Dimensioning resources of network slices for energy-performance trade-off

Wei Huang, Andrea Araldo, Hind Castel-Taleb, Badii Jouaber

Dimensioning resources of Network Slices for energy-performance trade-off

Wei Huang, Andrea Araldo, Hind Castel-Taleb, Badii Jouaber
Telecom SudParis, SAMOVAR, IP-Paris
{first_name}.{last_name}@telecom-sudparis.eu

***Abstract*—Within network slicing, Virtual Network Embedding has been vastly studied, i.e., deciding in which physical nodes and links to place virtual functions and links. However, the performance of slices does not only depend on *where* virtual functions and links are placed, but also on *how much resources* they can use, which has been mostly neglected in the literature.**

We thus propose a method for optimal resource dimensioning, via dimensioning capacities of multiple Jackson networks (one per slice) co-existing in the same resource-constrained network. Despite the long history of Jackson networks, we are the first, to the best of our knowledge, to model such a problem. The objective is to minimize energy consumption while satisfying the latency requirements of heterogeneous service providers. We show numerically that our solution is able to achieve both goals, differently from classic approaches, which assume that the amount of resources assigned to slices is fixed a-priori.

***Index Terms*—Network Slicing, Resource allocation, Jackson Networks, Optimization.**

I. Introduction

A network can now do much more than moving bytes back and forth: a single network node has the computation capabilities to perform diverse advanced functions, far beyond classic switching and routing. On the other hand, Service Providers (SPs) offering different types of applications (i.e., video streaming or, in the near future, augmented reality, autonomous driving, etc.) may be interested in exploiting the huge network capabilities to run a part of their computation *in-network*, achieving latency and traffic reduction. As an example, Netflix has signed agreements with some network operators to install in their access network dedicated hardware servers (Open Connect Appliances [1]). Obviously, installing hardware on physical networks is very costly and difficult to maintain and absolutely infeasible if hundreds of SPs have to be accommodated. Moreover, very low latency services (e.g., autonomous driving) require the computation to run very close to users, in edge nodes (e.g., base stations), in which it is impossible to install physical servers.

Network slicing is a technique that will enable SPs to run part of their computation in-network, thus exploiting the big computation capabilities potentially owned by network operators. With network slicing, a network operator virtualizes physical resources (e.g., CPU cycles in physical nodes and bandwidth in physical links) and provides a slice (a subset of such virtualized resources) to each SP. We assume SPs are organized under the microservice philosophy [2]: the logic of the SP is split in different software *components*), each performing a basic virtual function. Such components are interconnected via virtual links, each mapped to a physical path. Virtual Network Embedding (VNE) consists in deciding in which physical node we should place components and in which physical path we should place virtual links. This problem has vastly been studied in recent literature [3]–[6]. Papers in VNE typically assume that each component and virtual link "needs" a pre-defined amount of resources, declared by the SP.

However, we believe the assumption above is too strong. Indeed, a component can work under different amount of allocated computation capacity: if such capacity is large, computation will be faster. The same reasoning holds for virtual links. Therefore, how and who would fix a-priori the amount of computation and bandwidth "needed" by components and virtual links is unclear. Moreover, it is too restrictive to impose each component and virtual link to occupy always the same amount of resources. Such amount could instead be computed based on the current occupation of network. If many slices are overloading the network, it could be necessary to allocate to each slice less resources than when the network is underloaded, provided that latency constraints are not violated. Deciding the amount of resources is even more difficult if we consider that a single component can be replicated over multiple physical nodes and receive a different input request rates: the amount of CPU cycles to be assigned to each replica may thus vary accordingly. The same applies to virtual links.

Resource dimensioning has been overlooked in the literature on network slicing, under the strong assumption discussed above. The contribution of this paper is

- We study resource dimensioning in the context of network slicing, formalizing the problem as dimensioning the capacity of multiple Jackson networks, each corresponding to a slice, co-existing in the same physical network and subject to capacity constraints of physical nodes and links. To the best of our knowledge, we are the first to solve such a capacity dimensioning problem. Slices are heterogeneous, i.e., they have different latency constraints (which are constraints of the problem) and incoming request rates. The goal of the network operator is to minimize overall power consumption.

This work was partially funded by Beyond5G, a project of the French Government's recovery plan "France Relance".

978-1-6654-9792-3/22/$31.00 ©2022 IEEE

- In the numerical results we show that our optimal dimensioning can satisfy both (i) slice latency requirements and (ii) power minimization, whether classic approaches, which assume some pre-fixed amount of resources to each slice, fail in satisfying either (i) or (ii). Our code is open-source.[1]

II. Related work

In mobile networks, the radio channel is obviously a precious resource, whence the large work on Radio Access Network (RAN) sharing [7]–[9], which is orthogonal to our study: we focus on allocation of resources upstream of RAN: CPU in network nodes and bandwidth in the physical links between them.

Much work also exists on virtual network embedding, i.e., deciding where to place virtual nodes and links onto a physical network [3]–[6], [10]. In particular, [10] also decides where to route user requests and how to replicate components into multiple instances. They assume a reinforcement learning agent seats at each physical node and makes its own decisions. User requests can be seen as "hot potatoes" and each agent deals with them if the correspondent node has enough resources, otherwise sends them to neighboring nodes. In [11], SPs declare different possible configuration options (set of components) and let the Network Operator (NO) choose one, based on the load of the system. However, all the above work assumes each component and virtual link consumes an inelastic amount of resources, fixed a-priori and taken as exogenous input. In our paper instead, we do dimensioning resources of network slices, i.e., we decide such amount of resources, taking into account the impact on latency and energy consumption.

We model each slice as a stochastic network, in particular a Jackson network, composed of different queues, each corresponding to a virtual function or virtual link. Our goal is to dimension the capacity of such queues. Stochastic network dimensioning has been studied in [12]–[14].

However, all of them study one single stochastic network. Our network slicing context, instead, make our problem different: we model several stochastic networks, each corresponding to a slice, which all co-exist in the same physical network. They assume a capacity "budget" that should be distributed among the queues of their single stochastic network. They aim to minimize latency, so they use as much capacity as possible. Our constraint is not given by a budget per stochastic network: we instead have multiple capacity constraints, one per each physical node: if several slices have components there, the resources allocated to such components must not exceed the physical capacity. Similar reasoning applies to physical links.

In [15], Virtual network function (VNF) chains placement problems are also formulated as a variant of bin-packing problem and request scheduling problems are modeled based on the key concepts from open Jackson network. In [16], the authors model the Latency-aware Edge-Core Service Function Chains (SFCs) Migration problems based on open Jackson network and have proposed an algorithms to optimize the average latency of all SFCs in edge-core networks. The proposed approach consist in selecting in the first step the appropriate SFCs, VNFs and target servers for migration. Then, a better performance can be achieved due to the adjustment based on average resources utilization. In [17], a distributed dynamic allocation policy has been proposed, which offers strong performance guarantees in an adversarial setting. This algorithm compares and instantiates inference models through different locations of the network to maximize the total gain in their framework. In [18], delay constraints based on both link capacities and total flow are assumed to be satisfied as well as the capacity constraints. A convex relaxation of this NP-complete problems are given, along with some methods to obtain the upper and lower bound. In [19], instead of tackling the SFC embedding stage while taking the composition step as an assumption, the authors focused on the SFC composition problem and proposed an Integer Linear Programming (ILP) based approach to optimally solve it.

There are different service function chaining strategies : static and dynamic [20]. The dynamic service chaining model offers several advantages to operators, such as better utilization of network and compute resources, with a greater flexibility for end-to-end service provisioning. In terms of the resource allocation, in [15]–[19], a fixed amount of resources are allocated to each VNF based on its demand. We instead focus on the optimization of the amount of resources to give to each slice components and virtual links. In order to do so, it is important to correctly model the dependence between the allocated resources and the resulting request journey time. For this reason, we resort to queuing theory.

III. Model

We denote the underlying physical network as directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the sets of nodes, $\mathcal{E}$ the set of edges. Each node $v \in \mathcal{V}$ has resource R_v, which in our case is CPU cycles. We denote with $(v, v') \in \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ a link and with $R_{v,v'} \geq 0$ its bandwidth capacity. Note that $R_{v,v'}$ may be different than $R_{v',v}$. We assume that the physical network $\mathcal{G}$ is owned by a NO and that a set $\mathcal{S}$ of SPs are concurrently deployed in $\mathcal{G}$, each in a separate slice.

A. Model of a Service Provider

A SP s receives Poissonian user-requests from any node $v \in \mathcal{V}$ at rate $\lambda^s_{v,in} \geq 0$. The nodes for which $\lambda^s_{v,in} > 0$ are *ingress nodes* for s. We denote with $\mathcal{V}^s_{in} = \{v \in \mathcal{V} | \lambda^s_{v,in} > 0\}$ the set of such nodes.

SP s is deployed as a *chain* of n^s *software components* $\{c^s_i\}_{i=1,\dots,n^s}$, each characterized by a *computational complexity* $\alpha_{c^s_i}$, i.e., the amount of machine-level instructions to be executed at every request. Component c^s_i sends data to c^s_{i+1}. The *communication complexity* $\beta_{c^s_i, c^s_{i+1}}$ is the amount of data (in bits) sent from c^s_i to c^s_{i+1} at each request.

We also introduce β_{0,c^s_1} the amount of data coming directly from users and directing to the first processing component. Note that a component may be replicated in several nodes.

[1] https://github.com/Free-Wei/SliceDimensioning.git

Notation	Description
VNE	Virtual Network Embedding (Sec.I)
SPs	Service Providers (Sec.I)
RAN	Radio Access Network (Sec.II)
NO	Network Operator (Sec.II)
SLA	Service Level Agreement (Sec.III-A)
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Physical network with nodes $\mathcal{V}$ and links $\mathcal{E}$ (Sec.III)
R_v	Computational resources of node v (Sec.III)
$R_{v,v'}$	Bandwidth capacity of link $(v, v') \in \mathcal{E}$ (Sec.III)
$\mathcal{V}^s_{in}$	Set of ingress nodes for service provider s (Sec.III-A)
c^s_i	The i-th software component of a SP s (Sec.III-A)
$\alpha_{c^s_i}$	Computational complexity for component c^s_i in instructions (Sec.III-A)
$\beta_{c^s_i, c^s_{i+1}}$	Communication complexity between component c^s_i and component c^s_{i+1} in bits (Sec.III-A)
$c^s_{i,v}$	Replica of component c^s_i deployed in node v (Sec.III-A)
$\lambda^s_{v,in}$	Data rate of user-requests for SP s ingressing from each node v (Sec.III-A)
D^s	Time threshold for requests of SP s (Sec.III-A)
$r^s_{c_i,v}$	CPU allocated by NO to component replica $c^s_{i,v}$ (Sec.III-B)
$r^s_{i,v,i+1,v'}$	bandwidth allocated by the NO for the communication between component replicas $c^s_{i,v}$ and $c^s_{i+1,v'}$ (Sec.III-B)
r_q	Allocated resources to queue q (Sec.III-C)
$r_{u,u'}$	Total allocated bandwidth in physical link (u,u') (11)
$\mathcal{P}_{v,v'}$	Path from physical node v to v' (Sec.III-B)
$\mu_q(r_q)$	Service rate of queue q (Sec.III-C)
λ_q	Arrival rate of queue q (Sec.III-C)
$\mathcal{Q}_v$	Set of queues representing component replicas placed in node v (Sec.III-C)
$\mathcal{Q}_{u,u'}$	Set of queues representing virtual links mapped to physical link $(u, u') \in \mathcal{E}$ (Sec.III-C)
$\mathcal{Q}^s$	The set of queues for SP s (Sec. III-C)
$T^s\left(\{r_q\}_{q\in\mathcal{Q}^s}\right)$	Sojourn time of a request of SP s (Sec.III-C)
$P_{q,v}(r_q)$	Power consumption at node v for queue q (Eqn.4)
$P_{u,u'}(r_{u,u'})$	Power consumption on link (u,u') (Eqn.5)

Table I: Table of Notation

We denote by $c^s_{i,v}$ the replica of component c^s_i deployed in node $v \in \mathcal{V}$. We assume a deterministic fixed routing: for each replica $c^s_{i,v}$ a pre-defined routing function will return what is the replica of the next component which should receive the request, i.e., $c_{i+1,v'}$. Moreover, we assume that the path $\mathcal{P}_{v,v'}$ among any pair of physical nodes v, v' is given a-priori. Therefore, suppose a user request for SP s enters the network from an ingress node $v \in \mathcal{V}$. The routing function decides to which replica of c^s_1 this request will be forwarded. Suppose such a replica seats in node v'. The request goes from v to v' crossing the physical links specified in the pre-defined path $\mathcal{P}_{v,v'}$. The data transmitted (in bits) for this request in each of these links is : $\beta_{c^s_0, c^s_1}$.

The request is processed by the replica $c^s_{1,v'}$ and the output is sent to a replica of c^s_2, selected by the routing function, as explained before. This is repeated until the request reaches a replica of the last component $c^s_{n^s}$. We denote with T^s the mean *journey time*, i.e., the time in which the process just described is executed. Each SP s requires T^s to be less than a certain threshold $D^s > 0$. A SP s can be completely described by a

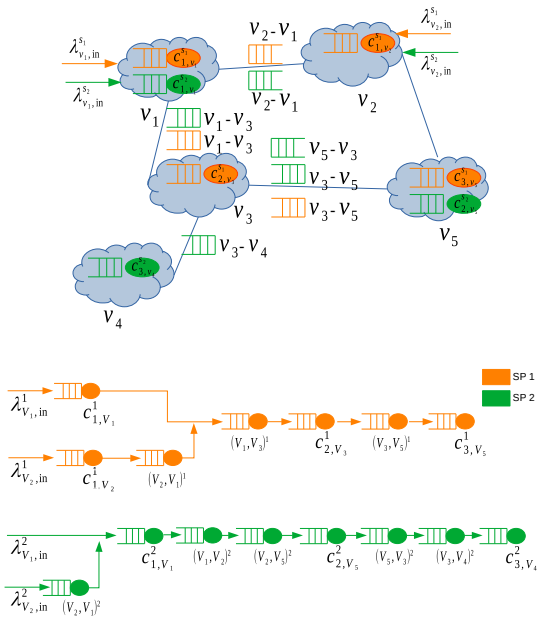


Figure 1: SP 1 and 2 modeled as Jackson networks

Service Level Agreement (SLA) in the following form:

$$s = \left(\{c^s_i\}_{i\le n^s}, \{\alpha^s_i\}_{i\le n^s}, \{\beta^s_{i,i+1}\}_{i<n^s}, \{\lambda^s_v\}_{v\in\mathcal{V}}, D^s\right) \quad (1)$$

Note that we are assuming that, for each component c^s_i its set of replicas $\{c^s_{i,v}\}$ in a subset of nodes $v \in \mathcal{V}$ is already defined. This is an input to our problem.

In the example of Fig. 1, SP s_1 has three components: component $c^{s_1}_1$ which has two replicas $c^{s_1}_{1,v_1}$ and $c^{s_1}_{1,v_2}$, in the physical nodes v_1 and v_2 respectively. The other components $c^{s_1}_2$ and $c^{s_1}_3$ have one replica each, deployed in nodes v_3 and v_5. The ingress nodes of s_1 are v_1 and v_2. The requests coming from v_1 are routed along v_1, v_3, v_5, while those coming from v_2 are routed along v_2, v_1, v_3, v_5.

B. Decision variables

For any SP s, we define $r^s_{i,v}$ as the CPU allocated by the NO to component replica $c^s_{i,v}$, and $r^s_{i,v,i+1,v'}$ as the bandwidth allocated by the NO for the communication between component replicas $c^s_{i,v}$ and $c^s_{i+1,v'}$. Observe that in this case, $r^s_{i,v,i+1,v'}$ is reserved in each physical link (u, u') belonging to path $\mathcal{P}_{v,v'}$. If v is an ingress node, $r^s_{0,v,1,v'}$ is the bandwidth reserved to transmit the requests entering v to component replica $c^s_{1,v'}$. By convention, we set $r^s_{i,v} = 0$ if there is no replica of component c^s_i in node v. Similarly, for $i > 0$, $r_{i,v,i+1,v_{i+1}} = 0$ if, according to pre-fixed routing, there is no direct communication between replicas $c^s_{i,v}$ and $c^s_{i+1,v_{i+1}}$. Similarly, $r_{0,v,1,v_1} = 0$ if the pre-fixed routing does not route user requests entering v toward component replica c^s_{1,v_1}. Observe that in this way we satisfy the *isolation* requirement of slicing: each SP has reserved CPU and bandwidth for its components replicas and communication between them, as if

it was running alone on a physical infrastructure having those resources. In other words, each SP sees a virtual node with capacity $r^s_{i,v}$, where component replica $c^s_{i,v}$ runs and sees a virtual channel of capacity $r_{i,v,i+1,v'}$ in each physical link (u,u') along the path $\mathcal{P}_{v,v'}$, used to send data from $c^s_{i,v}$ to $c^s_{i+1,v'}$. We therefore assume no interference between slices.

C. Jackson network formulation

We model each component replica $c^s_{i,v}$ as a M/M/1 queue q, whose service time is distributed exponentially with rate $\frac{r^s_{i,v}}{\alpha_{c^s_i}}$ We denote with $\mathcal{Q}_v$ the set of queues representing component replicas placed in node v, no matter the SP they belong to.

Similarly, each physical link (u,u') belonging to the path $\mathcal{P}(v,v')$, reserved to the communication between replicas $c^s_{i,v}$ and $c^s_{i+1,v'}$, has a bandwidth $r^s_{i,v,i+1,v'}$. They can be modelled as a M/M/1 queue q with the service rate $\frac{r^s_{i,v,i+1,v'}}{\beta^s_{i,v,i+1,v'}}$. We denote with $\mathcal{Q}_{u,u'}$ the set of queues representing virtual links mapped to physical link $(u,u') \in \mathcal{E}$, no matter the pair of component replica connected by these virtual links and no matter the SP they belong to.

In the example of Fig. 1, $\mathcal{Q}_{v_1} = \{q_1, q_2\}$ where q_1 and q_2 represent replicas $c^{s_1}_{1,v_1}$ and $c^{s_2}_{1,v_1}$, respectively. Moreover, $\mathcal{Q}_{v_1,v_2} = \{q', q''\}$, where q' represents the channel, reserved into physical link (v_1,v_3), for the virtual link between $c^{s_1}_{1,v_1}$ and $c^{s_1}_{2,v_3}$ and q'' represents the channel, reserved into the same physical link, for the virtual link between $c^{s_2}_{1,v_1}$ and $c^{s_2}_{2,v_5}$ (note that such virtual link needs a reserved channel in physical links (v_1,v_3) and (v_3,v_5)).

In general, for a queue q we can write the service rate as the following function $\mu_q(r_q) = \frac{r_q}{\alpha_q}$ where: (i) $r_q = r^s_{i,v}$ and $\alpha_q = \alpha^s_{c_i,v}$ if q is representing component replica $c^s_{i,v}$ or (ii) $r_q = r^s_{i,v,i+1,v'}$ and $\alpha_q = \beta^s_{i,v,i+1,v'}$ if q represents the virtual channel reserved for the communication between $c^s_{i,v}$ and $c^s_{i+1,v'}$ in any link of the path between (v,v').

Let us consider again the example in Fig. 1. Under the queuing model just described, a request for SP $s=2$ entering ingress node v_2, first joins the queue representing the channel reserved to communication between node v_2 and c^2_{1,v_1}. After being processed there, the request needs to go to c^2_{2,v_5}. To do so, it first joins the queue representing the virtual channel reserved to SP 2 in the link v_1, v_3 and then the queue representing the virtual channel reserved to SP 2 in the link (v_3,v_5). It then joins the queue representing c^2_{2,v_5} etc.

We thus represent any SP s as a network of queues $\mathcal{Q}^s$. Observe that several networks of queues co-exist in the same physical network. The networks of queues corresponding to the two SPs are represented in Fig. 1. If a SP has two or more chains of components, we will consider each as a separate SP.

Let us consider any queue $q \in \mathcal{Q}^s$ of SP s and represent routing decisions with an indicator function $\mathbb{1}_{q',q}$, which is 1 if requests exiting queue q' are directed to queue q. Similarly, the indicator function $\mathbb{1}_{v,q} = 1$ if v is an ingress node and the user requests entering v from the outside are directed toward q. The request rate entering q is then

$$\lambda_q = \sum_{q' \in \mathcal{Q}^s} \lambda_{q'} \cdot \mathbb{1}_{q',q} + \sum_{v \in \mathcal{V}} \lambda^s_{v,in} \cdot \mathbb{1}_{v,q} \tag{2}$$

Referring to Fig. 1 and SP $s=2$ and denoting with q the component c^s_{1,v_1}, then the formula above becomes $\lambda_q = \lambda_{q'} + \lambda^s_{v_1,in}$, where q' is the queue corresponding to the virtual channel hosted in link (v_2,v_1) reserved to the communication between the ingress node v_2 and c^2_{1,v_1}.

In order for the network of queues representing SP s to be a Jackson network, we need to enforce stability conditions in each queue, i.e., $\mu_q(r_q) > \lambda_q, \forall q \in \mathcal{Q}^s$. Under such conditions, the sojourn time of a request of SP s is distributed exponentially with mean, according to Eqn (476) in [21]

$$T^s\left(\{r_q\}_{q \in \mathcal{Q}^s}\right) = \frac{1}{\lambda^s} \cdot \sum_{q \in \mathcal{Q}^s} \frac{\lambda_q}{\mu_q(r_q) - \lambda_q} \tag{3}$$

where $\lambda^s = \sum_{v \in \mathcal{V}^s} \lambda^s_{v,in}$ is the overall ingress rate.

D. Optimization constraints and objective

We wish to minimize power consumption, while satisfying all SPs' latency constraints. As for the power consumption, we only consider the variable part, due to CPU and bandwidth utilization, as it is the only one that depends on our decision variables. The power consumption due to turning on nodes and links is simply a constant that would add to our power computation, so we do not include it in our objective function.

Let us assume that a replica of a certain component of SP s runs in node v and denote with q its corresponding queue. To model the power consumption, we refer to Fan et al. [22], who have shown that it grows linearly with the CPU utilization [23]. We set power consumed at node v for processing q as (the coefficient 42.29 is from [24]):

$$P_{q,v}(r_q) = \frac{42.29 \cdot r_q}{R_v} [Watts] \tag{4}$$

As for power consumption on links, based on protocol 1000BASE-T in [25], authors of [26] observed that the power consumption tends to saturate around 55 Mb/s for 1518 byte packets. Indeed, when the sending rate is less than 55 Mb/s, packets are sent spaced on the link thus causing frequent transitions in and out from low power mode. The power increases linearly when using less than 55 Mb/s. The consumption of 10 Gb/s Ethernet is between 4.5W - 20W [27]. In our case, the total allocated bandwidth in a physical link (u,u') is $r_{u,u'} = \sum_{q \in \mathcal{Q}_{u,u'}} r_q$. Therefore, the power consumed is

$$P_{u,u'}(r_{u,u'}) = \begin{cases} \frac{14.555}{550} \cdot r_{u,u'} + 4.5, & \text{if } r_{u,u'} < 550\text{Mb/s} \\ 19.055 + 10^{-4} \cdot (r_{u,u'} - 550), & \text{otherwise} \end{cases} \tag{5}$$

Our goal is to minimize the total power consumption while satisfying the delay constraints of the different SPs. Our dimensioning resources problem consists in deciding the amount of CPU resources to each component replica and amount of bandwidth to each virtual link. We need to give more resources

Node computational capacity	$R_v = 1285.2 \cdot 10^9$ Instr. per sec., $\forall v \in \mathcal{V}$ [28]
Physical link capacity	$R_{u,u'} = 10$ Gb/s, $\forall (u,u') \in \mathcal{E}$
Comput. complexity of SP 2	$\alpha^2_{c_i} = 3 \cdot 10^9$ instr/req, $\forall c_i$ [29]
Comm. complexity of SP 2	$\beta^2_{c_i,c_{i+1}} = 85$ Kb/req, $\forall c_i, c_{i+1}$ [29]
Complexities of SP 1	$\alpha^1_{c_i} = \frac{1}{2}\alpha^2_{c_i}$; $\beta^1_{c_i} = \frac{1}{2}\beta^2_{c_i}$
Ingress req rate	$\lambda^1_{v,\text{in}} = \lambda^2_{v,\text{in}} = 20$ req/sec, $\forall v \in \mathcal{V}$
Delay constraints of SP 1	$D^1 = 0.1$ sec (in Fig. 3); $D^1 = 0.02$ sec (in Fig. 5).
Delay constraints of SP 2	$D^2 = 1$ sec (in Fig. 2 and 3); $D^2 = 0.1$ sec (in Fig. 4 and 5).

Table II: Scenario parameters.

to SPs with more stringent latency constraints and with larger request rate. We formalize the problem as follows:

$$\text{OptRes}: \min \sum_{v \in \mathcal{V}} \sum_{q \in \mathcal{Q}_v} P_{q,v}(r_q) + \sum_{(u,u') \in \mathcal{E}} P_{u,u'}(r_{u,u'}) \quad (6)$$

$$\text{s.t.} \quad \sum_{q \in \mathcal{Q}_v} r_q \leq R_v, \forall v \in \mathcal{V} \quad (7)$$

$$\sum_{q \in \mathcal{Q}_{u,u'}} r_q \leq R_{u,u'}, \forall (u,u') \in \mathcal{E} \quad (8)$$

$$T^s\left(\{r_q\}_{q \in \mathcal{Q}^s}\right) \leq D^s, \forall \text{SP } s \quad (9)$$

$$\mu_q(r_q) > \lambda_q, \forall q \in \mathcal{Q}^s \quad (10)$$

$$r_{u,u'} = \sum_{q \in \mathcal{Q}_{u,u'}} r_q, \forall (u,u') \in \mathcal{E} \quad (11)$$

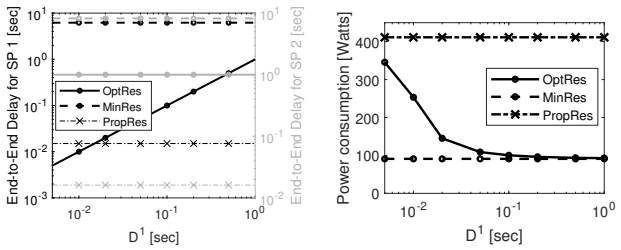
Eqn(7)-(8) guarantee that we do not allocate, on nodes and links, more capacity than available. Eqn (9) ensures that the average journey time of any request does not exceed the delay constraint of the respective slice. Eqn (10) guarantees the stability of Jackson network. Note that the model is unfeasible if not enough physical resources are available to satisfy stability (10) and latency (9) constraints. In this case, the NO should refuse some of the slices. We do not consider this case.

IV. NUMERICAL RESULTS

We consider the network in Fig. 1 and two SPs, with component replicas distributed as in the figure. We assume that user requests can enter the network from any node (i.e., all nodes are ingress nodes). If not otherwise specified, we will use the scenario parameters of Table II. All the journey time and power values presented in this section are expected values. Our model is implemented in the Matlab solver and solved with Sequential Quadratic Programming. Each solution is obtained in less than 5 minutes. We compare the performance of the allocation *OptRes* (6)-(11) with two others, which do not adapt the resource allocation, as common in VNE literature:

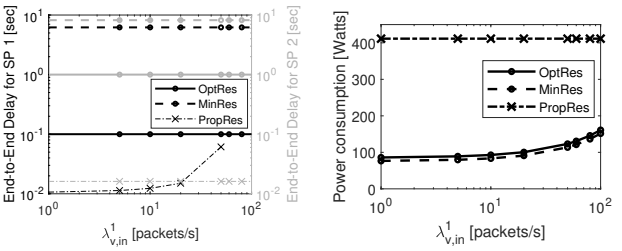
- *MinRes*: only the minimum resources needed to satisfy stability conditions (10) are allocated;
- *PropRes*: all physical link and node resources are used: if a node v hosts replica $c^1_{i,v}$ of SP1 and replica $c^2_{j,v}$ of SP2, the allocation is $r^1_{i,v} = \frac{\alpha_{c^1_{i,v}}}{\alpha_{c^1_{i,v}} + \alpha_{c^2_{j,v}}} \cdot R_v$, $r^2_{j,v} = \frac{\alpha_{c^2_{j,v}}}{\alpha_{c^1_{i,v}} + \alpha_{c^2_{j,v}}} \cdot R_v$.

Impact on different latency requirements. In Fig. 2, we see the performance of resource allocation when the latency requirement of SP 1 varies between 0.005 and 1 seconds.



(a) Delay (b) Power consumption

Figure 2: Delay (SP1 in black and SP2 in grey) and system power used for different latency requirements D^1 of SP 1



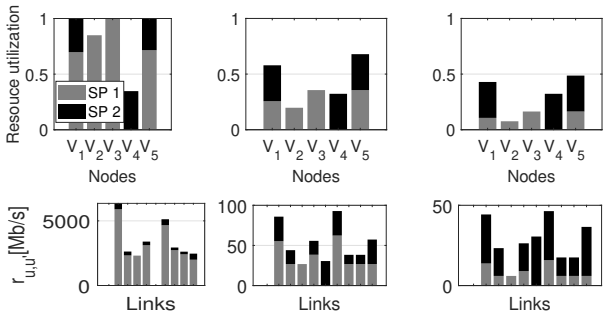
(a) Delay (b) Power consumption

Figure 3: Delay (SP1 in black and SP2 in grey) and power consumption with different values of load $\lambda^1_{v,in}$ of SP 1

As expected, MinRes consumes the least energy but fails to meet he latency requirements of both SPs. On the other hand, PropRes is very energy inefficient. Moreover, it biases allocation in favor of SP 2 (the one with highest request load) without taking into account the different latency requirements of the SPs. As a consequence, when the latency requirements of SP 1 are stringent, PropRes cannot satisfy them. OptRes manages instead to satisfy the latency requirements of both SPs, while being energy efficient. Note that, when latency requirements of SP 1 are not stringent, OptRes is as energy efficient as MinRes.

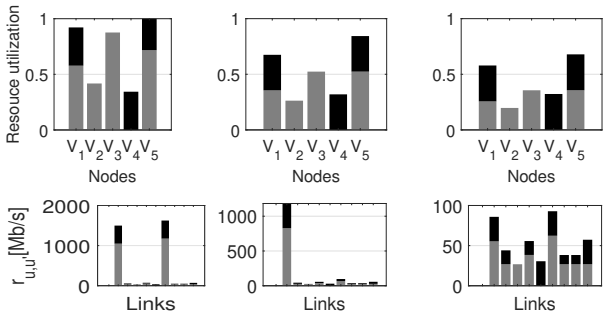
Impact on different load. Fig. 3 shows system performance with different values of ingress request rate $\lambda^1_{v,in}$ of SP 1 (note this value is assumed the same in all nodes). A similar trend to the previous figure is observed: MinRes is energy efficient but does not meet the latency requirements of any SP. On the other extreme, PropRes is energy inefficient. Moreover, it is not able to satisfy the stability conditions of SP 1 when the load is high, i.e., $\lambda^1_{v,in} = 100$ req/sec. With OptRes we manage to adapt well to different loads, getting practically the same energy consumption of MinRes, and satisfying all latency requirements, also with high load.

Node resources utilization and allocated bandwidth: Now we analyze the resource allocation decided by OptRes, with regard to D^1 and $\lambda^1_{v,in}$ (the same for any $v \in \mathcal{V}$). Obviously, the NO allocates 100% computational resources to SP 1 in nodes with no components of SP 2 (nodes v_2, v_3). Note that link v_1, v_2 is never used (see also Fig. 1).



(a) D^1 = 0.005 sec (b) D^1 = 0.02 sec (c) D^1 = 0.1 sec

Figure 4: Impact of delay constraint D^1 on allocation of node resources (a)-(c) and link bandwidth (d)-(e),The bars correspond to the following physical links, in order:
$(v_1, v_2), (v_1, v_3), (v_2, v_1), (v_2, v_5), (v_3, v_1), (v_3, v_4),$
$(v_3, v_5), (v_4, v_3), (v_5, v_2), (v_5, v_3)$



(a) $\lambda^1_{v,in}$ = 100 req/sec (b) $\lambda^1_{v,in}$ = 50 req/sec (c) $\lambda^1_{v,in}$ = 20 req/sec

Figure 5: Impact of the arrival rate $\lambda^1_{v,\text{in}}$ on allocation of node resources (a)-(c) and link bandwidth (d)-(e)

It is can be observed that SP 1 gets more resources when its latency requirements are tighter (Fig. 4) or its load is higher (Fig. 5), while resources to SP 2 are not affected. Note that node CPUs are used a lot, while the 10 Gb/s available on physical links is never fully used. Further analysis (not presented here for lack of space) shows that indeed requests spend much more time in nodes (for being processed) than in links (to be transmitted). Therefore, to meet latency constraints, it is more important to decrease processing time (using CPU cycles) than transmission time. In Fig. 5.(a)-(b), note that links (v_1, v_3) and (v_3, v_5) present a higher consumption than all other. This is due to the power profile of links (5): when exceeding 550 Mb/sec, the marginal power consumption is very small, i.e., the cost of additional bandwidth utilization is negligible.

V. Conclusion

We proposed a novel strategy for optimal slice resource dimensioning, modeled as capacity dimensioning of multiple Jackson networks co-existing in the same resource-constrainted network. Numerical results show that if resource allocation is not adapted to slice requirements and characteristics (as neglected in most work on network slicing), either the overall system is energy-inefficient or some latency constraints are violated. Our resource allocation meets instead both objectives. In future work we will integrate in our model the VNE problem, consider larger networks and real time constraints, instead of the used average time constraints we now use.

Acknowledgement

This work was carried out in the context of Beyond5G, a project funded by the French government as part of the economic recovery plan, namely "France Relance", and the investments for the future program

References

[1] T. V. Doan *et al.*, "A Longitudinal View of Netflix: Content Delivery over IPv6 and Content Cache Deployments," in *IEEE INFOCOM*, 2020.
[2] A. Magalhaes *et al.*, "REPO: A Microservices Elastic Management System for Cost Reduction in the Cloud," in *IEEE ISCC*, 2018.
[3] C. Papagianni *et al.*, "On the optimal allocation of virtual resources in cloud computing networks," *IEEE Transactions on Computers*, 2013.
[4] M. Chowdhury *et al.*, "ViNEYard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping," *IEEE/ACM Tr.on Net.*, 2012.
[5] P. Rodis *et al.*, "Intelligent Network Service Embedding using Genetic Algorithms," in *IEEE ISCC*, 2021.
[6] M. Elkael *et al.*, "Improved Monte Carlo Tree Search for Virtual Network Embedding," in *IEEE LCN*, 2021.
[7] M. Kassis *et al.*, "Flexible Multi-Operator RAN Sharing: Experimentation and Validation Using Open Source 4G/5G Prototype," in *EuCNC 6G*, 2021.
[8] S. D'oro *et al.*, "The Slice Is Served: Enforcing Radio Access Network Slicing in Virtualized 5G Systems," *IEEE INFOCOM*, 2019.
[9] C. Canpolat *et al.*, "Dynamic User Count Aware Resource Allocation for Network Slicing in Virtualized Radio Access Networks," in *IEEE ISCC*, 2020.
[10] S. Schneider *et al.*, "Distributed Online Service Coordination Using Deep Reinforcement Learning," *IEEE ICDCS*, 2021.
[11] A. Araldo *et al.*, "Resource allocation for edge computing with multiple tenant configurations," in *ACM SAC*, 2020.
[12] K. L., *Communication Nets: Stochastic Message Flow and Delay*. McGraw-Hill Book Co., 1964.
[13] L. M. Wein, "Capacity allocation in generalized Jackson networks," *Operations Research Letters*, 1989.
[14] A. B. Dieker *et al.*, "Optimal Resource Capacity Management for Stochastic Networks," *Oper. Res.*, 2017.
[15] Q. Zhang *et al.*, "Joint Optimization of Chain Placement and Request Scheduling for Network Function Virtualization," in *IEEE ICDCS*, 2017.
[16] W. Liang *et al.*, "Low-latency service function chain migration in edge-core networks based on open Jackson networks," *J. Syst. Archit.*, 2022.
[17] T. S. Salem *et al.*, "Towards Inference Delivery Networks: Distributing Machine Learning with Optimality Guarantees," *MedComNet*, 2021.
[18] W. Ben-Ameur *et al.*, "Mathematical Models of the Delay Constrained Routing Problem." *Algorithmic Oper. Res.*, 2006.
[19] A. F. Ocampo *et al.*, "Optimal Service Function Chain Composition in Network Functions Virtualization," in *IFIP Int. Conf. AIMS*, 2017.
[20] K. Kaur *et al.*, "A comprehensive survey of service function chain provisioning approaches in SDN and NFV architecture," *Comp. Sci. Rev.*, 2020.
[21] M. Zukerman, "Introduction to Queueing Theory and Stochastic Teletraffic Models," *ArXiv*, 2013.
[22] X. Fan *et al.*, "Power provisioning for a warehouse-sized computer," in *Proc. ISCA*, 2007.
[23] V. Gupta *et al.*, "An analysis of power reduction in datacenters using heterogeneous chip multiprocessors," *SIGMETRICS PER*, 2011.
[24] S. woo Ham *et al.*, "Simplified server model to simulate data center cooling energy consumption," *Energy and Buildings*, 2015.
[25] P. Reviriego *et al.*, "An Initial Evaluation of Energy Efficient Ethernet," *IEEE Commun. Lett.*, 2011.

[26] ——, "An energy consumption model for Energy Efficient Ethernet switches," in *Int. Conf. HPCS*, 2012.
[27] R. Sohan *et al.*, "Characterizing 10 Gbps network interface energy consumption," in *IEEE Conf. LCN*, 2010.
[28] X. Lyu *et al.*, "Energy-Efficient Admission of Delay-Sensitive Tasks for Mobile Edge Computing," *IEEE Trans. Commun.*, 2018.
[29] S. Josilo *et al.*, "Joint Wireless and Edge Computing Resource Management with Dynamic Network Slice Selection," *CoRR*, 2020.