Exploiting Congestion Games to Achieve Distributed Service Chaining in NFV Networks

Salvatore D'Oro, *Student Member, IEEE*, Laura Galluccio, *Member, IEEE*, Sergio Palazzo, *Senior Member, IEEE*, Giovanni Schembra

Abstract

The Network Function Virtualization (NFV) paradigm has gained increasing interest in both academia and industry as it promises scalable and flexible network management and orchestration. In NFV networks, network services are provided as chains of different Virtual Network Functions (VNFs), which are instantiated and executed on dedicated VNF-compliant servers. The problem of composing those chains is referred to as the Service Chain Composition problem. In contrast to centralized solutions that suffer from scalability and privacy issues, in this paper we leverage *non-cooperative* game theory to achieve a low-complexity distributed solution to the above problem. Specifically, to account for selfish and competitive behavior of users, we formulate the service chain composition problem as an atomic weighted *congestion game* with unsplittable flows and player-specific cost functions. We show that the game possesses a weighted potential function and admits a Nash Equilibrium (NE). We prove that the price of anarchy (PoA) is upper-bounded, and also propose a distributed and privacy-preserving algorithm which provably converges towards a NE of the game in polynomial time. Finally, through extensive numerical results, we assess the performance of the proposed distributed solution to the service chain composition problem.

Index Terms

Game Theory, Congestion Games, Service Chaining, Network Function Virtualization (NFV).

I. Introduction

With the enormous increasing of network traffic during the last few years, and because of the inflexible and ossified structure of the Internet, Telco Operators are introducing a large variety of proprietary hardware appliances (middle-boxes) to provide network functions such as firewalls, Deep Packet Inspectors, Network Address Translator, load balancers, Quality-of-Service (QoS) analyzers, etc. However, deployment of specialized hardware devices for each new network function is very expensive and does not scale, thus causing drastic consequences in both capital expenditures (CAPEX) and operating expenditures (OPEX) [1], and making the network purpose-built and optimized

All authors are with the Dipartimento di Ingegneria Elettrica, Elettronica ed Informatica, University of Catania, Italy;

E-Mail: {name.surname}@dieei.unict.it

This paper has been accepted for publication on *IEEE JSAC Special Issue on Game Theory for Networks*. This is a preprint version of the accepted paper. The final paper is available at doi: 10.1109/JSAC.2017.2659298.

for a few static services only [2, 3]. Recently, Network Function Virtualization (NFV) [4, 5] has been identified as a valid alternative to hardware and vendor-specific middle-boxes. In fact, by decoupling network functions from the physical equipment on which they run, NFV introduces software-based middle-boxes running on cheaper Commercial-Off-The-Shelf (COTS) servers [6] placed in data centers and network nodes. Therefore, NFV has gained a lot of interest in both academia and industry as it promises to reduce CAPEX and OPEX by making networks more scalable and flexible, and leading to increased service agility.

At the same time, Software Defined Networking (SDN), introduced to decouple network control from data forwarding [6], offers the possibility of applying different policies [7, 8] for traffic steering through appropriate network functions, by programmatically configuring forwarding rules implemented in a centralized node called the SDN Controller.

The application of SDN to a network that uses the NFV paradigm represents a major shift in network design, as recognized by both manufacturers and operators [9–11]. In fact, it gives the possibility of running network functions within virtual machines (VMs), which can be migrated and managed by using policies inherited from data center technologies, and steering traffic through arbitrary routes based on user requirements, i.e., along routes which are neither necessarily shortest paths, nor even loop-free [12, 13].

Recent market analysis and forecasts [14] have shown how the SDN/NFV market is expected to exponentially grow in the next years.

As an evidence, a big standardization work has been done in the last years by the ETSI NFV group to develop standards for a reference architecture [15], a management and orchestration (MANO) framework [16] and some use cases [17]. Moreover, a lot of research activities have been carried out in both academia and industry to deploy the SDN/NFV paradigm.

One of the most critical aspects that is slowing the spread of the SDN/NFV paradigm is the service-chaining task, term used ”to describe the deployment of Virtual Network Functions (VNFs), and the network operator’s process of specifying an ordered list of such functions that should be applied to a deterministic set of traffic flows” [18]. The problem of composing an ordered chain of VNFs can be referred to as the *Service Chain Composition problem* [18]. More specifically, this problem regards the decision about the number of instances that have to run on the network for each VNF composing a Network Service (NS), the *VNF Servers* where to execute these instances, and their concatenation to achieve the service chains for each traffic flow in the network. This has to be achieved taking into account the users’ requirements in terms of both QoS parameters and costs to pay.

With rapid increases in traffic volume, traffic variety, and service requirements, Telco Operators need a flexible, agile and scalable method to solve this problem. To this purpose, a great amount of work has been done in the last few years. However, the most of existing literature recognizes that it is an NP-hard problem, and proposes sub-optimal centralized solutions that aim at optimizing a cost function from the Telco Operator point of view. Nevertheless, previous work lacks at least in the following aspects that, indeed, are of extreme importance to the broad market penetration of the SDN/NFV paradigm:

1) Centralized solutions are mainly used to optimize network-wide cost functions. Whilst they lead to a social optimum, those solutions often do not consider user-specific and individualistic behavior of network users.

Accordingly, they do not capture the non-cooperative behavior among competitive and individualistic entities, which aim at their own benefit instead of a social one;

2) To calculate a centralized solution, full knowledge w.r.t. the network state and system parameters is required. It means that the network has to be continuously monitored and each user is required to disclose its private information that is relevant to calculate their cost functions. While private and local information can be disclosed in some applications, there are several scenarios where, for privacy issues, the disclosure of private and local information is an unrealistic and unfeasible assumption as users cannot (or just do not want to) disclose their own private information, such as their cost functions or their system parameters, to other entities in the network. A classic example is that of auctions where users (i.e., the bidders) are expected to compete, and not to share, with other bidders their monetary budget, their private valuation of the auctioned good, and policies they intend to apply to decide the best trade-off between costs and performance [19]. Accordingly, to design an efficient centralized solution in such scenarios is a hard task;
3) Finally, due to the NP-hardness of the VNF composition problem, centralized approaches do not well-scale and require large computational times to converge to an optimal solution. Accordingly, those approaches fail either when the number of variables in the problem is large, or when dealing with dynamic scenarios where the traffic rapidly varies in time and users dynamically access the network. Even though sub-optimal solutions have been proposed in the literature [3, 20–22], those approaches mainly rely on the assumption that full-knowledge is available. An assumption which is not realistic in many application scenarios such as the ones in the above.

With all this in mind, in this paper we aim at proposing a game-theoretic model that approaches the Service Chain Composition problem in a distributed way. To account for the individual and selfish behavior of network users, we exploit non-cooperative game theory. Specifically, contrary to the classical approach that entirely concentrates this task to the Orchestrator [16, 23, 24], here we provide a distributed solution to the problem, and allow network users to play a congestion game to individually find the best service chain of VNF instances that accommodates their individual requirements. We consider NFV-specific network requirements and characteristics, such as the congestion level on VNF servers, the latency incurred by traffic flows, and the price charged by VNF Servers to execute VNFs.

In more detail, to model the congestion level experienced by users on the VNF Servers of the chain, we formulate the problem as an atomic weighted congestion game with unsplittable flows and player-specific cost functions, which also allows us to account for user asymmetries such as different locations in the network and different transmission rates. We show that the congestion game admits a weighted potential function. We also show the existence of a Nash Equilibrium (NE) and that the Price of Anarchy (PoA) is bounded by $\frac{1}{2}(3+\sqrt{5})$. This result is of practical importance as it provides an upper-bound on the performance degradation generated by distributing the service chain composition problem. We propose an algorithm which provably converges in polynomial time to a NE of the game, and can be implemented in a fully distributed fashion while preserving the privacy of network users. Finally, through experimental results, we assess and verify the effectiveness of the proposed distributed service chaining algorithm.

The remainder of this paper is organized as follows. Related work is presented in Section II. In Section III, we describe the reference system. A game-theoretic model of the Service Chain Composition problem is formalized in Section IV, and its implementation aspects are discussed in Section V. Numerical results are illustrated in Section VI,

and an experimental proof-of-concept is presented in Section VII. Finally, in Section VIII conclusions are drawn.

II. Related Work

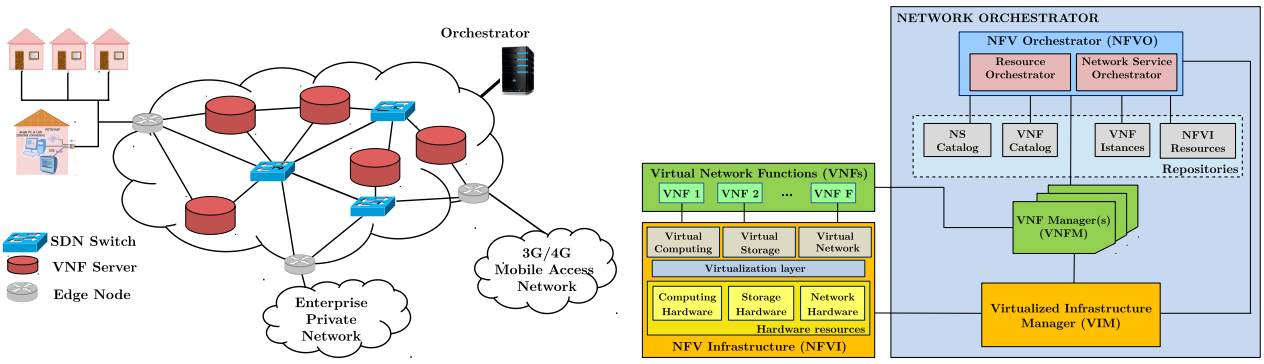
Service chaining is one of the most challenging problems in SDN/NFV network deployment. It aims at routing traffic flows according to a "service graph" so that complex network services can be implemented in software according to the NFV paradigm, thus avoiding any need to make changes to the network at the hardware level. The challenge is to provide Telco Operators to dynamically include VNFs in a network-processing path by addressing requirements for both optimization of the network, through better utilization of resources, and revenue, through the provisioning of services that are tailored to the customer context. In this regard, [17] describes the use case "VNF forwarding graph", and show how traditional physical appliance forwarding graphs are less efficient than VNF forwarding graphs in terms of resilience, flexibility, complexity and deployability. Although service chaining has already been enabled by recent SDN architecture proposals [25], the efficient composition of those service graphs is of crucial importance. Therefore, optimal and sub-optimal solutions to the service chain composition problem have been proposed in the literature. Due to its similarity with the Virtual Network Embedding Problem (VNEP) [26] where virtually orthogonal networks are instantiated on top of a shared network infrastructure, many optimal solutions rely on approaches proposed to solve the VNEP. For example, Integer Linear Programming (ILP) [22, 27, 28] and Mixed ILP (MILP) [29] formulations derived from the VNEP have been widely exploited to optimize a variety of network performance metrics and provide solutions to the service chain composition problem, while a MILP formulation for a coordinated node and link mapping onto the underlying network infrastructure is proposed in [30].

Nevertheless, while optimal solutions are desirable, this is often obtained at a high cost in terms of both computational complexity. In fact, those solutions result in NP-hard problems that expose significant scalability issues. Therefore, unless static scenarios and long optimization times are considered, optimal solutions cannot be derived in large networks due to the combinatorial nature of the problem.

As a workaround, the literature proposes some sub-optimal approaches to the service chain composition problem. For example, [31] introduces a greedy approximation solution to the problem of placing a minimum number of network functions under minimum performance guarantee. Similar approaches have been considered in [20], while heuristics [21, 22] and approximations [3] have also been proposed. Nevertheless, while the above approaches are able to provide sub-optimal solutions in polynomial time, they require full-knowledge of network parameters such as user-specific and private parameters. That is, to obtain a solution, each user should partially or completely disclose its private and local parameters to the centralized entity.

Under the above assumption, centralized approaches fail as the lack of perfect information does not allow to obtain an efficient and globally optimal solution. This is because distributed mechanisms should be adopted as they are well-suited to easily tackle the above requirements and provide efficient solutions to the service chain composition problem.

Moreover, it is worth noting that network users are often competitive entities that selfishly aim at maximizing their own benefits or minimizing costs and, in almost all real-world applications, they are not aware of the identity of the other users or how many users are accessing the network. Therefore, a non-cooperative game-theoretic approach can



(a) (b)

Fig. 1. a) The network scenario; b) The reference NFV System Architecture.

be effectively used to model such a competitive scenario under the above privacy requirements. A framework that is well-suited to provide a distributed VNF service chain composition mechanism is that of congestion games.

Congestion games have been proposed in many networking scenarios thanks to their ability to capture congestion of network resources, and to model a hierarchical structure like the one distinctive of the VNF chaining problem. Also, congestion games have been shown to possess some interesting properties. For example, the existence of potential functions in particular classes of potential games has been proven in [32, 33]. By exploiting the above properties, in [34, 35] centralized polynomial algorithms that provably converge to a NE of the game are proposed. However, the above approaches do not consider either user-specific functions, or cannot be implemented in a privacy-preserving and distributed fashion as they require the whole knowledge of the network configuration, e.g., the network graph and the current VNF chaining configuration. Therefore, they cannot be used to solve the considered service chaining problem, which instead is the objective of this paper.

III. The Reference System

As shown in Fig. 1(a), in this paper we consider a network owned by a Telco Operator whose objective is to provide its customers with network services following the SDN/NFV approach. To this purpose, together with some legacy nodes, in the network there are SDN switches, as well as NFV-compliant nodes here referred to as VNF Servers. The former, according to the SDN paradigm, are able to steer traffic according to some rules received from a remote SDN Controller. The latter follow the specifications in [36], with an architecture like the one shown on the left part of Fig. 1(b). More specifically, in the VNF Servers a virtualization layer is installed above the computing, storage and network hardware resources to have the possibility of running instances of VNFs in virtual machines (VMs), applying the same virtualization technologies used in data centers and clouds. All the above nodes constitute the NFV Infrastructure (NFVI).

Traffic flows are generated by the Telco Operator network customers, which access the Telco Operator network through *edge nodes*. Each customer can generate different traffic flows and, for each of them, it requests a network

service with a specific level of quality, according to the type and the importance of the flow[1].

Any traffic flow is characterized by its ingress edge node, its edge egress node, and the required network service with the associated level of quality. A *network service* (NS) can be either constituted by a single VNF, or composed by a chain of VNFs. Without losing in generality and to simplify readability, in the following we will refer to a traffic flow as either a single traffic flow, or an aggregate of traffic flows that, although generated by different customers or different applications of the same customer, have the same ingress and egress edge nodes, and require the same NS with the same level of quality.

The key role of orchestration and resource management in the system shown in Fig. 1(a) is played by the *Network Orchestrator*. In order to be compliant with the Management and Orchestration (MANO) specifications defined for NFV networks in [16], its architecture is as the one shown on the right part of Fig. 1(b). The main target of the Network Orchestrator is to instantiate VMs to run VNFs on the available VNF Servers, decide the amount of resources that servers have to reserve to each VNF in terms of computing, storage and network resources, and arranging the NFVI resources in such a way to provide customers with the required network services with the relative expected quality. More in depth, the NFV Orchestrator (NFVO) block within the Network Orchestrator is responsible of the orchestration of the NFVI resources, fulfilling the Resource Orchestration functions and the Network Service Orchestration functions, the last regarding the lifecycle management of network services.

For network services composed by a single VNF, the Network Service Orchestrator decides how many VNF instances have to be executed in the network, and the VNF Servers that have to run them. This is achieved by interacting with the Virtualized Infrastructure Manager (VIM) blocks that are in charge of managing the NFVI resources. Likewise, in the case of more complex network services that are realized as chains of different VNFs, the Network Service Orchestrator decides the number of instances for each component VNF, and their placement. In addition, it is in charge of the Service Chain Composition task, which consists in composing network services by chaining the running VNF instances. Of course, due to the presence of different instances for each VNF, many chains can coexist to provide different customers with the same network service.

This task is challenging because it constitutes a very hard optimization problem with many constraints. Therefore, a complex and centralized optimization algorithm to support the NFVO in its decision process is needed to satisfy user requirements while minimizing a given cost function. An additional related issue is that the NFVO, in order to achieve this task, should know all the information regarding each traffic flow, some of which can be private and sensitive.

The target of this paper, which will be addressed in the following section, is to define a game that allows distributing this task, in such a way that both the computational scalability and customer privacy issues are addressed.

To this purpose, we introduce a new entity, the *Network Service Broker* (NSB), whose instances work as *players* of

[1]For example, as compared to video streaming flows, email traffic flows require a different network service, and even two different video streaming flows may require different levels of quality, according to their importance and the willingness of the customers to pay no more than a proper price. Network services with the associated levels of quality can be requested by single customers, typically when customers are the owners of enterprise or residential private networks. Alternatively, when many customers enter the network through an access network owned by a third party provider, e.g., a 3G/4G mobile access network, network services provided to network customers are predefined by the access network provider. In some cases, traffic can be divided in classes and a network service is assigned to each traffic class.

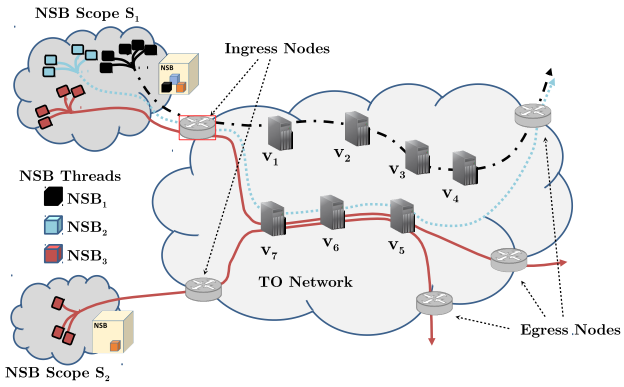


Fig. 2. An illustrative example of Network Service Brokers and their scopes.

the game we will define in the following section. An NSB instance is a software application that covers a delimited portion of network, which in the following will be indicated as *NSB Scope*. An instance of NSB can be run either 1) on user terminals, 2) on Customer Premise Equipments (CPEs) that are used to access the Internet from a private network (e.g. the home router in a residential local area network, or the enterprise router in a business environment), or 3) on the access network of third party providers, for example in the case of 3G/4G mobile access networks. Accordingly, the NSB works for the flows generated from either the same terminal (case 1), the same private network (case 2), or the same access network (case 3). In particular, an NSB starts a thread for each flow to be managed. All NSB threads that request the same network service and belong to different NSB Scopes will compete with each other as players of a non-cooperative game.

Fig. 2 shows an example highlighting two NSB Scopes, S_1 and S_2, the first with three flows and the second one with only one flow. Therefore, three different NSB threads run in the NSB Scope S_1, while one NSB thread works in the NSB Scope S_2. Since the flow coming from S_2 requires the same NS of one of the flows from S_1, the two threads labeled as NSB_3 in Fig. 2 (one in each NSB Scope) compete with each other as players of the same game.

IV. Game Model

In this section, we formulate the Service Chain Composition problem as a congestion game [37]. Specifically, in Section IV-A, we present the notation used in the paper and the relevant assumptions that will be used to define the game-theoretic framework. We show that the congestion game admits both a weighted potential and a NE that can be computed in a fully distributed fashion by exploiting unilateral best response dynamics in a finite amount of iterations. A brief description of the congestion games will be provided in Section IV-B. Then, in Section IV-C we derive both a closed-form weighted potential function and an upper bound for the price of anarchy (PoA).

A. Model Notation and Assumptions

Let us introduce some notation and the assumptions needed to model the system described so far. All relevant system parameters are summarized in Table I.

TABLE I
NOTATION

Variable	Description
$\mathcal{F}$	The considered network service
$\mathcal{V}$	Set of VNF Servers
$\mathcal{N}$	Set of players in the network
$\mathcal{F}_v \subseteq \mathcal{F}$	The set of VNF functions offered by VNF Server $v \in \mathcal{V}$
F, V, N	Number of VNF functions, VNF Servers and players
$p_{v,f}$	Price imposed by server v to execute VNF $f \in \mathcal{F}$
$\mathcal{W}$	All possible NS configurations for each player
$\mathbf{D}^{(v-v)}, \mathbf{D}^{(in-v)}, \mathbf{D}^{(v-out)}$	Inter-server, Ingress-to-Server and Server-to-Egress latency matrices
$d_{a,b}$	Latency between network nodes a and b
(s_i, t_i)	Ingress and egress nodes for flow managed by player $i \in \mathcal{N}$
λ_i	Bit rate of flow managed by player i
$w_i(f) \in \mathcal{V}$	VNF Server selected by player i to execute VNF f
$\underline{w}_i = (w_i(1), w_i(2), \ldots, w_i(F)) \in \mathcal{W}$	NS configuration of player i
$c_i^{(\mathcal{P})}, c_i^{(\mathcal{T})}, c_i^{(\mathcal{C})}$	Price, latency and congestion costs of the service chain for player i
$\mathcal{C}_i$	Cost function for player i
γ_i, β_i	Weighting parameters in (6) for player i

In this paper, we focus on one Network Service (NS) out of those in the NS Catalog located in the Network Orchestrator. The NS can be modeled as an NFV forwarding graph consisting of a chain of VNFs. Throughout the paper, the considered network service will be denoted as $\mathcal{F}$ and the NSB threads will be identified with their corresponding NSB[2]. Accordingly, we have that $\mathcal{F} = \{1, 2, \ldots, F\}$ is the considered service chain, defined as an ordered sequence of VNFs, where F is the last VNF that has to be executed to complete the NS.

The game will be defined among all the NSBs that are in charge of the management of $\mathcal{F}$. We assume that V dedicated VNF Servers are deployed in the network to execute the Virtual Network Functions (VNFs). Let $\mathcal{V} = \{1, 2, \ldots, V\}$ be the set of the available VNF Servers.

For each server $v \in \mathcal{V}$, let $\mathcal{F}_v \subseteq \mathcal{F}$ be the (sub)set of VNFs provided by the server v. Accordingly, for each VNF $f \in \mathcal{F}_v$, let $p_{v,f}$ be the *VNF price* to execute an instance of the VNF f on the server v. In the following, and without loss of generality, we will assume that all servers in $\mathcal{V}$ have all the VNFs in $\mathcal{F}$, i.e. $\mathcal{F}_v = \mathcal{F}$. However, our model also applies to the case when $\mathcal{F}_v \subset \mathcal{F}$ for some $v \in \mathcal{V}$.

Let $\mathcal{N}$ be the set of the NSBs in the network whose flows request the network service $\mathcal{F}$, and N be the cardinality of $\mathcal{N}$. Since network customers are located at different areas of the network, for each NSB $i \in \mathcal{N}$, we define a 2-tuple (s_i, t_i), where s_i and t_i are the ingress and the egress nodes of the flow managed by NSB $i \in \mathcal{N}$, respectively. Also, let $\lambda_i > 0$ be the bit rate of the flow handled by the i-th NSB.

[2]Note that the more general case where many network services are provided to network users can be tackled with the same approach presented in this paper by exploiting the orthogonality provided by network slicing. This allows to virtually create independent network and computing slices for each service chain without mixing traffic generated by NSB threads requesting different chains.

To take network latencies into account, let $\mathbf{D}^{(v-v)} = \left(d^{(v-v)}_{v'v''}\right)_{v',v''\in\mathcal{V}}$ be the inter-server latency matrix containing the latencies between all VNF Servers involved in the Network Service $\mathcal{F}$. Analogously, let $\mathbf{D}^{(in-v)} = \left(d^{(in-v)}_{i,v}\right)_{i\in\mathcal{N},v\in\mathcal{V}}$ be the ingress-to-Server latency matrix which contains the latencies between all ingress nodes and VNF Servers. Finally, let $\mathbf{D}^{(v-out)} = \left(d^{(v-out)}_{v,i}\right)_{v\in\mathcal{V},i\in\mathcal{N}}$ be the Server-to-egress latency matrix containing all latencies between VNF Servers and egress nodes.

For the sake of readability, and to simplify the notation, in the following we omit all superscripts and use the uniform notation $d_{a,b}$ to identify the latency between nodes a and b, where a, b can be either servers in $\mathcal{V}$, ingress or egress nodes. Also, without loss of generality, we consider the case where, for any two distinct servers $v', v'' \in \mathcal{V}$, there always exists a path in the network that interconnects v' to v'' (and viceversa).

We assume that the NS is composed by a chain of F VNFs and that the available VNF Servers are V. Accordingly, the number of possible chaining combinations is $|\mathcal{W}| = V^F$. Let $\mathcal{W}$ be the set containing all possible NS configurations, and let $\underline{w}_i \in \mathcal{W}$ be the NS configuration chosen by the NSB representing the flow $i \in \mathcal{N}$. In the following, we refer to this NSB as the i-th *player*, and we use terms NSB and player interchangeably. In our model, the NS configuration $\underline{w}_i$ is the F-tuple $\underline{w}_i = (w_i(1), w_i(2), \ldots, w_i(F))$, where $w_i(f) \in \mathcal{V}$ is the server which has been chosen by the player i to execute the function $f \in \mathcal{F}$.

For each player $i \in \mathcal{N}$ and any given NS configuration $\underline{w}_i$ chosen by i, the price of the NS configuration can be written as

$$c_i^{(\mathcal{P})}(\underline{w}_i) = \sum_{f=1}^{F} p_{w_i(f),f} \tag{1}$$

Similarly, the overall latency experienced by the flow managed by the player i when choosing the NS configuration $\underline{w}_i$ can be expressed as

$$c_i^{(\mathcal{T})}(\underline{w}_i) = d_{i,w_i(1)} + \sum_{f=2}^{F} d_{w_i(f-1),w_i(f)} + d_{w_i(F),i} \tag{2}$$

where the term $d_{i,w_i(1)} \in \mathbf{D}^{(in-v)}$ is the latency between node s_i, representing the ingress node of the flow managed by player i, and the server $w_i(1) \in \mathcal{V}$, the term $d_{w_i(f-1),w_i(f)} \in \mathbf{D}^{(v-v)}$ is the inter-server latency between servers $w_i(f-1)$ and $w_i(f)$, and the term $d_{w_i(F),i} \in \mathbf{D}^{(v-out)}$ is the latency between the server $w_i(F)$ and the egress node of the i-th NSB's flow.

Finally, since different flows can use the same instance of the VNF f, they will experience a congestion level due to the load on the server where f is running. Let $\mathbf{w}_{-i}$ be the set of all the NS configurations chosen by the NSBs in $\mathcal{N}\backslash\{i\}$, that is, $\mathbf{w}_{-i} = (\underline{w}_1, \underline{w}_2, \ldots, \underline{w}_{i-1}, \underline{w}_{i+1}, \ldots, \underline{w}_N)$. Furthermore, let $\Gamma^f(\underline{w}_i, \mathbf{w}_{-i}, v)$ be the set of NSBs in $\mathcal{N}$ which have chosen the server $v \in \mathcal{V}$ to receive the function $f \in \mathcal{F}$. Let us indicate the NS configuration chosen by the NSB j as $\underline{w}_j$, and the set of the NS configurations chosen for all the NSBs requiring the considered network service $\mathcal{F}$ as $(\underline{w}_i, \mathbf{w}_{-i}) \in \mathcal{W}^N$, where $\mathcal{W}^N$ stands for the N-ary Cartesian power of the set $\mathcal{W}$. In a formal way, we have

$$\Gamma^f(\underline{w}_i, \mathbf{w}_{-i}, v) = \left\{ j \in \mathcal{N} : w_j(f) = v, w_j(f) \in \underline{w}_j, \underline{w}_j \in (\underline{w}_i, \mathbf{w}_{-i}) \right\} \tag{3}$$

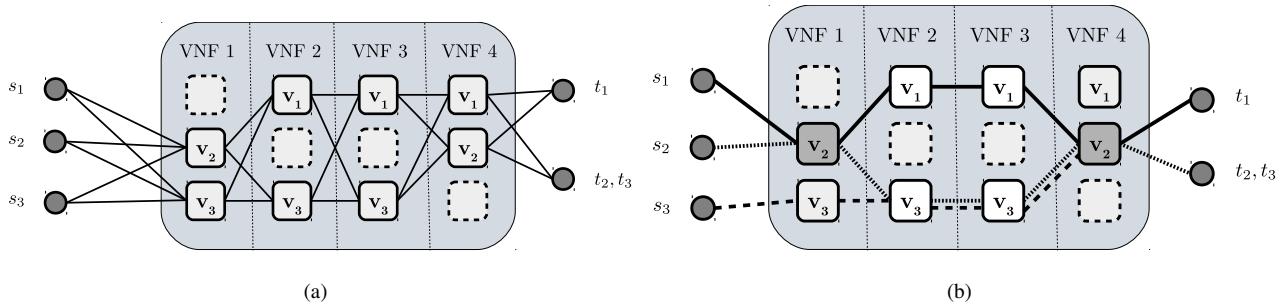


Fig. 3. a) Service Chain Model with $F = 4$, $V = 3$ and $N = 3$; b) An illustrative example of a possible service chain configuration.

Accordingly, we have that the congestion level experienced by the flow corresponding to the NSB i for function f on the chosen server $w_i(f)$ is

$$\delta_f(\underline{w}_i, \mathbf{w}_{-i}) = \sum_{j \in \Gamma^f\left(\underline{w}_i, \mathbf{w}_{-i}, w_i(f)\right)} \lambda_j \tag{4}$$

where $\delta_f(\underline{w}_i, \mathbf{w}_{-i}) = \delta_f(\underline{w}_j, \mathbf{w}_{-j})$ if $w_j(f) = w_i(f)$.

By exploiting (4), we have that the overall congestion level experienced by the flow of player $i \in \mathcal{N}$ on the whole service chain when the other players have chosen the NS configurations described by $\mathbf{w}_{-i}$, is

$$c_i^{(\mathcal{C})}(\underline{w}_i, \mathbf{w}_{-i}) = \sum_{f=1}^{F} \delta_f(\underline{w}_i, \mathbf{w}_{-i}) \tag{5}$$

Accordingly, the total cost experienced by each NSB $i \in \mathcal{N}$ can be expressed by the following *cost function*:

$$\mathcal{C}_i(\underline{w}_i, \mathbf{w}_{-i}) = c^{(\mathcal{C})}(\underline{w}_i, \mathbf{w}_{-i}) + \gamma_i \left(c_i^{(\mathcal{T})}(\underline{w}_i) + \beta_i c_i^{(\mathcal{P})}(\underline{w}_i) \right) \tag{6}$$

where γ_i and β_i are two non-negative weighting parameters decided by the player i.

For the sake of illustration, the considered service chain model is illustrated in Fig. 3(a). Specifically, we show an illustrative system configuration where $F = 4$ VNFs compose the NS $\mathcal{F} = \{1, 2, 3, 4\}$, $V = 3$ VNF Servers are in the network, and $N = 3$ players play the game in order to choose the NS configuration for their flows. As shown in Fig. 3(a), some servers, highlighted with dashed border lines, do not provide one or more functions in the chain. Also, it may happen that some players in $\mathcal{N}$ share the same ingress and/or egress nodes, i.e., NSBs $i = 2, 3$ that have the same egress node $t_2 = t_3$, even if they have different ingress nodes $s_2 \neq s_3$.

Moreover, in Fig. 3(b), we show an illustrative example where we represent how service chaining is performed in the considered scenario. The service chains chosen for the three flows, $\underline{w}_1$, $\underline{w}_2$ and $\underline{w}_3$, are represented by solid, dotted and dashed lines, respectively. As shown in Fig. 3(b), flows 1 and 2 share the same server v_2 to execute the VNF 1, and all of the three flows share server v_2 to execute the VNF 4. Therefore, from (4) we have that the server load at v_2 w.r.t. VNF 1 is equal to $\lambda_1 + \lambda_2$. Instead, the server load on the same server w.r.t. VNF 4 is $\lambda_1 + \lambda_2 + \lambda_3$, while the server load at v_3 w.r.t. VNF 1 is equal to λ_3, as only flow 3 has selected v_3 to execute VNF 1.

B. Congestion Games in a Nutshell

Congestion games are a particular class of games where a set $\mathcal{N}$ of players compete with each other over a given finite set $\mathcal{R}$ of resources. In such games, which were first introduced in [37] by Rosenthal, each player's strategy consists in selecting a resource $r \in \mathcal{R}$. This is done taking into account that the cost $c_r(n_r)$ of selecting such resource, which can also be referred to as the congestion over resource r, depends on the number $n_r \leq N$ of players in $\mathcal{N}$ which have selected resource r. Congestion games are well-known to be *potential games* [37]. In fact, they admit a *potential function*, which is a real-valued function that tracks the changes in the cost functions of players which unilaterally deviate from a strategy to another.

A particular extension of congestion games, which is also referred to as *weighted congestion game*, assigns a weight λ_i to each player $i \in \mathcal{N}$ such that the cost functions are player-specific functions of the form $c_{i,r}(n_r, (\lambda_i)_{i\in\mathcal{N}})$. The latter formulation finds application in many wired and wireless network scenarios [38–40] where weights can be associated to the traffic generated by network users, e.g., transmission data rate, and player-specific cost functions can be used to capture channel-state information and different positions of users in the network.

An important aspect which should not be neglected in congestion games is related to the efficiency of equilibrium points w.r.t. an optimal solution. To this purpose, the concept of price of anarchy (PoA) has been considered. Specifically, if the objective of players is to minimize a given cost function, the PoA is defined as the ratio between the cost experienced by all players at the worst NE of the game and that achieved by an optimal centralized solution. Intuitively, the PoA measures the worst-case performance of the game. A unitary PoA means that the NE is also optimal. Instead, the higher the PoA, the worse the efficiency of the NE.

C. Game Formulation

Each NSB is a player and its objective is to minimize the cost function defined in (6) for the flows it manages. In this section we will define the game that the NSBs play to create service chains for their flows.

Accordingly, we define the weighted congestion game as the tuple

$$\mathcal{G} = \left(\mathcal{N}, (\lambda_i)_{i\in\mathcal{N}}, (\mathcal{V}_f)_{f\in\mathcal{F}}, \mathcal{W}^N, (c_{i,v,f})_{i\in\mathcal{N},v\in\mathcal{V},f\in\mathcal{F}}\right)$$

where $\mathcal{N}$ is the set of players, the *weights* are the bit rates λ_i of the relative flows, the set $\mathcal{V}_f$ of servers which provide VNF $f \in \mathcal{F}$ corresponds to the set of *resources* that can be selected by players, and $\mathcal{W}$ are all the possible NS configurations that can be chosen by the player $i \in \mathcal{N}$. In the following of this paper, we will refer to the VNF Servers as the resources of the congestion games. Finally, for each function $f \in \mathcal{F}$ and resource $w_i(f) \in \mathcal{V}_f$ chosen by player i to execute VNF f, $c_{i,w_i(f),f}$ is the per-resource cost function which is defined as follows:

$$c_{i,w_i(f),f} = \begin{cases} \delta_f(w_i(1), \mathbf{w}_{-i}) + \gamma_i \left(d_{i,w_i(1)} + \beta_i p_{w_i(1),1}\right) & \text{if } f = 1 \\ \delta_f(w_i(f), \mathbf{w}_{-i}) + \gamma_i \left(d_{w_i(f-1),w_i(f)} + \beta_i p_{w_i(f),1}\right) & \text{if } 1 < f < F \\ \delta_f(w_i(F), \mathbf{w}_{-i}) + \gamma_i \left(d_{w_i(f-1),w_i(f)} + d_{w_i(f),i} + \beta_i p_{w_i(F),1}\right) & \text{if } f = F \end{cases} \tag{7}$$

where $\delta_f(v, \mathbf{w}_{-i})$ is defined in (4) and represents the server load at VNF Server $v \in \mathcal{V}$ w.r.t. VNF f; $w_i(f-1) \in \mathcal{V}_{f-1}$ is the server chosen by the player i to provide the flow with the VNF $f-1$; $d_{i,w_i(1)} \in \mathbf{D}^{(in-v)}$, $d_{w_i(f-1),w_i(f)} \in \mathbf{D}^{(v-v)}$ and $d_{w_i(f),i} \in \mathbf{D}^{(v-out)}$ are the Ingress-to-Server, Inter-server and Server-to-Egress latencies, respectively.

From (7), it is worth noting that the cost experienced by each flow to receive a given function f on a given server $v \in \mathcal{V}_f$ depends on the congestion level $\delta_f(v, \mathbf{w}_{-i})$ at that given server, the latency to reach that server (which also depends on the server $w_i(f-1)$ that has been chosen to execute function $f-1$), and on the price $p_{v,f}$ to execute function f on server v.

Accordingly, a *strategy* for a player $i \in \mathcal{N}$ consists of a NS configuration, i.e., a sequence of servers where each function in $\mathcal{F}$ has to be executed. In more detail, we have that $\underline{w}_i = (w_i(1), w_i(2), \ldots, w_i(F))$ is a strategy for player $i \in \mathcal{N}$, where each $w_i(f) \in \mathcal{V}_f$ for all $f \in \mathcal{F}$. Therefore, a *strategy profile* for the game $\mathcal{G}$ consists of a tuple $\mathbf{w} = (\underline{w}_1, \underline{w}_2, \ldots, \underline{w}_N) \in \mathcal{W}^N$ such that $\underline{w}_i \in \mathcal{W}$ for all $i \in \mathcal{N}$.

With a more in-depth look at the above weighted congestion game, we can notice that $\mathcal{G}$ is *atomic* as the traffic is generated by single applications in the network, is $F-$*layered* because all paths for any 2-tuple (s_i, t_i) have length exactly equal to F, and has *player-specific* cost functions as (7) varies from a player to another one. Finally, $\mathcal{G}$ is a congestion game with *unsplittable flows* since a flow follows a unique path from its ingress node to its egress node.

Definition 1. *A strategy profile $(\underline{w}_1^*, \underline{w}_2^*, \ldots, \underline{w}_N^*) \in \mathcal{W}^N$ is a Nash Equilibrium if, $\forall i \in \mathcal{N}$ and $\forall \underline{w}_i \in \mathcal{W}$, we have:*

$$\mathcal{C}_i(\underline{w}_i^*, \mathbf{w}_{-i}^*) \leq \mathcal{C}_i(\underline{w}_i, \mathbf{w}_{-i}^*)$$

that is, $(\underline{w}_1^*, \underline{w}_2^*, \ldots, \underline{w}_N^*)$ is a strategy profile where no player has incentive to deviate unilaterally.

Definition 2. *A weighted potential function for the game $\mathcal{G}$ is a function $\Phi : \mathcal{W}^N \to \mathbb{R}$ such that, $\forall i \in \mathcal{N}$, $\forall \underline{w}_i, \underline{w}_i' \in \mathcal{W}$, and $\forall \mathbf{w}_{-i} \in \mathcal{W}^{N-1}$, there exists a vector $b = (b_i)_{i \in \mathcal{N}}$ such that*

$$\Phi(\underline{w}_i, \mathbf{w}_{-i}) - \Phi(\underline{w}_i', \mathbf{w}_{-i}) = b_i \left[\mathcal{C}_i(\underline{w}_i, \mathbf{w}_{-i}) - \mathcal{C}_i(\underline{w}_i', \mathbf{w}_{-i})\right] \tag{8}$$

In the following, we first show that game $\mathcal{G}$ admits a *weighted potential function*, and then we show that it admits at least a NE.

Theorem 1. *The congestion game $\mathcal{G}$ admits a weighted potential function with weighting vector $b = (2\lambda_i)_{i \in \mathcal{N}}$ given by*

$$\Phi(\underline{w}_i, \mathbf{w}_{-i}) = 2 \sum_{i \in \mathcal{N}} \lambda_i \tilde{c}_i(\underline{w}_i) + \sum_{f=1}^{F} \sum_{v \in \mathcal{V}} \left[\delta_f(\underline{w}_i, \mathbf{w}_{-i})\right]^2 \tag{9}$$

where $\tilde{c}_i(\underline{w}_i) = \gamma_i \left(c_i^{(\mathcal{T})}(\underline{w}_i) + \beta_i c_i^{(\mathcal{P})}(\underline{w}_i) \right)$

Proof: Let $\mathbf{w}_{-k} \in \mathcal{W}^{N-1}$ be the NS configuration (or strategy profile) chosen by all the players in $\mathcal{N}$ except for the player k. Also, let $\underline{x}$ and $\underline{y}$ be two different NS configurations[3] in $\mathcal{W}$ chosen by the player k. To prove the theorem, according to Definition 8, we need to show that the following equation holds

$$\Phi(\underline{x}, \mathbf{w}_{-k}) - \Phi(\underline{y}, \mathbf{w}_{-k}) = 2\lambda_i \left(\mathcal{C}_k(\underline{x}, \mathbf{w}_{-k}) - \mathcal{C}_k(\underline{y}, \mathbf{w}_{-k}) \right) \tag{10}$$

[3]Two NS configurations $\underline{x}, \underline{y} \in \mathcal{W}$ are said to be different if there exists at least one VNF $f \in \mathcal{F}$ such that $x(f) \neq y(f)$.

From (6), we have that

$$\mathcal{C}_k(\underline{x}, \mathbf{w}_{-k}) - \mathcal{C}_k(\underline{y}, \mathbf{w}_{-k}) = \left[\tilde{c}_k(\underline{x}) - \tilde{c}_k(\underline{y})\right] + \sum_{f \in F} \left(c^{(\mathcal{C})}(\underline{x}, \mathbf{w}_{-k}) - c^{(\mathcal{C})}(\underline{y}, \mathbf{w}_{-k})\right) \tag{11}$$

Let $F_{\underline{x}\setminus\underline{y}} = \{f \in \mathcal{F} : x(f) \neq y(f)\}$ be the set of functions which are executed on different servers when NS configurations $\underline{x}$ and $\underline{y}$ are chosen by the player k. It is easy to realize that $F_{\underline{x}\setminus\underline{y}} = F_{\underline{y}\setminus\underline{x}}$. From (5) and (4), we have that $\delta_f(\underline{x}, \mathbf{w}_{-i}) = \delta_f(\underline{y}, \mathbf{w}_{-i})$ for all $f \notin F_{\underline{x}\setminus\underline{y}}$. That is, the congestion level δ_f does not vary on those servers which are selected by the player k in both NS configurations $\underline{x}$ and $\underline{y}$. On the contrary, it is straightforward to show that $\delta_f(\underline{x}, \mathbf{w}_{-i}) \neq \delta_f(\underline{y}, \mathbf{w}_{-i})$ if $f \in F_{\underline{x}\setminus\underline{y}}$. Accordingly, from (4), (11) can be rewritten as

$$\mathcal{C}_k(\underline{x}, \mathbf{w}_{-k}) - \mathcal{C}_k(\underline{y}, \mathbf{w}_{-k}) = \left[\tilde{c}_k(\underline{x}) - \tilde{c}_k(\underline{y})\right] + \sum_{f \in F_{\underline{x}\setminus\underline{y}}} \left(\sum_{j \in \Gamma^f(\underline{x}, \mathbf{w}_{-k}, x(f))} \lambda_j - \sum_{j \in \Gamma^f(\underline{y}, \mathbf{w}_{-k}, y(f))} \lambda_j \right) \tag{12}$$

Now we show that (9) satisfies (8), i.e., the function Φ is a weighted potential function for game $\mathcal{G}$. From (9), we have that

$$\begin{aligned} \Phi(\underline{x}, \mathbf{w}_{-k}) - \Phi(\underline{y}, \mathbf{w}_{-k}) = {} & 2 \sum_{i \in \mathcal{N}\setminus\{k\}} \lambda_j \left(\tilde{c}_j(\underline{x}) - \tilde{c}_j(\underline{y})\right) \\ & + 2\lambda_k \left(\tilde{c}_k(\underline{x}) - \tilde{c}_k(\underline{y})\right) + \sum_{f=1}^{F} \sum_{v \in \mathcal{V}_f} \left[\left(\delta_f(\underline{x}, \mathbf{w}_{-k})\right)^2 - \left(\delta_f(\underline{y}, \mathbf{w}_{-k})\right)^2\right] \end{aligned} \tag{13}$$

By assumption, only the player k is changing its strategy from $\underline{y}$ to $\underline{x}$; accordingly, we have that $\tilde{c}_j(\underline{x}) = \tilde{c}_j(\underline{y})$ for all $j \neq k$. Thus, we have that $\sum_{i \in \mathcal{N}\setminus\{k\}} \left(\tilde{c}_j(\underline{x}) - \tilde{c}_j(\underline{y})\right) = 0$. Also, recall that $\delta_f(\underline{x}, \mathbf{w}_{-i}) = \delta_f(\underline{y}, \mathbf{w}_{-i})$ for all $f \notin F_{\underline{x}\setminus\underline{y}}$. Therefore, (13) can be rewritten as follows

$$\begin{aligned} \Phi(\underline{x}, \mathbf{w}_{-k}) - \Phi(\underline{y}, \mathbf{w}_{-k}) = {} & 2\lambda_k(\tilde{c}_k(\underline{x}) - \tilde{c}_k(\underline{y})) \\ & + \sum_{f \in F_{\underline{x}\setminus\underline{y}}} \left[\left(\sum_{j \in \Gamma^f(\underline{x}, \mathbf{w}_{-k}, x(f))} \lambda_j \right)^2 - \left(\sum_{j \in \Gamma^f(\underline{y}, \mathbf{w}_{-k}, x(f))} \lambda_j \right)^2 \right] \\ & + \sum_{f \in F_{\underline{x}\setminus\underline{y}}} \left[\left(\sum_{j \in \Gamma^f(\underline{x}, \mathbf{w}_{-k}, y(f))} \lambda_j \right)^2 - \left(\sum_{j \in \Gamma^f(\underline{y}, \mathbf{w}_{-k}, y(f))} \lambda_j \right)^2 \right] \end{aligned} \tag{14}$$

If the NS configuration is $\underline{y}$, for a given server $x(f)$ with $f \in F_{\underline{x}\setminus\underline{y}}$, we have that

$$\sum_{j \in \Gamma^f(\underline{y}, \mathbf{w}_{-k}, x(f))} \lambda_j = \sum_{j \in \Gamma^f(\underline{x}, \mathbf{w}_{-k}, x(f))} \lambda_j - \lambda_k \tag{15}$$

as server $x(f)$ is not used in NS configuration $\underline{y}$. Similarly, when the NS configuration is $\underline{x}$, server $y(f)$ with $f \in F_{\underline{x}\setminus\underline{y}}$ is such that

$$\sum_{j \in \Gamma^f(\underline{x}, \mathbf{w}_{-k}, y(f))} \lambda_j = \sum_{j \in \Gamma^f(\underline{y}, \mathbf{w}_{-k}, y(f))} \lambda_j - \lambda_k \tag{16}$$

as server $y(f)$ is not used in NS configuration $\underline{x}$.

Accordingly, from (4), (15) and (16), we rewrite (14) as

$$\begin{aligned}\Phi(\underline{x},\mathbf{w}_{-k})-\Phi(\underline{y},\mathbf{w}_{-k}) &= 2\lambda_k(\tilde{c}_k(\underline{x})-\tilde{c}_k(\underline{x}))\\ &+\sum_{f\in F_{\underline{x}\setminus\underline{y}}}\left[\left(\sum_{j\in\Gamma^f(\underline{x},\mathbf{w}_{-k},x(f))}\lambda_j\right)^2-\left(\sum_{j\in\Gamma^f(\underline{x},\mathbf{w}_{-k},x(f))}\lambda_j-\lambda_k\right)^2\right]\\ &+\sum_{f\in F_{\underline{x}\setminus\underline{y}}}\left[\left(\sum_{j\in\Gamma^f(\underline{y},\mathbf{w}_{-k},y(f))}\lambda_j-\lambda_k\right)^2-\left(\sum_{j\in\Gamma^f(\underline{y},\mathbf{w}_{-k},y(f))}\lambda_j\right)^2\right] \qquad (17)\end{aligned}$$

Finally, by expanding the quadratic terms in (17) and simplifying them, we obtain

$$\begin{aligned}\Phi(\underline{x},\mathbf{w}_{-k})-\Phi(\underline{y},\mathbf{w}_{-k}) &= 2\lambda_k(\tilde{c}_k(\underline{x})-\tilde{c}_k(\underline{x}))+2\lambda_k\sum_{f\in F_{\underline{x}\setminus\underline{y}}}\left[\delta_f(\underline{x},\mathbf{w}_{-k})-\delta_f(\underline{y},\mathbf{w}_{-k})\right]\\ &= 2\lambda_k\left(\mathcal{C}_k(\underline{x},\mathbf{w}_{-k})-\mathcal{C}_k(\underline{y},\mathbf{w}_{-k})\right) \qquad (18)\end{aligned}$$

which satisfies (8) with $b_k = 2\lambda_k$ for all $k \in \mathcal{N}$. Therefore, the proof is concluded. ■

From Theorem 1, we get the Corollary 1.

Corollary 1. *The game $\mathcal{G}$ admits at least one NE. The price of anarchy $\mathrm{PoA}(\mathcal{G})$ of game $\mathcal{G}$ is upper bounded by $\frac{1}{2}(3+\sqrt{5})$.*

Proof: To prove Corollary 1, it suffices to note that finite potential games always admit at least one NE [37, 41]. From [42], we have that the PoA of weighted congestion games with unsplittable flows and affine player-specific cost functions of the form $c_{i,v,f}(\xi) = a_{i,v,f}\xi + b_{i,v,f}$ is equal to

$$\mathrm{PoA}(\mathcal{G}) = \frac{\Delta(\mathcal{G})+2+\sqrt{\Delta(\mathcal{G})(\Delta(\mathcal{G})+4)}}{2\Delta(\mathcal{G})} \qquad (19)$$

where $a_{i,v,f}$ and $b_{i,v,f}$ are user and resource-specific parameters, while $\Delta(\mathcal{G})$ is a game-specific parameter derived in [42]. Let us notice that, from (7), we have that $c_{i,v,f}$ is an affine function in the congestion experienced by player flows.

From [42], we have that $\Delta(\mathcal{G}) = \max_{v\in\mathcal{V},f\in\mathcal{F},i,j\in\mathcal{N}}\{a_{i,v,f}/a_{j,v,f}\}$. Accordingly, since $a_{i,v,f} = a_{j,v',f'} = 1$ for all $i,j \in \mathcal{N}$, $v,v' \in \mathcal{V}$ and $f,f' \in \mathcal{F}$, we have that $\Delta(\mathcal{G}) = 1$. Therefore, we obtain that the PoA of the game $\mathcal{G}$ is upper bounded by $\frac{1}{2}(3+\sqrt{5})$, which corresponds to the exact PoA of weighted congestion games with unsplittable flows derived in [43]. ■

From Corollary 1, it follows an important result. In fact, it guarantees the existence of a pure strategy NE, and also provides a worst-case performance upper-bound on the achievable performance of the system. While it has been shown that many weighted congestion games have unbounded PoA [44], Corollary 1 shows that the PoA of game $\mathcal{G}$ is always upper-bounded by a fixed and finite constant factor. As we will show in the numerical analysis, however, the PoA of game $\mathcal{G}$ is lower than this bound, and near-optimal performance can be achieved.

Figs.

Algorithm 1 Unilateral Service Chain Selection

Inputs: Player $i \in \mathcal{N}$, $\mathcal{V}, \mathbf{p}, \mathbf{D}^{(v-v)}, \mathbf{D}^{(in-v)}, \mathbf{D}^{(v-out)}, \boldsymbol{\delta}^A$ and the initial configuration $\underline{w}_i$;

find shortest path : $\underline{w}_i^* \leftarrow$ `findShortestPath` $(\mathcal{V}, \mathbf{p}, \mathbf{D}^{(v-v)}, \mathbf{D}^{(in-v)}, \mathbf{D}^{(v-out)}, \boldsymbol{\delta}^A)$;

if User i is unsatisfied, i.e., $\underline{w}_i^* \neq \underline{w}_i$ **then**

update the NS configuration : $\underline{w}_i \leftarrow \underline{w}_i^*$;

notify service chain update;

return

V. Algorithmic Implementation

Even though the existence of a pure strategy NE is an interesting and important result by itself, how to reach such equilibrium point in a distributed and efficient way is still an unanswered question. Accordingly, in this section we illustrate a simple, low-complexity and fully-distributed algorithm which provably converges to a NE of the game $\mathcal{G}$. Specifically, we derive a closed-form upper-bound on the number of iterations needed to reach the equilibrium, and discuss the computational complexity of the proposed distributed algorithm.

To provide the network with a distributed service chain composition mechanism, we will rely on the following lemma.

Lemma 1. *[41, Lemma 2.3] Every finite ordinal potential game has the Finite Improvement Property (FIP).*

The FIP is a property of those games where players are allowed to unilaterally update their strategy by selecting a better one. A sequence of those unilateral strategy updates is called an *improvement path*. Lemma 1 ensures that, starting from any initial strategy profile, every improvement path is finite, i.e., the number of unilateral strategy updates is finite. Therefore, Lemma 1 guarantees that the improvement path always converges to a NE of game $\mathcal{G}$ in a finite amount of iterations.

In many networking applications such as the one we are considering in this paper, to fast converge to an equilibrium point is of extreme importance. While the general theory of [41] shows that convergence to a NE is obtained in a finite amount of iterations, it does not provide any result on the number of iterations needed to reach the NE. Accordingly, in Algorithm 1 we present a low-complexity and distributed unilateral service-chain composition mechanism; then, we investigate its computational complexity and show that its convergence to a NE can be guaranteed in polynomial time if some mild conditions are satisfied.

To this purpose, let us consider player $i \in \mathcal{N}$, and let $\mathbf{w} = (\underline{w}_i, \mathbf{w}_{-i})$ be the current NS configuration. Furthermore, let use define $\mathbf{p} = (p_{v,f})_{v \in \mathcal{V}, f \in \mathcal{F}}$ and $\boldsymbol{\delta}^A = (\delta^A_{v,f})_{v \in \mathcal{V}, f \in \mathcal{F}}$, where

$$\delta^A_{v,f} = \sum_{j \in \Gamma^f(\mathbf{w}, v)} \lambda_j \tag{20}$$

represents the congestion level for each function on each server under NS configuration $\mathbf{w}$.

Algorithm 1 is executed independently by each NSB in the network. Specifically, each NSB executes a unilateral strategy update that corresponds to a unilateral service chain selection. To exploit the FIP, we assume that only one

NSB is allowed to update its NS configuration at any given iteration. Therefore, unilateral service chain composition problem results in finding the NS configuration that guarantees the minimum cost when congestion levels on each server are given by $\boldsymbol{\delta}^A$. This latter problem can be modeled as a single-source single-destination shortest path problem. Since our model is F-layered and has an acyclic structure (cfr. Fig. 3(a)), the network can be transformed into a directed acyclic graph (DAG) and the shortest path problem can be solved by traditional dynamic programming techniques. Accordingly, the shortest path problem in Algorithm 1 is solved by the method `findShortestPath()`. Specifically, this method tranforms the network into a DAG $G = (V_G, E_G)$, with $V_G = V \cdot F$ vertices and $E_G = 2V + V^2(F-2)$ edges, where edge costs are generated according to (7), in which δ_f is substituted by elements in $\boldsymbol{\delta}^A$. Furthermore, it is well-known that the complexity of the shortest path problem on DAGs through dynamic programming is $\mathcal{O}(E_G + V_G)$. Therefore, in our case, finding the shortest path has complexity $\mathcal{O}(V^2F)$.

In Proposition 1, we provide useful convergence properties for Algorithm 1.

Proposition 1. *For any initial NS configuration, any improvement path whose improvement steps are generated according to Algorithm 1 converges in a finite amount of iterations to a NE. In more detail, the computational complexity of Algorithm 1 is $\mathcal{O}(N^2F^2V^3)$, i.e., the complexity of the algorithm is polynomial.*

Proof: For the sake of illustration, and without loss in generality, let us assume that all system parameters are integer numbers. Let $i \in \mathcal{N}$ be the player which performs a unilateral improvement step, and let $(\underline{w}_i, \mathbf{w}_{-i})$ be an initial NS configuration such that there exists $\underline{\omega} \in \mathcal{W}$ such that $\mathcal{C}_i(\underline{\omega}, \mathbf{w}_{-i}) < \mathcal{C}_i(\underline{w}_i, \mathbf{w}_{-i})$, i.e., player i is unsatisfied with the current NS configuration and wants to update its strategy. To this purpose, it runs Algorithm 1 and calculates the shortest path $\underline{w}_i^*$. Note that any unilateral NS configuration update generated by Algorithm 1 produces a $\underline{w}_i^*$ which is the best response, i.e., $\underline{w}_i^* = \arg\min_{\underline{\omega} \in \mathcal{W}} \mathcal{C}_i(\underline{\omega}, \mathbf{w}_{-i})$. After the unilateral improvement step generated by Algorithm 1, the cost of player i has to decrease at least by 1. Accordingly, from (8), and having in mind that $b_i = 2\lambda_i$, we have that

$$\Phi(\underline{w}_i^*, \mathbf{w}_{-i}) - \Phi(\underline{w}_i, \mathbf{w}_{-i}) \leq -2\lambda_i \tag{21}$$

that is, each unilateral improvement step contributes to a reduction of at least $2\lambda_i$ in the potential function.

From (9), it can be shown that, for any possible NS configuration $\mathbf{w} \in \mathcal{W}^N$, the potential function $\Phi(\mathbf{w})$ is always upper-bounded by

$$\Phi(\mathbf{w}) \leq 2NF\lambda_{\max} c_{\max} + FVN^2\lambda_{\max}^2 \tag{22}$$

where $\lambda_{\max} = \max_{i \in \mathcal{N}} \lambda_i$ and $c_{\max} = \max_i(\max_{\underline{w}_i \in \mathcal{W}}(\tilde{c}_i(\mathbf{w})))$ which, by maximizing the right-hand side, leads to

$$\begin{aligned} \Phi(\mathbf{w}) &\leq 2NF\lambda_{\max}(c_{\max} + NV\lambda_{\max}) \\ &\leq 2NF\lambda_{\max}^2 V(c_{\max} + 1) \end{aligned} \tag{23}$$

From (9), we have that $\Phi(\mathbf{w}) \geq 0$. Also, from (21) we know that any unsatisfied NSB i that executes Algorithm 1 will reduce the potential by a factor higher than or equal to $2\lambda_{\min}$, where $\lambda_{\min} = \min_{i \in \mathcal{N}} \lambda_i$. Therefore, the maximum number of unilateral improvement steps is

$$\frac{2NF\lambda_{\max}^2 V(c_{\max} + 1)}{2\lambda_{\min}}$$

It follows that the maximum number of improvement steps is $\mathcal{O}(N^2FV\frac{\lambda_{\max}^2}{\lambda_{\min}})$. In general, both $\lambda_{\max}$ and $\lambda_{\min}$ do not depend on the number of players, servers or functions in the system. By combining this latter result with the complexity $\mathcal{O}(V^2F)$ of the method `findShortestPath()`, we obtain that the overall complexity of Algorithm 1 is $\mathcal{O}(N^2F^2V^3)$. ■

It is worth noting that Algorithm 1 relies on information that is publicly available (i.e., $\mathcal{V}, \mathbf{p}, \mathbf{D}^{(v-v)}$) or private (or local) parameters (i.e., $\mathbf{D}^{(in-v)}, \mathbf{D}^{(v-out)}$). The only information that has to be revealed to the NSBs to execute the algorithm is $\boldsymbol{\delta}^A$. Accordingly, $\boldsymbol{\delta}^A$ is broadcast by the Telco Operator to all players in the network after each improvement step. But, more importantly, the identity and the NS configuration of other players in the network cannot be directly extracted from $\boldsymbol{\delta}^A$. Therefore, since Algorithm 1 does not require any communication among players, it can be implemented in a fully-distributed and privacy-preserving way.

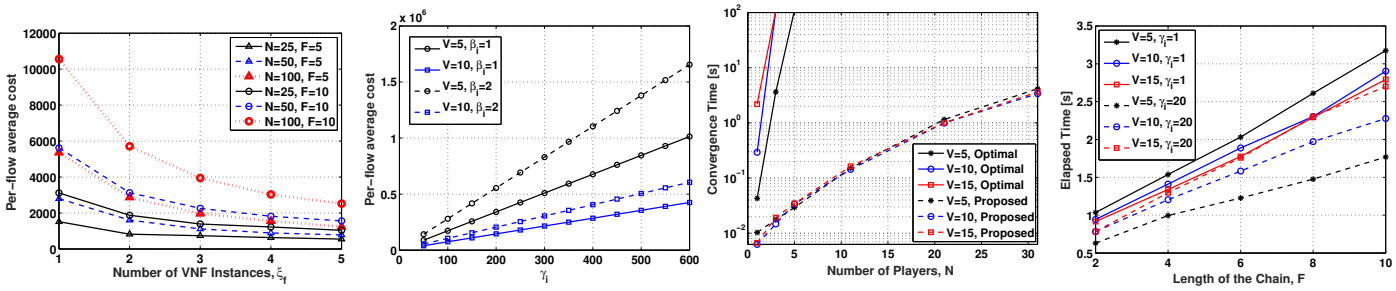
Moreover, let us notice that Algorithm 1 is similar to the Nashify algorithm proposed in [34]. However, while Algorithm 1 is distributed, Nashify is not intended to be implemented in a distributed way as it requires full knowledge of the rates λ_i generated by all NSBs. Accordingly, to directly apply Nashify to the scenario we are considering in this paper is not feasible.

The main steps of the proposed distributed service chain composition mechanism based on congestion games are as follows:

1) The Network Orchestrator broadcasts $(\mathcal{V}, \mathbf{p}, \mathbf{d}, \delta^A)$ to all NSBs. Instead, $\mathbf{D}^{(in-v)}$ and $\mathbf{D}^{(v-out)}$ are sent individually to each NSB $i \in \mathcal{N}$. At the beginning, no NS configurations have been selected. Accordingly, each $\delta^A_{v,f} \in \boldsymbol{\delta}^A$ is set to zero, i.e., $\delta^A_{v,f} = 0$;
2) All NSBs simultaneously execute Algorithm 1 and find their own NS configuration given that $\delta^A_{v,f} = 0$ for all $v \in \mathcal{V}$ and $f \in \mathcal{F}$. The selected NS configurations are sent to the Network Orchestrator;
3) The Network Orchestrator updates $\boldsymbol{\delta}^A$ according to the NS configurations received from the NSB, and broadcasts it to all NSBs;
4) Each unsatisfied NSB executes Algorithm 1 again. Accordingly, it unilaterally updates its NS configuration and sends the new configuration to the Network Orchestrator;
5) Steps 3-4 are repeatedly executed until all NSBs are satisfied.

VI. Numerical Results

In this section, through extensive numerical results, we assess the performance of the congestion game proposed to solve the distributed service chain composition problem. In the following, and unless stated otherwise, for each flow managed by NSB $i \in \mathcal{N}$ we assume a transmission data rate $\lambda_i = 10$ Mbit/s and $\beta_i = 1$. The number of functions in the chain is set to $F = 5$, and the price parameter $p_{v,f}$ is modeled as a uniformly distributed variable that takes integer values in the interval $[1, 100]$. Finally, the latency parameters in the $\mathbf{D}^{(v-v)}$, $\mathbf{D}^{(in-v)}$ and $\mathbf{D}^{(v-out)}$ matrices are generated according to a Gamma distribution with mean value of 8 ms and variance equal to 0.004 ms. All the results presented in the following are averaged over 100 simulation runs.



(a) (b) (c) (d)

Fig. 4. a) Per-flow average cost $\tilde{\mathcal{C}}(\mathbf{w})$ as a function of the number ξ_f of VNF instances; b) Per-flow average cost $\tilde{\mathcal{C}}(\mathbf{w})$ as a function of γ_i; c) Scalability of the proposed solution versus the optimal one w.r.t. the number N of players; b) Scalability of the proposed solution w.r.t. the number F of VNFs composing the service chain.

Accordingly, the performance of the system is investigated in Section VI-A. In Section VI-B, the convergence rate and the scalability of Algorithm 1 are evaluated. The adaptability and dynamic behavior of the algorithm are assessed in Section VI-C, and the PoA of the proposed solution is discussed in Section VI-D.

A. *Performance Evaluation*

To investigate the impact of the number of VNF instances on the performance of the system, we consider $\xi_f \in [1, V]$ as a parameter that represents the number of VNF instances for each function $f \in \mathcal{F}$ that are instantiated in the network. In more detail, $\xi_f = 1$ means that VNF f is instantiated on only one server. Instead, $\xi_f = V$ implies that all servers provide VNF f. In the following, we assume $V = 5$, and we consider two scenarios where $F = 5$ and $F = 10$; also we assume that $\xi_{f'} = \xi_{f''}$ for all $f', f'' \in \mathcal{F}$. Accordingly, in Fig. 4(a) we show the per-flow average cost as a function of $\xi_f \in [1, V]$ for different values of the number N of flows and the number V of servers in the network. Specifically, for any given NS configuration $\mathbf{w}$, the per-flow average cost is defined as follows:

$$\tilde{\mathcal{C}}(\mathbf{w}) = \frac{1}{N} \sum_{j \in \mathcal{N}} \mathcal{C}_j(\mathbf{w}) \tag{24}$$

By increasing the number of instances of the same VNF, NSBs are able to select different VNF Servers, which leads to lower congestion levels. Instead, if the number of instances is small, NSBs are forced to select the same server, thus increasing the congestion level on that server. Therefore, in Fig. 4(a) it is shown that the per-flow average cost decreases as ξ_f increases. It is worth noting that the cost increases when the number F of functions in the chain increases. In fact, a higher number of functions leads to longer paths in the network that eventually results in higher costs $c_i^{(\mathcal{T})}$ and $c_i^{(\mathcal{P})}$. Furthermore, Fig. 4(a) shows that an increase in the number N of NSB flows also causes an increase in the congestion level $c_i^{(\mathcal{C})}$, which also corresponds to higher experienced costs.

In Fig. 4(b), we show the per-flow average cost $\tilde{\mathcal{C}}(\mathbf{w})$ as a function of γ_i for different numbers V of servers, when $\beta_i = 1$ and $\beta_i = 2$. From (6), we have that $\tilde{\mathcal{C}}(\mathbf{w})$ linearly increases as γ_i increases and, as expected, it also increases as β_i increases as well. Instead, it is interesting to note that, by increasing the number V of servers, the average cost decreases. In fact, by increasing the number of servers, NSBs are able to select different servers, thus lowering the congestion level, and consequently reducing the corresponding average cost.

B. Convergence and Scalability Analysis

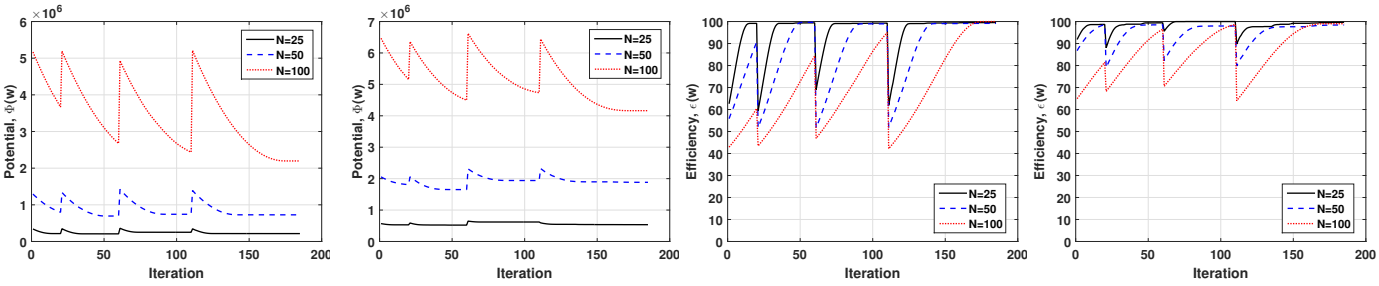
In Fig. 4(c), we compare the scalability of the proposed algorithm with that of the optimal solution as a function of the number N of NSB flows for different values of available servers V. It is shown that the time elapsed to find the optimal solution is large, and exponentially increases with the number of players. Instead, our proposed algorithm well scales with the number N of players and allows the system to reach a NE in few seconds. Specifically, as shown in Proposition 1, the scalability w.r.t. the number N of players is $\mathcal{O}(N^2)$.

Fig. 4(d) shows the scalability of the proposed solution versus the optimal one w.r.t. the number F of VNFs composing the service chain for different values of both N and V. Similarly to what we have shown in Fig. 4(c), Fig. 4(d) shows that the proposed approach well scales with the number F of VNFs that compose the service chain. Specifically, we have that for $\gamma_i = 1$, NSBs equally weighs all the contributions in (6). Instead, when $\gamma_i = 20$, the contribution generated by the two terms $c_i^{(\mathcal{P})}(\underline{w}_i)$ and $c_i^{(\mathcal{T})}(\underline{w}_i)$ is more relevant as compared to the congestion cost $c_i^{(\mathcal{C})}(\underline{w}_i, \mathbf{w}_{-i})$. Accordingly, when γ_i is large, NSBs are not much influenced by the decisions taken by others because the dependence of the decisions taken by the player i on the behavior of the other players is present only in the term $c_i^{(\mathcal{C})}(\underline{w}_i, \mathbf{w}_{-i})$. Instead, if γ_i is small, the cost function $\mathcal{C}_i(\underline{w}_i, \mathbf{w}_{-i})$ mainly depends on $\underline{w}_i$. Accordingly, for large values of the γ_i parameters, convergence to the NE is attained in a shorter amount of time. Specifically, from Proposition 1 we have that the convergence rate w.r.t. the number F of VNFs is $\mathcal{O}(F^2)$. However, in realistic scenarios, the number F of functions in the chain is not large. It means that in many real-world scenarios the impact of F on the overall complexity is low if compared to that of V and N.

C. Dynamic Behavior Analysis

To investigate the adaptability of the proposed algorithm, in Figs. 5(a) and 5(b), we illustrate its dynamic behavior by showing the evolution of the potential function $\Phi(\mathbf{w})$ for different network configurations and values of the γ_i parameter. We assume that the chain is composed by $F = 5$ VNFs and each server only provides $\xi_f = 3$ different VNF instances. For illustrative purposes, we consider a worst-case scenario where all VNF instances are migrated from a VNF Server to another one. Accordingly, at iteration indexes $t = \{20, 40, 50, 75\}$ we randomly migrate all VNF instances among the available VNF Servers.

Fig. 5(a) shows that our proposed algorithm is able to rapidly adapt to network changes in a few iterations and converges to the NE of the game which is attained when the potential converges to a minimum. Both Figs. 5(a) and 5(b) show that the convergence rate for $\gamma_i \in \{1, 5\}$ is fast when the number of players in the network is small. Instead, if the number of players is large, i.e., $N = 100$, Fig. 5(a) shows that such convergence is faster when the value of the γ_i parameter is large. On the other hand, Fig. 5(b) illustrates how smaller values of γ_i slow down the convergence of the algorithm. This result has already been identified in Fig. 4(d). That is, higher values of γ_i make the impact of congestion on the experienced cost of less importance if compared to that of latencies and prices, thus reducing the number of interactions among players which results in a higher convergence rate. In any case, as shown by Figs. 5(a) and 5(b), the convergence towards the NE is slow when the number N of players is large. This stems from the fact that the complexity of the proposed distributed algorithm is $\mathcal{O}(N^2F^2V^3)$. Therefore, if the VNF instances are migrated very frequently, the algorithm might not be able to reach the NE in large-scale networks. However, in the following we



(a) $\gamma_i = 1$ (b) $\gamma_i = 5$ (c) $\gamma_i = 1$ (d) $\gamma_i = 5$

Fig. 5. a-b) Dynamic evolution of the potential function $\Phi(\mathbf{w})$; c-d) Dynamic evolution of the potential efficiency function $\epsilon(\mathbf{w}(j))$.

show that the proposed algorithm is anyway able to provide an efficient NS configuration, even if a NE has not been successfully reached.

Let us define the *potential efficiency* function ϵ as follows:

$$\epsilon(\mathbf{w}(j)) = \frac{\Phi(\mathbf{w}^{\mathrm{NE}})}{\Phi(\mathbf{w}(j))} \tag{25}$$

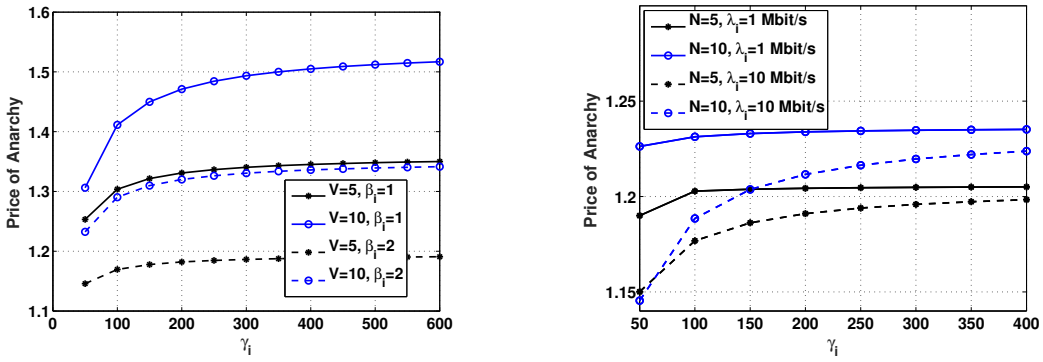
where $\underline{w}(j)$ and $\underline{w}^{\mathrm{NE}}$ are the NS configurations at iteration j and at the NE, respectively. The value of $\epsilon(\mathbf{w}(j))$ at iteration j represents the efficiency of the current NS configuration $\mathbf{w}(j)$. High values mean that the current NS configuration is highly efficient and the achieved performance are near-optimal. On the contrary, low values indicate poor performance.

In Figs. 5(c) and 5(d), we show the dynamic evolution of the potential efficiency function $\epsilon(\mathbf{w}(j))$ when $\underline{w}(j)$ and $\underline{w}^{\mathrm{NE}}$ are calculated as in Figs. 5(a) and 5(b). Although, in some cases, the NE is not reached due to the frequent VNF migration, the results obtained show that the algorithm achieves high efficiency after a few iterations even when N is large. For example, in Fig. 5(d) it is shown that the algorithm achieves an efficiency of 80% at iteration $j = 20$, i.e., when the VNF instances are migrated. It is worth noting that the complexity of the algorithm when $N = 100$ is $\mathcal{O}(N^2) = \mathcal{O}(10^4)$, therefore in the worst case the algorithm would reach the NE in ten thousands iterations. In conclusion, Figs. 5(c) and 5(d) show that, although the convergence towards the NE is not possible in some cases, a few tens of iterations suffice to reach an efficient NS configuration.

D. PoA Analysis

In this section, we investigate the impact of system parameters on the PoA of the game $\mathcal{G}$. In Fig. 6(a), we show the PoA when $N = 10$ as a function of γ_i for different values of V, when $\beta_i = 1$ and $\beta_i = 2$. It is shown that the PoA increases as the value of γ_i increases. However, it is worth noting that the PoA does not linearly increases, but it exhibits a concave behavior that asymptotically converges to some value smaller than the upper-bound derived in Corollary 1. Furthermore, it also increases when a higher number of servers is considered.

Finally, in Fig. 6(b) we show the PoA when $V = 5$ as a function of γ_i for different values of N, when $\lambda_i = 1$ Mbit/s and $\lambda_i = 10$ Mbit/s. Similarly to what has been shown in Fig. 6(a), the PoA is an increasing function of γ_i and



(a) PoA for different values of V and β_i

(b) PoA for different values of N and λ_i

Fig. 6. Price of Anarchy as a function of γ_i.

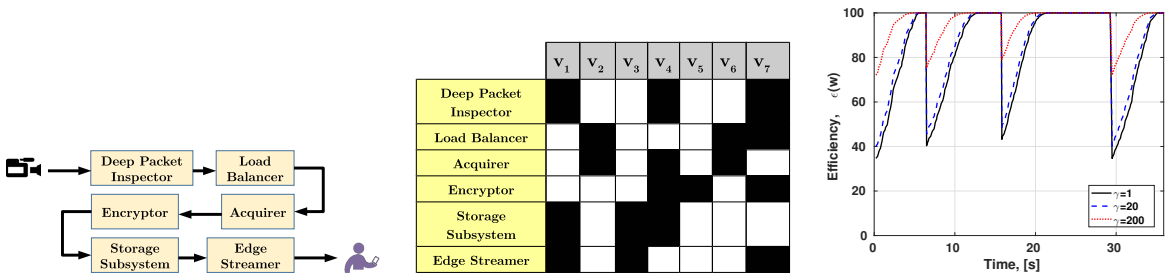
converges to an horizontal asymptote. Furthermore, it also increases with the number of players while it decreases as the data rate λ_i decreases.

VII. Experimental Analysis

To assess and validate the proposed framework, we have considered the SDN/NFV video broadcasting platform described in [45] as a case study, and implemented it in Planetlab [46] as a proof-of-concept. The considered platform enables big/small live streaming content providers to transmit video flows to either fixed or mobile users, with no need to deploy a dedicated and expensive data delivery infrastructure. Each flow generated by any content provider is intercepted by its access node, and routed towards the video receiver through a service chain like the one shown in Fig. 7(a). The service chain consists of $F = 6$ VNFs, whose functionalities are detailed in [45].

The implemented testbed has been realized by using a Planetlab slice of $V = 7$ remotely dislocated nodes working as VNF Servers, and a number $\xi_f = 3$ of simultaneous VNF instances that are instantiated on the available VNF Servers. The network has been loaded with $N = 80$ flows, specifically 30 flows with bit rate $\lambda_i = 300$ kbit/s, 30 with bit rate $\lambda_i = 400$ kbit/s, and 20 with bit rate $\lambda_i = 500$ kbit/s. For the sake of simplicity, we assumed symmetric users w.r.t. the γ_i parameter, i.e., $\gamma_i = \gamma$ for all $i \in \mathcal{N}$. In order to verify the responsiveness of the proposed algorithm, we have created some discontinuities in the network behavior, shutting down one VNF Server at $t = 7$ seconds, another VNF Server at $t = 16$ seconds, and then turning them on again at $t = 29$ seconds. The VNF instances running in each VNF Server are listed in Fig. 7(b), and are indicated by black entries. In Fig. 7(c), we show the dynamic evolution of the potential efficiency function $\epsilon(\mathbf{w}(j))$ for different values of the parameter γ. It is shown that the algorithm converges towards the NE in a few seconds every time the servers are shut down or turned on again. Also, the algorithm converges faster as larger values of γ are considered.

Although the implemented testbed is far from covering the case study of large scale networks, the obtained experimental results fairly provide an insightful glimpse of how the proposed algorithm can effectively achieve service chaining in NFV networks.



(a) (b) (c)

Fig. 7. a) The considered live streaming service chain; b) The allocation table of the considered VNFs; c) Dynamic evolution of the potential efficiency function $\epsilon(\mathbf{w}(j))$.

VIII. Conclusions

In this paper, a distributed solution to address the Service Chain Composition problem in NFV networks has been proposed. To capture competitive behavior among network users, tools from non-cooperative game theory have been exploited. Accordingly, the problem has been formulated as an atomic weighted congestion game with unsplittable flows and player-specific cost functions. It has been proved that the game is also a potential game and admits a weighted potential function whose closed form equation has been derived. By exploiting properties of potential games, the existence of a NE has been demonstrated. An algorithmic implementation of the proposed solution has been discussed, and it has been shown that the proposed algorithm can be implemented in a distributed and privacy-preserving way. Furthermore, its scalability has been studied by proving that it converges to a NE in polynomial time. The efficiency of the proposed solution has been investigated by showing that the PoA is upper-bounded. Finally, numerical and experimental results have assessed the performance and efficiency of the proposed game theoretic distributed solution.

References

[1] A. Manzalini, R. Saracco, C. Buyukkoc, P. Chemouil, S. Kukliński, A. Gladisch, M. Fukui, W. Shen, M. Fujiwara, K. Shimano *et al.*, "Software-defined networks for future networks and services: main technical challenges and business implications," *IEEE SDN4FNS, White Paper, January*, 2014.

[2] Y. Jiang, J. Lan, Z. Wang, and Y. Deng, "Embedding and reconfiguration algorithms for service aggregation in network virtualization," *International Journal of Communication Systems*, vol. 29, no. 1, pp. 33–46, 2016.

[3] P. Wang, J. Lan, X. Zhang, Y. Hu, and S. Chen, "Dynamic function composition for network service chain: Model and optimization," *Computer Networks*, vol. 92, pp. 408–418, 2015.

[4] Network Functions Virtualisation – Introductory White Paper, White paper, Oct 2012.

[5] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," 2015.

[6] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *Access, IEEE*, vol. 3, pp. 2542–2553, 2015.

[7] B. Anwer, T. Benson, N. Feamster, D. Levin, and J. Rexford, "A slick control plane for network middleboxes," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 147–148.

[8] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul, "Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, 2014, pp. 543–546.

[9] The Programmable Cloud Network: Primer on SDN and NFV. Alcatel Lucent white paper.

[10] The real-time Cloud: Combining Cloud, NFV, and service provider SDN. Ericson white paper.

[11] Cisco Open Network Environment: Adaptable Framework for the Internet of Everything. Cisco white paper.

[12] W. Rankothge, J. Ma, F. Le, A. Russo, and J. Lobo, "Towards making network function virtualization a cloud computing service," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*. IEEE, 2015, pp. 89–97.

[13] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "Simple-fying middlebox policy enforcement using sdn," in *ACM SIGCOMM computer communication review*, vol. 43, no. 4. ACM, 2013, pp. 27–38.

[14] SDxCentral's 2015 SDN and NFV Market Size and Forecast Report, 2015.

[15] ETSI NVF GS, Network Functions Virtualization (NFV): Architectural framework, NFV 002 V1.1.1, Oct 2013.

[16] ETSI NFV GS, Network Function Virtualization (NFV) Management and Orchestration, NFV-MAN 001 v0.8.1, Nov 2014.

[17] ETSI NVF GS, Network Functions Virtualization (NFV): Use cases, NFV 001 V1.1.1, Oct 2013.

[18] T. Nadeau and P. Quinn, "Problem Statement for Service Function Chaining," RFC 7498, Nov. 2015.

[19] M. Naor, B. Pinkas, and R. Sumner, "Privacy preserving auctions and mechanism design," in *Proceedings of the 1st ACM conference on Electronic commerce*. ACM, 1999, pp. 129–139.

[20] B. Batouche, Y. Naudet, and F. Guinand, "Semantic web services composition optimized by multi-objective evolutionary algorithms," in *Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on*, May 2010, pp. 180–185.

[21] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani, "An approach for qos-aware service composition based on genetic algorithms," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. ACM, 2005, pp. 1069–1075.

[22] S. Sahhaf, W. Tavernier, M. Rost, S. Schmid, D. Colle, M. Pickavet, and P. Demeester, "Network service chaining with optimized network function embedding supporting service decompositions," *Computer Networks*, vol. 93, pp. 492–505, 2015.

[23] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint vm placement and routing for data center traffic engineering," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 2876–2880.

[24] H. Moens and F. De Turck, "Vnf-p: A model for efficient placement of virtualized network functions," in *Network and Service Management (CNSM), 2014 10th International Conference on*. IEEE, 2014, pp. 418–423.

[25] A. Blenk, A. Basta, and W. Kellerer, "Hyperflex: An SDN virtualization architecture with flexible hypervisor function allocation," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 397–405.

[26] A. Fischer, J. F. Botero, M. Till Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *Communications Surveys & Tutorials, IEEE*, vol. 15, no. 4, pp. 1888–1906, 2013.

[27] G. Cheng, H. Chen, H. Hu, Z. Wang, and J. Lan, "Enabling network function combination via service chain instantiation," *Computer Networks*, vol. 92, pp. 396–407, 2015.

[28] A. Lombardo, A. Manzalini, V. Riccobene, and G. Schembra, "An analytical tool for performance evaluation of software defined networking services," *Proc. of IEEE SDNMO, Krakow, Poland*, 2014.

[29] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*. IEEE, 2014, pp. 7–13.

[30] R. Guerzoni, R. Trivisonno, I. Vaishnavi, Z. Despotovic, A. Hecker, S. Beker, and D. Soldani, "A novel approach to virtual networks embedding for sdn management and orchestration," in *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, 2014, pp. 1–7.

[31] T. Lukovszki, M. Rost, and S. Schmid, "It's a match!: Near-optimal and incremental middlebox deployment," *ACM SIGCOMM Computer Communication Review*, vol. 46, no. 1, pp. 30–36, 2016.

[32] D. Fotakis, S. Kontogiannis, and P. Spirakis, "Selfish unsplittable flows," *Theoretical Computer Science*, vol. 348, no. 2, pp. 226–239, 2005.

[33] M. Mavronicolas, I. Milchtaich, B. Monien, and K. Tiemann, "Congestion games with player-specific constants," in *Mathematical Foundations of Computer Science 2007*. Springer, 2007, pp. 633–644.

[34] P. N. Panagopoulou and P. G. Spirakis, "Algorithms for pure nash equilibria in weighted congestion games," *Journal of Experimental Algorithmics (JEA)*, vol. 11, pp. 2–7, 2007.

[35] R. Feldmann, M. Gairing, T. Lücking, B. Monien, and M. Rode, "Nashification and the coordination ratio for a selfish routing game," in *Automata, Languages and Programming*. Springer, 2003, pp. 514–526.

[36] ETSI NVF GS, Network Functions Virtualization (NFV) Infrastructure Overview, NFV-INF 001 V1.1.1, Jan 2015.

[37] R. W. Rosenthal, "A class of games possessing pure-strategy nash equilibria," *International Journal of Game Theory*, vol. 2, no. 1, pp. 65–67, 1973.

[38] C. Tekin, M. Liu, R. Southwell, J. Huang, and S. H. A. Ahmad, "Atomic congestion games on graphs and their applications in networking," *Networking, IEEE/ACM Transactions on*, vol. 20, no. 5, pp. 1541–1552, 2012.

[39] E. Altman, T. Boulogne, R. El-Azouzi, T. Jiménez, and L. Wynter, "A survey on networking games in telecommunications," *Computers &*

Operations Research, vol. 33, no. 2, pp. 286–311, 2006.

[40] M. Liu and Y. Wu, "Spectum sharing as congestion games," in *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*. IEEE, 2008, pp. 1146–1153.

[41] D. Monderer and L. S. Shapley, "Potential games," *Games and economic behavior*, vol. 14, no. 1, pp. 124–143, 1996.

[42] M. Gairing, B. Monien, and K. Tiemann, "Routing (un-) splittable flow in games with player-specific linear latency functions," in *Automata, Languages and Programming*. Springer, 2006, pp. 501–512.

[43] B. Awerbuch, Y. Azar, and A. Epstein, "The price of routing unsplittable flow," in *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*. ACM, 2005, pp. 57–66.

[44] S. Kontogiannis and P. Spirakis, "Atomic selfish routing in networks: A survey," in *Internet and Network Economics*. Springer, 2005, pp. 989–1002.

[45] V. D'Amico, A. Lombardo, M. Melita, C. Rametta, and G. Schembra, "An SDN/NFV telco operator platform for video broadcasting," *to appear in Communications Magazine, IEEE*, 2016.

[46] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: an overlay testbed for broad-coverage services," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 3–12, 2003.