Computing at the Edge: But, what Edge?

Luis M. Contreras[‡], Javier Baliosian[*†], Pedro Martnez-Julia[§], and Joan Serrat[†]

[‡]Transport and IP Networks Dept. Telefnica Global CTIO Unit Madrid, Spain,
[*]Universidad de la República, Uruguay,
[†]Universitat Politcnica de Catalunya, Spain,
[§]National Institute of Information and Communications Technology (NICT) Tokyo, Japan,
luismiguel.contrerasmurillo@telefonica.com, baliosian@fing.edu.uy, pedro@nict.go.jp, serrat@tsc.upc.edu

***Abstract*—The traditional telecommunications business is evolving towards offering a richer set of services beyond basic connectivity, leveraging on network programmability and virtualization. A versatile execution environment is required, capable of running different workloads in different locations in the network. Cloud computing is the key paradigm that allows fostering this trending change. One interesting question to solve is to what extent those computing environments have to move towards the edge. Some services can be enabled by environments with increased capillarity, while others can be implemented in environments with more relaxed constraints (e.g., in terms of latency). This paper explores this topic by differentiating service edge from physical network edge and proposing an architecture based on the ALTO server for assisting orchestration systems in discriminating the suitable environments for each service. In addition to that, we propose a network-flow based strategy for assigning service functions to infrastructure elements following those precepts, together with an initial validation on the scalability of the assignation solution, a well-known problem of this task.**

I. Introduction

The cloud computing paradigm has provided a new model for service delivery where Data Centers (DCs) hosting a pool of Information Technology (IT) resources, are able to attend multiple service demands by means of a dynamic assignment of capabilities, such as CPU or storage capacity, either as physical or virtual resources (in the latter, by using some abstraction mechanisms). The virtualization technology in the cloud allows the flexible management of those IT resources, distributing them per service as needed (following an Infrastructure-as-a-Service, IaaS, approach) either among distinct servers into a single data center, or spreading them across several interconnected data centers, even across multiple administrative domains.

The computing resources are then allocated on-demand depending on the customer (or tenant) requests. This elasticity on resource consumption allows and encourages efficient resource utilization and an agile adaptation to the business and service needs in every moment.

Originally, those DCs were conceived as large centralized facilities concentrating a significant number of computing resources. However, new service needs (e.g., requiring low latency or benefitting from the proximity to the end-user) are influencing this design by reconsidering the need of deploying more and more computing capabilities towards the network edge. The emerging approach is to deploy multi-purpose hardware resources at the edge of a telco operator's network to dynamically deploy the application logic close to the end-user device.

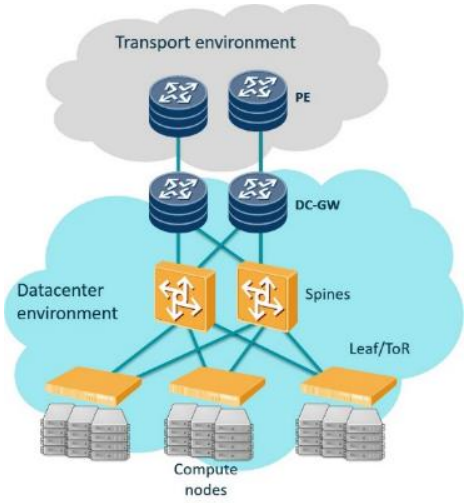

Fig. 1. Centralized cloud site architecture.

Such trend, while initially can be seen as natural, supposes larger investments as well as the introduction of adaptation mechanisms in the control operations of the network in order to offer flexibility for agile connectivity of workloads variable in time, origin, etc. It is therefore essential to understand what is the optimal edge for each of the services that will be supported by the network, thus avoiding any overinvestment and over- dimensioning of the network systems with computing execution environments and transport capacity connecting them. This is, however, not easy because of the intrinsic uncertainty of the services that will be deployed on those systems, and where (location) and when (time) they will be deployed, especially in the advent of 5G.

Moreover, even assuming a certain degree of distribution of the IT resources across the network, it is not clear what can be the criteria and procedures for identifying the most convenient location for each service at each time, pursuing efficiency in the sense of not starving precious resources for services that could be accommodated in other less important or critical infrastructures. All of this, for sure, depends on the kind of service to be deployed, since whatever can be essential for a service does not necessarily correspond with a key constraint for another one.

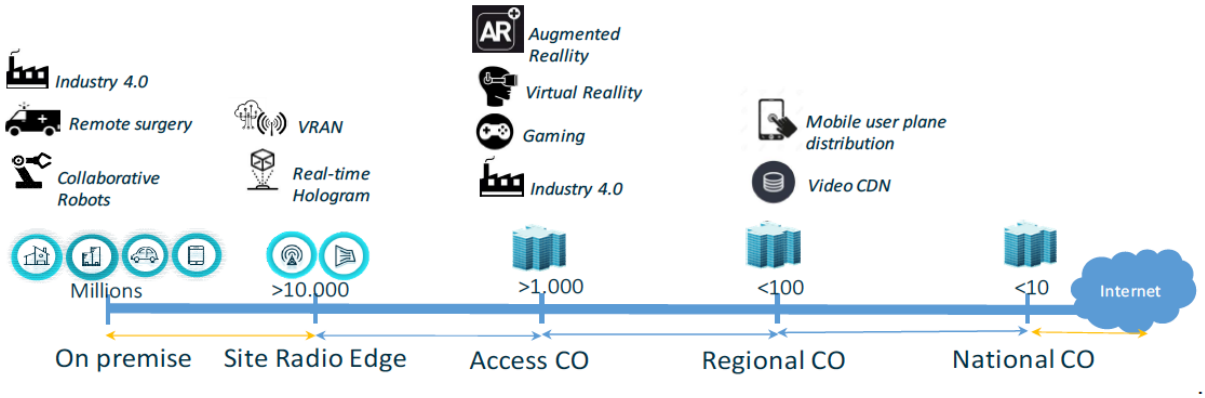

Fig. 2. Placement options depending on the type of service.

This paper proposes an initial approach to this issue by exploring what kind of parameters can be taken into account for infrastructure discrimination, as well as proposing an initial idea of integration of these decision mechanisms (based on ALTO) in a management and orchestration system governing many cloud infrastructures in the network. It introduces a combinatorial optimization process that leverages on the particularities of the domain to address the typically untreatable problem of optimally assigning services to host infrastructure in a practicable manner.

The paper is organized as follows. Section II provides insights on the evolution of existing carrier networks with the introduction of computing environments for allowing a flexible deployment of services. Section III stresses the difference between the physical network edge and the logical service edge, as the motivation for selecting appropriate edges for each type of service. Section IV presents a network flow-based combinatorial optimization approach for integrating such decision criteria on the management and orchestration system in a tractable manner. Finally, Section VI concludes the paper and presents some future steps with regards to the objective of the paper.

II. Cloud-based Telco Networks

Network operators have started to evolve their networks [1] by introducing computer-based execution environments and developing control mechanisms to manage and operate the transport equipment connecting those environments. The objective is two-fold: provisioning cloud services to their customers and enabling the virtualization of network functions. To achieve this goal, we need to define mechanisms that are able to orchestrate the cloud environments with different and heterogeneous access and core networks, dynamically controlling intra- and inter-DC connectivity enabling high throughput and low-latency services [2].

A. Evolution from conventional telecom networks

Traditional telco network architectures (for both fixed and mobile) have been typically designed as centralized with a hierarchical structure, having a clear function of aggregating traffic and offering access to content out in the Internet [3].

This traditional approach is being questioned as a result of the need of introducing and managing diverse and dissimilar services for a multitude of heterogeneous terminals connected through different access networks compelling a wide variety of QoS performance requirements, bandwidths, traffic profiles, and connectivity types. Future telco networks are expected to support the needs of a hyper-connected society, which is continuously demanding very high data rate access, independence from the technology of attachment to the network, and an increasing number of almost permanently connected devices. Conventional ways of engineering services are not valid anymore, i.e., based on monolithic devices statically located in the network. Evolution in time, location, and requirements of the workloads generated by the end-users advocates for a flexible infrastructure able to allocate resources that can be instantiated and removed, scaled-up and down, and being made closer to the user, according to the real needs of the overall services, in real-time.

The efficient integration of cloud-based services among distributed DCs, including the interconnecting network, becomes then a challenge to provide performance guarantees, localization, and high availability properties. The transport network must increase its flexibility in terms of automatic configuration and adaptive bandwidth allocation to support such services. As a mean of achieving that, it is foreseen an overall control of the resources (both for network and cloud) in an automatic fashion.

B. Network Softwarization and Virtualization

Generally speaking, these IaaS-based cloud services can be used for the deployment of virtualized network functions, respecting the requirements of telecom services such as high availability (i.e., 5-nines), very low latency, and sophisticated networking.

Network Function Virtualization (NFV) advocates for the instantiation of Network Functions (NFs) on commodity hardware, as opposed to the monolithic approach of vertical software and hardware integration, common up to recent years. The possibility of instantiating services as a composition of Virtualized NFs (VNFs) distributed along the network provides

TABLE I
SERVICE CHARACTERIZATION BY TYPE [4].

Scenario	End-to-end latency	Jitter	Traffic density
Discrete automation motion control	1 ms	$1\mu s$	$1Tbps/km^2$
Discrete automation	10 ms	$100\mu s$	$1Tbps/km^2$
Process automation – remote control	50 ms	20 ms	$100Gbps/km^2$
Process automation – monitoring	50 ms	20 ms	$10Gbps/km^2$
Electricity distribution – medium voltage	25 ms	25 ms	$10Gbps/km^2$
Electricity distribution – high voltage	5 ms	1 ms	$100Gbps/km^2$
Intelligent transport systems/infrastructure backhaul	10 ms	20 ms	$10Gbps/km^2$
Tactile interaction	0,5 ms	TBC	[Low]
Remote control	[5 ms]	TBC	[Low]

great flexibility and facilitate the adaptation to whatever specific need of a service.

One further step to take is introducing the network slicing concept, leveraging on network softwarization and virtualization. The idea behind this concept is the possibility of defining an arbitrary amount of logically independent network partitions or slices, each comprising different resources and NFs, which are interconnected and are involved in the delivery and the operation of a specific service. By instantiating network slices, the network will be able to provide completely different services in a dynamic way over the same infrastructure. Each slice will behave and appear as a fully-functional network, despite those slices actually, operate over the same physical infrastructure.

C. Cloud-based services as an enabler of network slicing

Network providers are nowadays deploying cloud-like facilities, termed NFV infrastructures (NFVI) in NFV terminology, which will serve to host VNFs. An NFVI allows the deployment of VNFs in a dynamic way that can be adapted to the specific needs of each requested service. In Telefnica, those NFVI capabilities are based on UNICA [5], the architecture being globally deployed at all its operation centers. The overarching architecture of UNICA is based on cloud concepts to allow large-scale deployment across multiple sites, but also covers provider needs such as carrier-grade performance, scalability, and operational capabilities.

For the NFVI infrastructure to be operational, it will be required to define several types of sites or points of presence along the operator's network to create the topology needed to satisfy the requirements of the services according to their needs in terms of latency, processing capacity, bandwidth, etc. The aim is to distribute workloads efficiently and smartly across the network. The objective is to distribute network loads or applications that improve the efficiency of the network and the user's service experience.

Conventional centralized IT data center architectures for cloud computing could play a role in the above context, but they are clearly insufficient. In fact, new communication trends require higher levels of capillarity in terms of edge cloud locations.

Thus, the concept of *edge* can be tackled at different points of the network [6], such as the operator's core network, the aggregation central offices, the base stations or even the on-premises or on-device. Distinct applications with different kind of workloads and different service requirements could be in principle constrained to certain of those locations to work properly. Figure 2 provides a first approximation to this idea together with some potential workload distribution. However, having an assessment method of this fact could help to select the proper execution environment for a given service and rationalize the investment needed to support plenty of computing capabilities across the network.

In summary, different environments can be further complemented with more cloud facilities in different parts of the network. With a potential plethora of computing capabilities across the network, selecting the most appropriate execution environment for each service is then an exercise of efficiency and optimization.

III. PHYSICAL EDGE VS. SERVICE EDGE

The evolutionary roadmap of existing telco networks will then offer multiple levels of processing/storage (local, edge cloud, remote, and federated cloud). The criteria to decide where to deploy the service (i.e., the NFs defining the service) must be defined by considering a combination of several factors, among which it can be mentioned the service performance parameters, the minimization of energy consumption, the network and cloud load balancing, etc. Such decisions should be transparent to the user.

Many references can be followed for understanding the different needs of future 5G services. In [4] and [7], 3GPP has defined three types of service categories, namely enhanced Mobile Broadband (eMBB), ultra Reliable Low Latency (uRLLC), and massive Machine Type Communications (mMTC), as well as the corresponding traffic requirements for two of them, the eMBB and uRLLC slices. Traffic requirements for mMTC are defined by NGMN in [8]. Finally, 5G PPP has provided several 5G use cases in [9], identifying a set of basic characteristics for them.

Table I presents some examples of the characterization of different forthcoming 5G service scenarios, in terms of latency or traffic needs, for example. It is then clear that different constraints can be relaxed or on the contrary, considered as mandatory at the time of identifying from which point in the network is most appropriate to carry out the service delivery.

For instance, for the motion control in the discrete automation case, the end-to-end latency is limited to 1 ms, requiring at the same time a high traffic density of up to 1 Tbps/km2. This suggests the need for delivering the service very close

to the physical edge of the network. On the contrary, in the remote control for process automation, both the latency and the traffic density can be relaxed up to 50 ms and 10 Gbps/km2 respectively, which in principle can be accommodated in DCs more deeply in the network, despite it could also be hosted close to the physical edge of the network, as before. This reflects the fact that the physical edge of the network does not necessarily correspond with the suitable edges for each of the services to be deployed.

Here we assume that the 5G services being deployed will basically consist of five technical dimensions:

- Bandwidth (B), characterized by indicators like data rate, accumulated data volume, etc.;
- Delay (T), articulated around parameters like latency, jitter, etc.;
- Computation (C), determined by aspects like the processing imposed by the number of sessions to be maintained, processing needs for the service, etc.;
- Storage (S), influenced by memory size, the volume of data to be stored, etc.;
- Durability (V), defined by the ephemeral duration or permanent behavior of the service to be deployed.

All these dimensions can be taken into account at the time of deciding where and when to deploy a service, or what resources and functions allocate for creating a supportive slice for that given service. Other parameters can also influence, like geographic or regulatory limitations that can condition the number of selectable edges, but these are not addressed here for the sake of brevity. Then, for each service to be deployed, it is required to identify how it maps against the referred technical dimensions, in such a way that compliant DCs can be discriminated and selected as suitable candidates for deployment.

The set of cloud infrastructures I available at the time of deploying a service are composed of many IT resources R for computation and storage in a specific location L and will be accessible through the network via some links of a given capacity A. The resources R can be checked against the service needs in terms of computation C and storage S; the capacity A can be checked against the needed bandwidth, B; and finally, the location L can be checked against the delay T, e.g. by means of monitoring data obtained through active probing between the access point of interest and the targeted cloud sites. Furthermore, the temporal availability of the resources could be restricted, for instance, due to a pre-scheduled future use because of, e.g., a calendaring schema (in this work, we do not address this issue either).

When a service is to be deployed, it can be modeled in terms of parameters B, T, C and S, and also characterized by the expectation on the durability of the service, V. Discriminating what of those cloud infrastructures can properly host the newly requested service and choosing a proper matching of services and infrastructures is known to be a computationally exigent task, known in general as the assignment problem. Bellow, we propose one solution based on classic network flow-based combinatorial optimization, that leverages the particularities of the problem to control its computational complexity. Once identified what the environments suitable for the deployment of a given service are, and its best distribution (or at least a very good one), this information can be consumed by the orchestration system ensuring that the service KPIs can be satisfied.

IV. The Service Assignment Problem

The assignment problem, is one form of the facility location problem, largely studied in works such as [10] and, as most of them, can be seen as a combination of special cases of the Knapsack Problem (KP). In the classic KP, we have an item set N, consisting of n items j (they can be the services in this paper) with profit p_j and weight w_j(a service requirement, for example), and the capacity value c (the infrastructure capacity). The objective of a KP is to select a subset of N such that the total profit of the selected items is maximized and the total weight does not exceed c. However, since the service requirements of our own problem are multidimensional, we have elements of the *d-dimensional knapsack problem* (d-KP) [11], which can be seen as a knapsack problem with a collection of different resource constraints or one constraint consisting of a multidimensional attribute. On top of that, because we have multiple infrastructure elements, we may see the problem also as a *Multiple Knapsacks Problem* (MKP) [11]. MKP is a generalization of KP from a single knapsack to m knapsacks. Its objective is to assign each item to at most one of the knapsacks such that none of the capacity constraints are violated and the total profit of the items put into knapsacks is maximized.

Following this characterization, our service assignment problem can be formulated in terms of Linear Programming (LP) as follows. We have a set of service requests $F = f_1, ..., f_n$ with profits p_j (the profit is independent of which infrastructure element host it) and the previously mentioned requirements on storage s_j, bandwidth b_j, and computation c_j, all with $j = 1, \ldots, n$. We have also a set of infrastructure elements $R = r_1, \ldots, r_m$ with positive capacities σ_i, β_i, and κ, corresponding to storage, bandwidth, and computation resources, with $i = 1, \ldots, m$. By now, we leave out other constraints that are not related with the *capacity* of the Infrastructures but with its location (e.g., delay and jitter); they will be considered as part of the heuristics applied later in Section IV-B.

We call a subset $\hat{F} \subseteq F$ feasible if the items of F can be assigned to the infrastructure elements without exceeding their capacities, i.e. if $\hat{F}$ can be partitioned into m disjoint sets F_i, such that $s(F_i) \leqslant \sigma_i$, $b(F_i) \leqslant \beta_i$, and $c(F_i) \leqslant \kappa_i$ with $i = 1, ..., m$. The objective is to select a feasible subset $\hat{F}$, such that the total profit of F is maximized.

The problem is stated as:

$$\text{maximise} \sum_{i=1}^{m} \sum_{j=i}^{n} p_j x_{ij} \tag{1}$$

$$\text{subject to} \sum_{j=1}^{n} s_j x_{ij} \leq \sigma_i, i = 1, \ldots, m, \tag{2}$$

$$\sum_{j=1}^{n} b_j x_{ij} \leq \beta_i, i = 1, \ldots, m, \tag{3}$$

$$\sum_{j=1}^{n} c_j x_{ij} \leq \kappa_i, i = 1, \ldots, m, \tag{4}$$

$$\sum_{i=1}^{m} x_{ij} \leq 1, j = 1, \ldots, n, \tag{5}$$

$$x_{ij} \in 0, 1, i = 1, \ldots, m, j = 1, \ldots, n \tag{6}$$

where variable $x_{ij} = 1$ if function j is placed infrastructure element i and zero otherwise. To guarantee that each function can be placed in an infrastructure element, we assume:

$$s_j \leq \sigma_i, b_j \leq \beta_i, c_j \leq \kappa_i, i = 1, \ldots, m, j = 1, \ldots, n \tag{7}$$

To avoid trivial constraints it is assumed that:

$$\sum_{j=1}^{n} s_j \geq \sigma_i, \sum_{j=1}^{n} b_j \geq \beta_i, \sum_{j=1}^{n} c_j \geq \kappa_i, i = 1, \ldots, m \tag{8}$$

A problem as the one formulated above is well known to be NP-hard, with no pseudo-polynomial solutions (unless $\mathcal{P} = \mathcal{NP}$). Because this, many heuristics have been developed to address complex KPs, some based on branch-and-bound strategies, other through approximation algorithms (for example, see [12], [13]). However, having in mind the particular dynamics of the problem, this is, both the available infrastructure and the demand pattern are not very diverse and change slowly, we have chosen a method similar to the one in [14] that, instead, shifts most of the computation complexity to an algorithm that builds a potentially vast flow network but only when the infrastructure or the demand pattern change.

To distribute the computational load in a way that left the heaviest parts to be performed offline and not too often, we divided the problem into two main tasks.

The first and most onerous task is to get the set of infrastructure cloud elements –with their RAL characteristics– and a demand profile of services –with their BTCSV requirements–, and compute an optimal and generic assignment of service requests to infrastructure elements. That includes addressing the combinatorial problem of considering all the possible service types combinations that an infrastructure element can host and chose the optimal one (below, we describe how we reduce the size of this problem). This part can be computed only when the infrastructure changes and using either a particular set of service requests or the expected demand profile for a particular time-frame. We call this part of the problem "Service Demand-pattern Assignment."

The second task is to get the result generic result of the "Service Demand-pattern Assignment," and a particular instance of service requests set, and assign them to each infrastructure element. We call this task, "Service Request Assignment."

Below, we present the details of how these two tasks are addressed.

A. Service Demand-pattern Assignment

A 5G infrastructure, such as the one considered in this paper, might be sized in some thousands of nodes and the number of service requests in such a network would be of a similar order. Addressing such scales naively might be unpractical even for an offline task. Thus, to reduce the size of the problem, we assume that each of infrastructure element r_j belongs to a type (such as Access CO or Site Radio Edge, as those in Table I) with common characteristics such as storage capacity, computation power, and bandwidth[1]. Hence, we have a set of different infrastructure types $\mathcal{R} = \mathcal{R}_i$, such that $|\mathcal{R}| \ll |R|$. Second, that each of the services requests f_i belongs to also a type of service (e.g., Video CDN, VRAN) with common requirements such as storage capacity, computation power, and bandwidth[2]. Consequently, we have a set of different service types $\mathcal{F} = \mathcal{F}_i$, such that $|\mathcal{F}| \ll |F|$. We assume that all services belonging to a type $\mathcal{F}_i$ imply the same profit p_i.

It is quite common to use flow networks to model combinatorial problems [15]. Network models of this type have several elements: capacities on the edges, indicating how much "flow" they can carry; source nodes, which generate traffic; sink (or destination) nodes which "absorb" flow as it arrives; and finally, the flow itself, which is transmitted across the edges. The "flow" is an abstraction for different things that depend on the problem being modeled.

Formally, a flow network is a directed graph $G = (V, E)$ with the following features: a capacity c_e and cost o_e, associated with each edge e; a source node $s \in V$; and a sink node $t \in V$. In addition to this, no edge enters the source s and no edge leaves the sink t; there is at least one edge incident to each node; and finally, all capacities are integers. An $s-t$ flow is a function w that maps each edge e to a non-negative real number, $w : E \to R^+$; the value $w(e)$ represents the amount of flow carried by edge e. A flow w must satisfy the following two properties.

$$0 \leq w(e) \leq c_e, e \in E, \tag{9}$$

$$\sum_{e \text{ into } v} w(e) = \sum_{e \text{ out of } v} w(e), v \neq s, v \neq t \tag{10}$$

The value of a flow w, denoted $v(w)$, is defined to be the amount of flow generated at the source:

$$v(w) = \sum_{e \text{ out of } s} \tag{11}$$

Finally, the main purpose of all this formulation is to, given a flow network, find a flow of maximum possible value and minimum cost.

[1]Note that two Access CO with different resource configurations or locations may belong to different types

[2]Note that, for example, different Video CDN service requests with different requirements belong to different service types in this model.

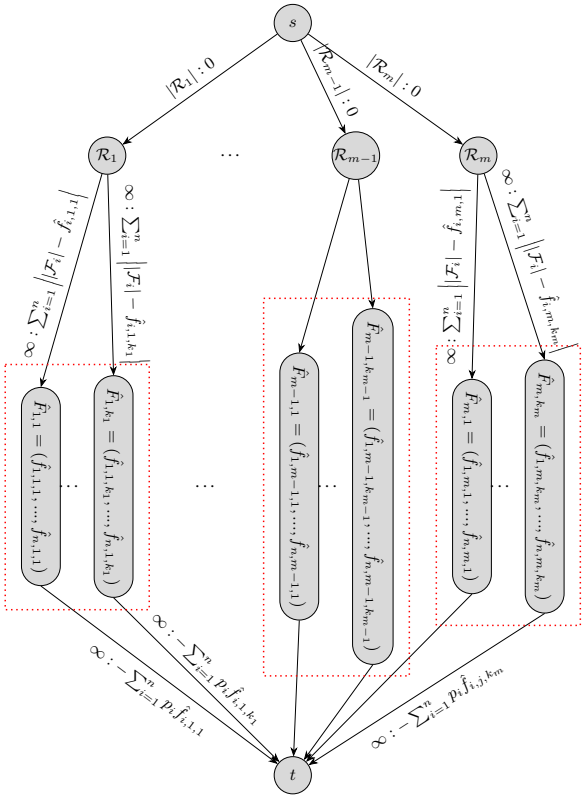

Fig. 3. Graph G_D, the Network Flow formulation of the *Demand-pattern Assignment Problem*. Each edge has a label "capacity:cost" on it although not of them are depicted for the sake of clarity. Each doted, red box represents a type of infrastructure element containing all feasible combinations of service functions it can host. Note that different infrastructure types can host different types of services.

Following this type of approach, we model the *Service Demand-pattern Assignment* as a minimum cost, maximum flow problem. To build the needed flow network, we create a graph G_D organized into four layers, like the one seen in Figure 3, which is typical of the bipartite matching problems. The top and bottom nodes are the source and the sink of the flow, respectively. The second layer from the top has the nodes $\mathcal{R}_1, \ldots, \mathcal{R}_m$ which represent the types of cloud infrastructure elements, and the edges $s - \mathcal{R}_i$ have no cost and a capacity $|\mathcal{R}_i|$ equal to the number of infrastructure elements included in the particular type. The third layer has the nodes $\hat{F}_{j,k_i}$, each of them represent a feasible assignment combination of the services, divided by service type, in an infrastructure element of type $\mathcal{R}_j$. This feasible assignment is represented by a vector $\hat{f}_{i,j,k}$ that represents the number of services of type $\mathcal{F}_j$ in k-th feasible placement at the infrastructure element of type $\mathcal{R}_j$. The capacities of all the edges between the second and the third layer are ∞ (there is no need to constrain these edges). In this way, we set the flow network to compute a bipartite matching between the infrastructure element types in the second layer and one combination of services in the third. Additionally, in order to choose the best matching, we add costs to the edges. We want to match an infrastructure type $\mathcal{R}_i$ with a combination service placements that, considering all the matchings, covers as many service requests in the pattern as possible. We express that idea as a minimization problem: the best distribution of services is the one that minimizes the distance between the vector of feasible assignations $\{\hat{f}_{i,j,k}\}$ and the vector with the total number of service functions request $\{|\mathcal{F}_i|\}$. This distance, and therefore the cost of the edges between the second and third layers, is computed as

$$o_e = \sum_{i=1}^{n} \left| |F_i| - \hat{f}_{i,j,k} \right| \tag{12}$$

and represents the cost of each edge between layers 2 and 3. Finally, as in the formulation of Equation 1, we want also to maximize the profit of our distribution of functions, thus, to each edge between nodes in Layer 3 and the sink, we set a cost

$$o_e = \sum_{i=1}^{n} p_i \hat{f}_{i,j,k} \tag{13}$$

that models the objective function in Equation 1. Although it is a simple demonstration, it is beyond the scope of this paper to prove that the minimum-cost flow of graph G_D assigns the service functions to the infrastructure elements in such a way that maximizes the profit and tries to assign as many service requests as possible on the existent infrastructure. To compute the minimum cost, maximum flow that this network can accommodate, we use the *dual-simplex* algorithm provided by Matlab®.

Most of the computational load of the procedure described above is on the building of the graph, in particular, its third layer. In the worst case, the number of nodes in this layer is bounded by $O(|\mathcal{R}|K^{|\mathcal{F}|})$, where K is the maximum number of service instances of a given type that can be placed in an infrastructure element. For example, if an infrastructure element has 96 cores, and the less exigent service needs two cores to work properly, then $K \leq 48$. Although it is a beyond a polynomial complexity, in practice, the number of service types and service instances that can be placed in an infrastructure element are bounded, and this number stays treatable (we are just using brute force to compute the third layer). Additionally, this layer has to be built only when there is a new function type or a new infrastructure type.

1) Experiments on the Service Demand-pattern Assignment: In order to show the feasibility of this approach, we depict some performance results of our implementation for the *Service Demand-pattern Assignment* problem in Table II. There, we show the running time for building graph G_D (G_D Build Time column) and for solving the Service Demand-pattern Assignment problem (SDPAP Processing Time column). The experiments where ran on an Intel® Core™ i7-8550U CPU, at 1.80GHz, with 16GB of RAM.

It can be seen that most of the time is used to compute the graph. It worth noting that graph G_D will only have to be built after a topology change or a new type of service is created. A change in the pattern of service demand only affects the costs of the edges (which can be updated in lineal time) and requires a new execution of the minimum cost algorithm, which can be run in some seconds even for large graphs, as can be seen in the table.

TABLE II
SOME EXPERIMENTS TO SHOW THE FEASIBILITY OF THE SERVICE DEMAND-PATTERN ASSIGNMENT PROBLEM.

Infr. Types	**Infr. elements**	**Service Types**	**Service requests**	G_D **Size (nodes)**	G_D **Build Time (s)**	**SDPAP Processing Time (s)**
5	250	2	200	487	0.37	0.01
5	250	3	300	1082	3.57	1.59
5	250	4	400	2512	90.77	4.97
5	250	5	500	5492	717.86	9.56
5	250	6	600	10542	3049.61	19.09

B. Service Request Assignment

The second task, the *Service Request Assignment*, is simpler. It takes the assignation made by the previous task, an actual set of service requests, and assigns them to infrastructure nodes. To perform this second assignation, we use the same network flow technique, but this time in its simpler version, which only tries to maximize the flow on a network without edge's costs. This is typically solved the Ford-Fulkerson [15] algorithm; when the capacities of the edges are integers, the run-time of this algorithm is bounded by $O(Ef)$, where E is the number of edges in the graph and f is the maximum flow in the graph. This low complexity makes the algorithm very useful to assign functions to infrastructure elements in run-time.

In this case, the flow network is the graph $G_S = E_S, V_S$ with the structure depicted in Figure 4. As with G_D, top and bottom nodes are the source s and the sink t of the flow; this time, the second layer from the top represents the set of service requests F with nodes f_i, and the edges from s to each node f_i have all unitary capacities. The third layer is computed starting from the nodes $\hat{F}_{i,j}$ selected by the *Service Demand-pattern Assignment* task. They are replicated as many times as flow enters in the same node in the solution to the *Demand-pattern Assignment* problem above, following the different sources of flow computed. In this way, if, for example, an edge $(\mathcal{R}_i, \hat{F}_{j,k})$ has a flow $w_{(\mathcal{R}_i,\hat{F}_{j,k})} = 2$, the third layer has two nodes named $\hat{F}_{j,k,I_1}$ and $\hat{F}_{j,k,r_2}$, where r_1 and r_2 are the identifiers of two infrastructure elements of type $\mathcal{R}_i$ chosen randomly among the set R_i. This is made in lineal time.

The edges $(f_i, \hat{F}_{i,j,k})$ connecting these two main layers also have unitary capacities. This is a typical bipartite matching graph, and computing the maximum flow in it is equivalent to assign a service to a particular infrastructure node.

In is important to note that edges $(f_i, \hat{F}_{i,j})$ have to be created following two restrictions:

- $(f_i, \hat{F}_{i,j,r_k}) \in E_S$ only if placing service f_i in the node r_k, complies with the delay and density requirements of f_i.
- $\hat{f}_{i,j,k} \geq \sum_e c_e : e = (f_x, \hat{F}_{i,j,r_k}), f_x \in \mathcal{F}_i$.

The first restriction re-introduces in our solution the service constraints that were left outside the initial linear programming problem formulation. The second restriction is enforced just removing incoming edges randomly until the restriction holds.

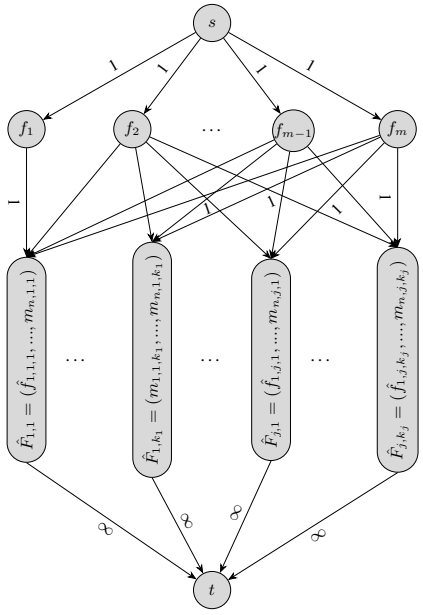

Fig. 4. Network Flow formulation of the *request assignment problem*. (not all capacities are shown for clarity)

C. Incrementally Updating the Set of Services

All the process described above works allocating a complete set of service requests at the same time (e.g., all the services to be deployed simultaneously during a period of time), but, in a more realistic scenario, new service requests arrive continuously to, for example, the *ALTO Server* that will be described below. It would be service disruptive to re-assign and re-allocate all the services every time a new request arrives, however, the method presented here permits to assign a new service to a hosting infrastructure just recomputing the flow in the second graph after adding a new node in the second layer. Ford-Fulkerson permits to do that in a time that is bounded by $O(m + n)$, where m is the number of edges and n the number of vertex of the graph. The same idea and complexity are valid for removing a service. For this, we are assuming that adding or removing a service, which should change the costs of G_D edges, do not impact significatively in the output of the *Service Demand-pattern Assignment*. We leave for future work a study on how this strategy impacts the optimal use of the infrastructure resources.

V. SERVICE EDGE VIEW BASED ON ALTO

This section describes a potential architecture based on ALTO concept [16], where the identification of the most appropriate edge execution environments, as proposed above, could be offered as an ALTO service. The idea is to relay on ALTO for retrieving the recommended edges for a given 5G service to be deployed in the network after running either both assignments process presented in Section IV or just the lighter *Service Request Assignment* process, depending on the dynamics of the demand and the network.

The proposed architecture is illustrated in Figure 5. It assumes that an Orchestrator, in charge of deploying a 5G services requiring to instantiate some capabilities at the edge

interfaces with an ALTO server in order to retrieve an indication of the computing environments that could satisfy the final service requirements. For doing so, the ALTO server interacts with a number of Edge Managers responsible for managing the compute and storage infrastructure of each of those execution environments spread across the network. These Edge Managers will be responsible mainly of providing information about the resource availability in each edge node under its control, and at the same time reporting sufficient information for helping to identify the topological location of the edge node itself. The resource-based information can be used to feed an Edge Resource database, assisting in the identification of resources available in each node. In order to allow real-time status collection of the resources in each edge node, the system can rely on telemetry information [17] provided by a monitoring system attached to the underlying infrastructure, interacting with the ALTO server either directly or through the particular Edge Managers. On the other hand, the topology information could be merged or integrated with the ALTO network maps, in order to create an overall topological view of the network and computing capabilities internal to the service provider. This topological view can be relevant to decide the placement of service components, for example, in the form of Virtual Network Functions (VNFs) or applications, that could show some restrictions (e.g., latency).

The ALTO server will provide a set of convenient edges for the specific service as requested by the Orchestrator. To do so, will run the *Service Demand-pattern Assignment* process presented in Section IV-A1 to obtain a set of suggested infrastructures (in general only the light assignation part) and, if needed, the much lighter *Service Request Assignment* process presented in Section IV-B. Notably, the application of some policies can also be foreseen, e.g., through specific modules, in order to assist the edge discrimination, for instance, policies for data sovereignty. Thus, this augmented ALTO server could result integrated into orchestration frameworks for edge computing. For instance, considering the ETSI Multi-access Edge Computing architecture [18], the Orchestrator in Figure 5 could be the Multi-access Edge Orchestrator at MEC system level, while the Edge Managers represented in the figure could be the per-host Virtual Infrastructure Managers in MEC.

It is important to note that ALTO will basically assist in the identification of the best execution environments for certain services, but it will not participate in the resource allocation, which will be the responsibility of the Edge Managers once the Orchestrator, based on ALTO, selects some edge node. Then, after resource allocation is performed for a given service, whenever any parameter is unmet, the virtual system is adapted as soon as possible to ensure the continuity of the service. A reactive or proactive method can be used, as discussed in [19].

VI. Concluding Remarks

One of the motivations for the concept of network slicing is to deploy services at the edge of the network for reducing latency or providing a certain service in proximity to the consumers of that service, e.g., content. With the different computing capabilities offered by infrastructure providers, it is essential to get informed decisions about where a service can be deployed to satisfy its specific requirements, being static or dynamic. In this way, having mechanisms for identifying suitable service edges can facilitate the work of the service orchestration by directly selecting the cloud execution environments from those that can match the service expectation.

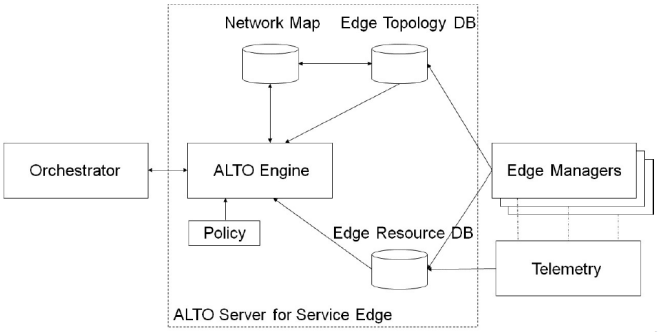

Fig. 5. Integration with the orchestration system.

Such a fact becomes clear when analyzing the characteristics of future 5G services. The particular characterization of the services into categories, such as eMBB or uRLLC, remarks that some parameters can be relaxed or can be seen as less critical from one service respect to the other, even falling on the same service type, an aspect exploited in this work. Some other studies have also revealed this fact when dealing for instance, with the QoS capabilities to be offered by the network [20].

This paper explores the convenience of assisting the decision of the orchestration systems. Such a decision is typically taken on the basis of a number of key parameters, such as bandwidth and latency, and large numbers of options resulting in intractable problems. We take advantage of the actual scale of the problem, the possibility of grouping services and infrastructures into types, and the dynamics of the topology and demand of the network, to propose a feasible solution of the typically intractable multi-dimensional assignment problem.

We have left out of the scope of this paper some formal aspects such as proving that the costs in the edges of graph G_d actually induce an optimal fit, although is intuitive to see that it maximises the profit. The same happened with the impact of incrementally assigning services to infrastructure, without disrupting those already deployed, but in base to slightly inaccurate models. Properly addressing those issues would not fit in this paper and is left to be presented in further work.

Acknowledgment

This work has been partly funded by the European Commission through the projects NECOS (Grant Agreement no. 777067) and 5G-TRANSFORMER (Grant Agreement no. 761536).

References

[1] L. M. Contreras, V. Lopez, O. G. De Dios, A. Tovar, F. Munoz, A. Azanon, J. P. Fernandez-Palacios, and J. Folgueira, "Toward cloud-ready transport networks," *IEEE Communications Magazine*,

vol. 50, no. 9, pp. 48–55, sep 2012. [Online]. Available: http://ieeexplore.ieee.org/document/6245958/

[2] L. Velasco, L. M. Contreras, G. Ferraris, A. Stavdas, F. Cugini, M. Wiegand, and J. P. Fernandez-Palacios, "A service-oriented hybrid access network and clouds architecture," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 159–165, apr 2015. [Online]. Available: http://ieeexplore.ieee.org/document/7081090/

[3] R. D. Doverspike, K. K. Ramakrishnan, and C. Chase, "Structural Overview of ISP Networks." Springer, London, 2010, pp. 19–93. [Online]. Available: http://link.springer.com/10.1007/978-1-84882-828-5{_}2

[4] 3GPP, "TS 122 261 - V15.5.0 - 5G; Service requirements for next generation new services and markets (3GPP TS 22.261 version 15.5.0 Release 15)," Tech. Rep., 2018. [Online]. Available: https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

[5] D. Cooperson and C. Chappell, "Telefónica's UNICA architecture strategy for network virtualisation," Tech. Rep., 2017. [Online]. Available: https://www.telefonica.com/documents/737979/140082548/Telefonica{_}Virtualisation{_}gCTO{_}FINAL.PDF/426a4b9d-6357-741f-9678-0f16dccf0e16?version=1.0

[6] Telefonica, "Telefónica Open Access and Edge Computing," no. February, 2019. [Online]. Available: https://www.telefonica.com/documents/737979/144981357/whitepaper-telefonica-opa-mec-feb-2019.pdf/b011b66e-982f-6163-9409-c3c9fddc6c89

[7] 3GPP, "TS 123 501 - V15.2.0 - 5G; System Architecture for the 5G System (3GPP TS 23.501 version 15.2.0 Release 15)," Tech. Rep., 2018. [Online]. Available: https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

[8] NGMN, "5G White Paper," 2015. [Online]. Available: https://www.ngmn.org/5g-white-paper/5g-white-paper.html

[9] Michał Maternia and Salah Eddine El Ayoubi, "5G PPP use cases and performance evaluation models," 5G PPP, Tech. Rep. [Online]. Available: http://www.5g-ppp.eu/

[10] Z. Drezner and H. W. Hamacher, *Facility location: applications and theory*. Springer, 2002. [Online]. Available: http://www.amazon.co.uk/Facility-Location-Applications-Zvi-Drezner/dp/3540213457

[11] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. [Online]. Available: http://link.springer.com/10.1007/978-3-540-24777-7

[12] A. Neebe and D. Dannenbring, "Algorithms for a specialized segregated storage problem," *University of North Carolina*, pp. 77–5, 1977.

[13] M. S. Hung and J. C. Fisk, "An algorithm for 0-1 multiple-knapsack problems," *Naval Research Logistics Quarterly*, vol. 25, no. 3, pp. 571–579, 1978. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800250316

[14] Y. Rochman, H. Levy, and E. Brosh, "Resource placement and assignment in distributed network topologies," in *2013 Proceedings IEEE INFOCOM*, April 2013, pp. 1914–1922.

[15] D. R. Ford and D. R. Fulkerson, *Flows in Networks*. Princeton, NJ, USA: Princeton University Press, 2010.

[16] S. Kiesel, W. Roome, R. Woundy, S. Previdi, S. Shalunov, R. Alimi, R. Penno, and Y. R. Yang, "Application-Layer Traffic Optimization (ALTO) Protocol," RFC 7285, sep 2014. [Online]. Available: https://rfc-editor.org/rfc/rfc7285.txt

[17] H. Song, Z. Li, P. Martinez-Julia, L. Ciavaglia, and A. Wang, "Network Telemetry Framework," Internet Engineering Task Force, Internet-Draft draft-opsawg-ntf-00, mar 2019. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-opsawg-ntf-00

[18] ETSI GS MEC 003, "Multi-access Edge Computing (MEC); Framework and Reference Architecture," ETSI, Tech. Rep., 2019. [Online]. Available: https://www.etsi.org/deliver/etsi{_}gs/MEC/001{_}099/003/02.01.01{_}60/gs{_}MEC003v020101p.pdf

[19] P. Martinez-Julia, V. P. Kafle, and H. Harai, "Achieving the autonomic adaptation of resources in virtualized network environments," in *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*. IEEE, mar 2017, pp. 52–59. [Online]. Available: http://ieeexplore.ieee.org/document/7899249/

[20] L. Cominardi, L. M. Contreras, C. J. Bcrnardos, and I. Berberana, "Understanding QoS Applicability in 5G Transport Networks," in *2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. IEEE, jun 2018, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/8436847/