, I. et al. (2018). Service orchestration and federation for verticals. IEEE WCNC Workshops.
Liu, Q., and Han, T. (2019). DIRECT: distributed cross-domain resource orchestration in cellular edge computing. ACM Mobihoc.
Liu, J., Lu, W., Zhou, F. et al. (2017). On dynamic service function chain deployment and readjustment. *IEEE Transactions on Network and Service Management* 14 (3): 543–553.
Malandrino, F., Chiasserini, C.F., Einziger, G., and Scalosub, G. (2019). Reducing service deployment cost through VNF sharing. *IEEE/ACM Transactions on Networking* 27 (6): 2363–2376.
Mandelli, S., Andrews, M., Borst, S., and Klein, S. (2019). Satisfying network slicing constraints via 5G mac scheduling. IEEE INFOCOM.
Marotta, A., and Kassler, A. (2016). A power efficient and robust virtual network functions placement problem. IEEE ITC.
Martini, B., Paganelli, F., Cappanera, P. et al. (2015). Latency-aware composition of virtual functions in 5G. IEEE NetSoft.
Mechtri, M., Ghribi, C., and Zeghlache, D. (2016). A scalable algorithm for the placement of service function chains. *IEEE Transactions on Network and Service Management* 13 (3): 533–546.
Malinverno, M., Avino, G., Casetti, C. et al. (2019). MEC-based collision avoidance for vehicles and vulnerable users. *IEEE Vehicular Technology Magazine*, in press.
NGMN Alliance (2016). Description of network slicing concept.
NGMN Alliance (2017). 5G network and service management including orchestration.
Pham, C., Tran, N.H., Ren, S. et al. (2017). Traffic-aware and energy-efficient VNF placement for service chaining: joint sampling and matching approach. *IEEE Transactions on Services Computing* 13 (1): 172–185.
Rost, P., Mannweiler, C., Michalopoulos, D.S. et al. (2017). Network slicing to enable scalability and flexibility in 5G mobile networks. *IEEE Communications Magazine* 55 (5): 72–79.

Samdanis, K., Wright, S., Banchs, A. et al. (2017). 5G network slicing – part 2: algorithms and practice. *IEEE Communications Magazine* 55 (8): 110–111.

Santos, M.A.S., Ranjbar, A., Biczók, G. et al. (2017). Security requirements for multi-operator virtualized network and service orchestration for 5G. In: *Guide to Security in SDN and NFV*. Springer.

Sayadi, B., Gramaglia, M., Friderikos, V. et al. (2016). SDN for 5G mobile networks: NORMA perspective. Springer Crowncom.

Soualah, O., Mechtri, M., Ghribi, C., and Zeghlache, D. (2017). Energy efficient algorithm for VNF placement and chaining. IEEE/ACM CCGRID.

Taleb, T., Ksentini, A., and Kobbane, A. (2014). Lightweight mobile core networks for machine type communications. *IEEE Access* 2, 1128–1137.

Taleb, T., Afolabi, I., and Bagaa, M. (2019). Orchestrating 5G network slices to support industrial internet and to shape next-generation smart factories. *IEEE Network* 33 (4): 146–154.

Tomassilli, A., Giroire, F., Huin, N., and Pérennes, S. (2018). Provably efficient algorithms for placement of service function chains with ordering constraints. IEEE INFOCOM.

Vassilaras, S., Gkatzikis, L., Liakopoulos, N. et al. (2017). The algorithmic aspects of network slicing. *IEEE Communications Magazine* 55 (8): 112–119.

Xia, W., Zhao, P., Wen, Y., and Xie, H. (2017). A survey on data center networking (DCN): infrastructure and operations. *IEEE Communications Surveys and Tutorials* 19 (1): 640–656.

Yi, B., Wang, X., and Huang, M. (2017). A generalized VNF sharing approach for service scheduling. *IEEE Communications Letters* 22 (1): 73–76.

Zhang, H., Liu, N., Chu, X. et al. (2017). Network slicing based 5G and future mobile networks: mobility, resource management, and challenges. *IEEE Communications Magazine* 55 (8): 138–145.

Zhang, J., Wang, Z., Peng, C. et al. (2019a). RABA: resource-aware backup allocation for a chain of virtual network functions. IEEE INFOCOM.

Zhang, Q., Liu, F., and Zeng, C. (2019b). Adaptive interference-aware VNF placement for service-customized 5G network slices. IEEE INFOCOM.

Zheng, Z., Bi, J., Yu, H. et al. (2019). Octans: optimal placement of service function chains in many-core systems. IEEE INFOCOM.

Further Reading

Nguyen, Q.-H., Morold, M., David, K., and Dressler, F. (2019). Adaptive safety context information for vulnerable road users with MEC support. IEEE/IFIP WONS.