Computer Networks 250 (2024) 110571

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Mobility-aware SFC migration in dynamic 5G-Edge networks

Juan Lucas Vieira[a,*], Evandro L.C. Macedo[b], Anselmo L.E. Battisti[a], Julia Noce[c], Paulo F. Pires[c], Débora C. Muchaluat-Saade[a], Ana C.B. Oliveira[c], Flavia C. Delicato[a]

[a] *MídiaCom Laboratory, Universidade Federal Fluminense, Niterói, Brazil*
[b] *High-Speed Networks Laboratory, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil*
[c] *Dell Research Brazil, Rio de Janeiro, Brazil*

ARTICLE INFO

Keywords:
Service function chain
Virtual network function
Migration
Edge computing
User mobility

ABSTRACT

The fifth-generation mobile networks bring an evolution in the development of novel classes of applications, increasing the demand for performance and flexibility to support new classes of services. Edge Computing and Network Function Virtualization are two key technologies that bring data processing closer to the user and allow the replacement of traditional hardware-based network functions with virtualized counterparts, which reduces costs and permits the provision of low-latency services. In this scenario, services are often implemented as a Service Function Chain (SFC), consisting of a set of ordered Virtual Network Functions (VNFs) that provide the required service functionality. During service execution, users can move from one location to another, which impacts service performance. An approach to reduce the impact of user mobility is to migrate services as users move to different places. However, the SFC migration problem imposes several challenges that require complex decision-making. VNFs should be migrated to nodes that have the necessary resources to execute the VNF instances. Also, the migration should be fast and effective, using as few resources as possible to reduce migration costs. In this paper, we propose a proactive and iterative SFC migration strategy to address the performance impact caused by user mobility. Our proposed solution anticipates the movement of users and estimates the impact of their mobility on service delay, proactively triggering service migrations when needed. The proposed algorithm decides which VNFs of the SFC should be migrated and which compute nodes will host the migrating VNFs, fulfilling service, resource, and migration constraints while maintaining a low migration overhead. Our algorithm also considers the dynamic nature of the edge environment in which other SFC placement and migrations might be occurring when an SFC is being migrated, which also introduces new challenges related to the availability and concurrency of resources required for migration.

1. Introduction

The fifth generation of mobile networks (5G) increased the demand for flexibility and stringent Quality of Service (QoS) to support several new services. Several supporting technologies are needed to deal with these requirements, from which Edge Computing and Network Functions Virtualization (NFV) stand out as key emerging technologies by the European 5G PPP research body [1].

Edge computing is a paradigm where servers and devices located at the edge of the network extend the resources traditionally provided by the cloud to perform computationally intensive tasks and store large amounts of data close to the user's equipment. Such proximity makes it possible to meet low latency requirements typical of various applications supported by 5G networks. Likewise, the Network Function Virtualization (NFV) has been used as a key technology to replace hardware-based functions with a software counterpart, called Virtual Network Function (VNF), that can be instantiated on commodity hardware. The NFV technology virtualizes computing, network, and storage resources, decoupling the functions from proprietary hardware and, thus, providing the required flexibility for on-demand and low-cost service provision in 5G-Edge scenarios.

In an NFV environment, services are provided by instantiating Service Function Chains (SFCs) in the underlying infrastructure. An SFC specifies a chain of VNFs that must be traversed by the application flow in a specific order to deliver the required service functionalities. In the context of a 5G-Edge network, a user connects to services through an access node (e.g., 5G base station) and both the computing nodes

* Corresponding author.
E-mail addresses: juanlucasvieira@id.uff.br (J.L. Vieira), evandro@ravel.ufrj.br (E.L.C. Macedo), anselmo@midiacom.uff.br (A.L.E. Battisti), julia.noce@dell.com (J. Noce), paulo.pires@dell.com (P.F. Pires), debora@midiacom.uff.br (D.C. Muchaluat-Saade), ana.oliveira@dell.com (A.C.B. Oliveira), fdelicato@ic.uff.br (F.C. Delicato).

https://doi.org/10.1016/j.comnet.2024.110571
Received 31 October 2023; Received in revised form 27 May 2024; Accepted 3 June 2024
Available online 6 June 2024

— *i.e.*, nodes where VNF instances composing the SFC are running — and the access nodes can be located at the edge of the network. When providing a service, the Service Level Agreement (SLA) specifies several requirements that must be satisfied throughout the service duration to maintain good performance standards. It is worth mentioning that the SLA defines the contractual obligations, including performance metrics (*i.e.,* QoS parameters), while QoS is a broader term denoting the ability of a service to satisfy the user's needs [2]. In this context, SLA refers to the contract between the service provider and the user, and it includes, among several other information, the set of QoS parameters indicating the user performance expectations that the provider should meet. If a performance metric value for any of the defined QoS parameters is outside the desired threshold during the service provision agreement, a violation of the SLA occurs. In other words, if the QoS delivered is below the contracted, there is an SLA violation, which is subject to penalties and generally leads to user dissatisfaction with the provider.

A well-known step in the service provision under the context of NFV is the SFC Placement [3–6], which consists of finding suitable nodes to host the VNFs of an SFC, satisfying the constraints imposed by the service SLAs. Once these nodes are found, the VNFs that will compose the service can be instantiated at the selected nodes, and the service path is formed by the network links that will connect the user's access node to the service's destination node, traversing the chain of VNFs. However, in 5G networks users often consume services through mobile equipment and ongoing services are highly prone to variations when the user that is being served moves between different locations, using different access nodes. As the user moves to the coverage area of a base station different from the one connected initially and starts using another base station that has a higher latency to the nodes where the service is running, services' packets will have a greater delay to traverse the SFC and reach their destination, which might violate the maximum delay requirement established by the SLA. Therefore, it is hard to guarantee seamless service continuity and meet the service's latency requirements, as the user's current access node may become too distant or new links may have a greater delay, especially for latency-stringent classes of service. Also, resource availability at nodes and links of the underlying infrastructure might change during the service provision period. Thus, after the initial placement of an SFC, it is essential to maintain the expected QoS, which is a challenging task.

To this end, VNF migration has been widely used as a way to enhance the network resilience to counter the degradation in the QoS [7–9]. VNF migrations can be triggered by different situations: (i) for load balance purposes - VNF migration can be a useful strategy to perform load balancing by migrating VNFs from overloaded nodes to less-loaded ones belonging to the same service provider; (ii) to reduce energy consumption - VNF migration can also be employed to migrate VNFs from less-loaded nodes, allowing them to be turned off to save energy; (iii) to cope with user mobility impact in services performance - migrating VNFs when users move towards access nodes whose location is too far from the service or the reconfigured path has a higher latency, which is the main goal of this work. However, the VNF migration problem imposes several challenges that must be tackled.

First, there must be an effective decision-making process on whether the migration must be triggered. The migration decision must correctly identify the need for migration and select an appropriate recipient node. A migration path must also be selected to transfer the required data (e.g., VNF image, VNF state) from the original VNF location to the recipient node. The resources and time required to migrate a VNF must also be considered, and a balance must be taken carefully between the performance gains and the operation cost generated by the migration. All of these aspects pertain to a single VNF migration. If we consider the migration of an SFC (composed of several VNFs), additional challenges and complexity arise. Because one VNF migration impacts the decision-making of another VNF migration in the chain, the SFC migration cannot be solved as a set of individual VNF migration decisions. Rather, the SFC migration problem must be solved by considering not only the constraints of each VNF but also the requirements of the whole chain. Moreover, migrating the whole SFC can be more costly since all VNFs of the chain must be migrated. Therefore, a decision algorithm for the SFC migration must consider the total migration overhead and reduce it whenever possible.

In this work, we address the SFC Migration problem by providing a proactive and iterative migration strategy that focuses on solving performance issues caused by user mobility by migrating the fewest number of VNFs possible. Our proposed solution anticipates the movement of users and estimates the impact of their mobility on compliance with the service SLA (regarding the delay), triggering a service migration procedure when needed. When triggered, the migration algorithm decides which VNFs of the SFC should be migrated and where to migrate to, addressing the challenges imposed by the service, resource utilization, and migration constraints, maintaining the cost of migration as low as possible. Our algorithm also considers a dynamic edge environment in which concurrent SFC placement and migrations might be occurring at the time that an SFC is being migrated to a new location, which also introduces new challenges related to the availability of resources required for migration.

To sum up, our proposal tackles two major challenges: (i) to mitigate the impact of user mobility on the performance of SFCs in execution, triggering services migrations whenever it is necessary; and (ii) to find a fast and low-cost SFC migration plan with suitable recipient nodes and migration paths and execute the migration procedure to reduce the impact of user mobility on service performance. To address such challenges, we propose the following original contributions: (i) a novel proactive algorithm that triggers SFC migrations based on identified performance issues. In particular, we use multiple Markov chains of different orders to predict user location depending on the trajectory data available and estimate service delay at these future locations. If violations of the required maximum service delay are detected, the migration process is triggered to avoid/reduce such issues; (iii) also, as part of the algorithm, an iterative approach to find the migration plan that migrates the fewest number of VNFs of the SFC, thus minimizing the resource consumption, an important requirement in 5G-Edge networks.

The remainder of this paper is organized as follows. Section 2 discusses some related work. Section 3 describes the proposal, including the system model considered in the solution, the strategy for estimating user mobility, and the proposed SFC migration strategy. Section 4 presents the performed evaluation and Section 5 brings our final remarks.

2. Related work

Many proposals in the literature address the VNF/SFC Migration problem for different purposes. In general, they focus on migrating VNFs to improve service performance. To this end, solutions usually model their approaches as an Integer Linear Programming problem — which is costly due to the complexity of achieving optimal solutions. Therefore, solutions often include a heuristic approach to allow better scalability. Recent solutions also rely on machine learning approaches to estimate load at a node and migrate VNFs according to such a load, in a proactive strategy. Current service migration solutions have shortcomings that reduce their applicability in a dynamic mobility-prone 5G-Edge environment. These include the lack of awareness of user mobility's impact on service latency or assuming that a service is provided by a single VNF. The following proposals consider the presence of SFCs and the consequent challenges involving VNF migration in such cases, mainly when the same VNF is shared by multiple SFCs.

In [10] the authors propose a VNF migration approach aiming to keep the balance and performance of the network, taking into account the state of nodes and links. They consider the impact of migration in the case of shared VNFs and seek to optimize the performance of the network and of the provided service. The authors argue that previous works in the area do not consider fluctuations in traffic, while they do take such dynamism into account, thus being a more realistic approach. Unlike our work, the approach is reactive rather than proactive, which takes away the advantages of planning and executing a migration before service degradation [11]. Furthermore, despite considering the variation in traffic, user mobility is not addressed in the proposal.

Two user mobility-aware migration proposals are described in [12] and [7]. In them, the authors deal with the following decisions that need to be taken in the event of a handoff in 5G networks: (i) which VNFs of the SFC should be migrated, (ii) to which edge server(s), and (iii) how many resources of each server should be allocated so that the latency requirements of the service provided by the SFC are not violated. They formulate such decisions using integer programming and show that the problem is NP. Then, they propose an online algorithm to solve it. In [13], the authors solve the problem of optimal migration of SFCs when the volume of traffic demanded changes. The main difference between these works and ours, in terms of the conception of the problem and its solution, is their reactive nature. By proposing a proactive approach, which incorporates a prediction mechanism, we hope to anticipate possible QoS violations arising from user mobility instead of reacting to their occurrence. Therefore, our solution has the potential to decrease the number (or duration) of SLA violations, benefiting both the providers and the users.

In [14], instead of addressing a way to carry out a migration to reduce delay or service downtime, the authors focus on minimizing the impact of migration on services that traverse the migrated VNF. Their work considers the impact of VNF migration as the processing delay and link load differences that SFCs experience before and after a VNF migration, based on the previous and new locations of the VNFs. Different from our approach – which considers user mobility and service delay to trigger a migration – their work considers that a migration should occur when the average load of a node or a link is above a threshold, based on the maximum resource capacity. Therefore, their main goal is to find new hosting nodes for VNFs that are experiencing node or link overload while minimizing the migration impact on other SFCs. However, their work does not address several other aspects of the migration process, such as the migration resource cost and delay, which must be considered when finding a migration plan. If those aspects are neglected, the resultant destination nodes might end up being unfeasible. For example, a VNF migration might reduce the delay of one or more SFCs, but the time required to migrate the VNF to the selected node might be longer than the service duration due to resource scarcity at the path used for migration. Besides addressing the service impact, in terms of delay gains, our work also considers the migration cost when deciding where to migrate VNFs.

Liang et al. [8] focus on reducing service latency through SFC migrations. In their work, SFCs are modeled as a special class of queuing networks, where it is assumed that the network is at equilibrium (i.e., the service rate of a VNF is greater than the flow rate that arrives at the VNF). The authors propose two different heuristics for the migration decision. In the first heuristic, SFCs are considered for migration depending on a metric that takes into account the latency and the amount of resources that each VNF consumes in the service. SFCs with a metric value above average are considered for migration, focusing on increasing gains in terms of service delays. The second heuristic focuses on migrating VNFs from over-utilized servers to under-utilized ones. Similarly to our work, their first heuristic avoids migrating all VNFs of the SFC, selecting the ones with the most impact on service delay. However, their decision model does not consider the time required to execute a migration, which is addressed by our proposal. Their work also does not address user mobility.

In [15], the authors propose a Topology-aware Min-latency SFC Migration (TMSM) algorithm for dynamic edge computing networks. Their approach aims to minimize the propagation latency of SFCs while adhering to a long-term migration cost budget. Key advantages of their method include a well-balanced latency-cost tradeoff, online adaptability to topology changes, and provable performance guarantees. However, their algorithm relies on some parameters, such as the Lyapunov control parameter and the Markov approximation ratio, which need to be tuned carefully to achieve the desired performance. The optimal values of these parameters may depend on the network characteristics and the workload, and may not be easy to determine in practice. Moreover, their algorithm does not specify the implementation mechanism of the SFC migration strategy, such as how to transfer the VNF states, how to update the forwarding rules, and how to coordinate the migration process. These issues may affect the migration time and the service continuity and may require additional techniques or protocols to address.

In [16], the paper discusses the challenges of SFC optimization in SDN/VNF-enabled cloud environments, emphasizing the need for fault tolerance and dynamic adaptation to traffic fluctuations. Inspired by deep reinforcement learning (DRL), they propose a novel approach that models the SFC optimization problem as a Markov decision process (MDP). The reward function considers minimizing energy consumption, migration cost, maximizing revenue benefit, and load balancing. To address this complex problem, two DRL-based algorithms were introduced: single-agent double deep Q-network (SADDQN) and multi-agent DDQN (MADDQN). The paper presents a model that accounts for the buffer space requirements during migration to avoid overflow and calculates migration costs based on delay and additional network resource consumption. It also considers the selection of flows for migration and the overall design of the SFC optimization approach, aiming to minimize energy consumption and migration cost while maximizing revenue and load balancing. However, the paper assumes a linear model of cloud server power consumption, which may not reflect the realistic scenario of the cloud environment. Moreover, another disadvantage is as the network size grows, the communication and coordination overhead among multiple agents in MADDQN could become a bottleneck.

In [17], the paper introduces an optimization framework called Trade-LCM for dynamically mapping VNFs within SFCs to different virtual machines (VMs) in 5G networks. The framework aims to minimize user application service latency while reducing the cost of migrating VNFs. It formulates the problem as a multi-objective linear programming (MOLP) model, balancing the trade-off between latency and migration cost. Trade-LCM can handle multiple SFC requests and control domains, providing a rigorous approach for efficient VNF mapping. On the other hand, it lacks theoretical analysis regarding optimality, complexity, and convergence, limiting its scalability and applicability. In addition, the paper assumes complete and accurate network state information, which may not hold in dynamic 5G environments.

Emerging technologies such as 6G networks have already been taken into consideration in the literature for this type of problem. The authors [18] propose a Deep Learning-based Two-Stage Algorithm (DLTSA) to solve the VNF migration problem, considering both load balance and SFC delay in a 6G mobile edge computing network. The algorithm utilizes deep learning techniques to select the best migration location for concurrently migrating VNF instances based on network conditions. The goal of VNF migration is to minimize the end-to-end delay of all influenced SFCs while guaranteeing network load balance after migration. To achieve this, the authors utilize the Hybrid Genetic Evolution Algorithm to derive training data and then apply the running algorithm to derive VNF migration schemes. The simulation results demonstrate that the algorithms are scalable and efficient in terms of time. However, a potential drawback of the work is the need for sufficient computational resources to support the deep learning techniques utilized by the algorithm and potential challenges in training the algorithm with realistic VNF traffic patterns. In comparison, our work focuses on migration with reduced computational cost. Our migration procedure adopts an iterative approach to address performance issues by migrating the minimum number of VNFs within the SFC. Specifically, it prioritizes those migrations that yield the highest reductions in delay.

3. A novel mobility-aware strategy for SFC migration

An SFC is composed of a set of chained VNFs that together provide a full-fledged service to the user. The SFC migration problem consists of selecting the VNFs of the chain that must be moved to different and suitable compute nodes to improve the QoS of the service when performance issues are identified. In this work, we tackle the problem of migrating SFCs to cope with the impact introduced by user mobility, where a user might move to another location and use an access point further from the service in execution.

The SFC migration must be fast and effective, efficiently using the available resources. Meeting these requirements is not a trivial task for several reasons. First, there are many resource allocation constraints to the problem that must be satisfied when selecting suitable compute nodes to receive the migrating VNFs. These constraints include finding candidate nodes that have the necessary resources available and whose new service path – the path that will traverse these new nodes receiving the migrating VNFs – must satisfy the delay requirements of the service without violating the established SLA. Second, decisions related to where to move the VNFs should also consider the migration costs in terms of resource utilization and time required to migrate the necessary VNFs. These decisions include migrating only the VNFs that have the highest impact on the service performance and selecting the fastest path available to transfer the migration data from the previous node to the target node. Furthermore, these decisions must also be made in a dynamic environment where available resources are subject to change due to other migrations that might be occurring or SFC placement procedures, where resources will be used to deploy new services in the edge nodes. Additionally, when multiple VNFs of an SFC must be migrated to solve the performance issue, the migration decision of a VNF can also be impacted by the decision made to the previous migrating VNF. In this context, we propose a novel iterative approach to address the SFC migration problem. The proposal is detailed in the next subsections.

3.1. System model and notations

This subsection describes the system model under consideration and its related notations. The SFC migration problem consists of finding the location to migrate an SFC that is executing at the underlying infrastructure, considering the problem constraints. The model contains elements of the underlying infrastructure and the entities that are required to implement the services, as well as notations to describe processes and requirements related to the SFC migration solution. An overview of the notations and their descriptions can be found in Table 1.

The underlying infrastructure is described as an undirected graph $G = (H, A, L)$ whose vertices represent the edge nodes, and the edges represent the network links that interconnect these nodes. The set $H = \{h_1, h_2, \dots, h_n\}$ represents the computing nodes (or host nodes) where VNF Instances can be executed. Each computing node has a certain capacity in terms of CPU and memory resources. The available resources at a node vary according to the resources that are in use by hosted VNFs and are limited by the node's total CPU and memory resource capacity. The available CPU and memory resources – i.e., the amount of CPU and memory resources that are not in use at a node at a given time – are represented respectively as ca_h and ma_h. The set $A = \{a_1, a_2, \dots, a_n\}$ represents access nodes that connect users to the network and whose resources cannot be used to place VNFs. However, both access and computing nodes can be used to forward packets in the network. The set $L = \{l^1_{h_1,h_2}, \dots, l^k_{h_n,h_m}\}$ represents the physical links that connect access or computing nodes and that transmit service packets between nodes. A link $l^k_{h_n,h_m}$ is the kth link – since more than one link can connect two nodes – that connects the host nodes h_n and h_m. Links can also connect an access node a_n to a host node h_m (e.g., $l^k_{a_n,h_m}$). The available bandwidth of a link is represented as bw_l,

Table 1
System model notations.

Notation	Description
$G = (H, A, L)$	Undirected graph of the physical network.
$H = \{h_1, h_2, \dots, h_n\}$	Set of nodes where the VNFs can be executed
$L = \{l^1_{h_1,h_2}, \dots, l^k_{h_n,h_m}\}$	Set of links in the network
bw_l	Bandwidth available at link
$p(h_n, h_m)$	Represent a set of links that form a path between nodes h_n and h_m
$d^s_{n,m,k}$	Delay of link k between nodes h_n and h_m, considering the load and bandwidth demand of SFC s
$A = \{a_1, a_2, \dots, a_n\}$	Set of access nodes that connect users to the network
$U = \{u_1, u_2, \dots, u_n\}$	All the users that request SFCs
X_u	Set that contains the current and next predicted location of user u
ca_h	Available CPU resource at node h
ma_h	Available memory resource at node h
$IM_h = \{vi_1, vi_2, \dots, vi_n\}$	List of VNF images available at node h
v^i_s	VNF Instance of SFC Instance s
$D_c(v^{type}_{i,s})$	CPU demand for the type of VNF Instance v^i_s
$D_m(v^{type}_{i,s})$	Memory demand for the type of VNF Instance v^i_s
$size(v^{type}_{i,s})$	Total size for the type of VNF Instance v^i_s
SFC^{req}_i	i-th SFC Request
SFC^{inst}_y	y-th SFC Instance that represents an executing service
D^s_{bw}	Required bandwidth for SFC Instance s
$R_{Time}(SFC^{inst}_y)$	Remaining service time for a SFC Instance
$d^{v^i_s,h_n}_{mig}$	Migration delay of VNF Instance v^i_s from its host node to node h_n
$d^{a_n}_s$	Estimated service delay of SFC Instance s, considering access node a_n
d^{max}_s	Maximum delay agreed in the SLA of SFC s
mp_s	Migration plan for SFC Instance s
$g^s_{i,k}$	Estimated delay gain when migrating VNFs in the interval $[i, k+1)$ of SFC Instance s
$o^s_i \in H$	The ith node considering the ordered nodes that are part of an SFC Instance s (VNF host node or destination node)
M	Set of VNFs of an SFC being considered for migration
p_{short}	Shortest path in terms of delay for between two nodes
$p_{curr}(o^s_i, o^s_j)$	Current path being used to connect the nodes o^s_i and o^s_j
$limit_{bw}$	Percentage of the available bandwidth to be initially considered during migration decision
max^l_{bw}	Maximum percentage of available bandwidth of link l to be used by one VNF migration
$node(v^i_s)$	Denotes the computing node that hosts the ith VNF Instance of the SFC s
$\|s\|$	Size of the SFC s in terms of number of VNFs

with $l \in L$. A path composed of a set of links that connect nodes h_n and h_m is denoted by $path(h_n, h_m)$. We assume that the shortest path that connects two nodes — denoted by p_{short} — is always calculated considering the estimated delay of each link that will compose the path, provided by a monitoring component. Therefore, the shortest path will be the one that incurs the shortest delay.

In the system, users are represented by the set $U = \{u_1, u_2, \dots, u_n\}$. We consider that these users might move from one place to another at any time. As mentioned above, each user accesses the network through an access node. Therefore, the current user location is abstracted as the access node that a user is currently using to access the network. We consider that a mobility predictor can predict the future user location – i.e., the next access node that a user will rely on to access the network at his/her next movement. The notation X_u denotes the set containing the current and predicted locations of a user u at a given time – i.e., an element of the set can be the access node that the user is currently using to access the network or an access node that the user is likely to use in its next mobility event.

An SFC describes an ordered chain of VNFs that together provide a specific service. Each VNF is responsible for performing a specific function when a service packet arrives, and it has a given type (e.g., image processing, firewall). A VNF image of a given type must be present at the node to instantiate a VNF, and each VNF of a given type demands a certain amount of CPU and memory resources, denoted as $D_c(v_{i,s}^{type})$ and $D_m(v_{i,s}^{type})$, that must be available at the hosting node. The computing node that is hosting a VNF Instance v_s^i is denoted as $node(v_s^i)$. The VNF processing results in a new packet that will be forwarded to the next VNF in the chain. A user can request a service by making SFC Requests, denoted by SFC_i^{req}, where the index i represents the ith SFC Request that arrived at the system. An SFC Request describes the set of VNFs of a specific type that must be instantiated to provide the service and meet the service SLA. Upon the arrival of an SFC Request, an SFC Placement algorithm is executed to find appropriate nodes to host the required chain of VNFs, respecting resource and service constraints – such as the maximum service delay d_s^{max} described at the SLA). Each computing node has a set of VNF images $IM_h = \{vi_1, vi_2, \dots, vi_n\}$ available. This set represents which nodes already have a VNF image, and this information is also used during the estimation of the migration delay. Choosing nodes that already contain the required VNF image can reduce the migration delay since transferring the image would not be necessary during migration, as opposed to the ones that do not have the required VNF image.

The placement of an SFC generates an SFC Instance and a set of new VNF instances that are executed at the respective computing nodes. The ith VNF Instance of an SFC Instance SFC_k^{inst} is represented as v_s^i. Alternatively, a new SFC Instance can be mapped to VNF Instances that are already running, sharing them with other services that require the same VNF type. An SFC Instance keeps track of the user that requested the SFC, the set of VNF Instances that implement the service, the remaining service time – denoted as $R_{Time}(SFC_y^{inst})$ –, the user's current access node and a destination node, to where the packets must be delivered after being processed by the chain. The SFC Instance also keeps track of the service path currently in use. The service path – composed of physical links – connects the user's access point to the destination path, traversing all nodes that host the VNFs of the SFC, and is the path that will be used to transmit the packets of a service.

When an SFC is instantiated, the links that form the service path have a portion of their bandwidth reserved, based on the demand of the service, denoted as D_{bw}^s. The estimated delay to transmit the packet of an SFC Instance s through a link is denoted as $d_{n,m,k}^s$, and considers the reserved bandwidth for a service and the service average load, based on its average packet size. The estimated total delay for a service, which comprises the VNFs packet processing delays and the transmission delay from the access node a_n (which can be the user's current or the next predicted access node) until its destination, is represented as $d_s^{a_n}$. Fig. 1 shows an illustration of an SFC Instance executing in the underlying infrastructure. In the topology view (bottom of the figure), users access services running at edge servers (i.e., computing nodes) through 5G-enabled devices that are connected to the edge network through 5G base stations (i.e., access nodes). The topology can be abstracted to show only the access nodes, computing nodes, and the communication links. Access nodes are the ones that users rely on to access the edge network while computing nodes execute VNFs of SFCs. Communication links are used to establish a service path that connects the user's current access node and the destination node, traversing the VNF chain that composes the service.

During the lifespan of a service executing in the infrastructure, SFC migrations might be necessary to cope with performance issues caused by user mobility. A migration can be triggered at any time to solve these issues by finding a new location for the service. The notation o_i^s represents the set of VNFs and the destination node of an SFC Instance, where i represents the order of the node in the SFC – respecting the chain order and with the destination node as the last element. The migration algorithm estimates a delay gain $g_{i,k}^s$ that can be achieved by migrating a subset M of VNFs in the interval $[i, i+k)$. The result of the migration decision process for an SFC Instance s is a migration plan mp_s, which contains VNFs that will be migrated and a destination node, a migration path, and an updated partial service path – that connects the VNF at its new location to the following VNF – for each migrating VNF. The migration path is the set of links that will be used to transfer data of a certain migrating VNF from its original host node to its new host node. After the migration, the VNF will be located at a new host. Thus, the service path will be updated to reflect this change.

3.2. System architecture

Our proposal assumes that multiple components will be responsible for the placement and migration of VNFs. In this section, we describe the responsibility of each component in our proposed architecture based on the ETSI's architectural framework specification [19]. Fig. 2 shows an overview of the subsystems and components of the proposed solution along with the stakeholders (i.e., End User, Service Provider and Infrastructure Provider), related to each subsystem/component.

The *SFC Client Portal* provides an interface for end users to create and submit SFC requests. This component interacts with the SFC Manager to request the placement of requested services and with the Registry to provide information on cataloged services that can be requested by the user. In the *SFC Manager Portal*, service managers can view the status of running services, monitor their performance, and specify new types of SFCs and VNFs. This component operation depends on interactions with the SFC Manager and the Registry. The *Registry* component maintains a catalog of the different types of SFCs and VNFs that can be instantiated in the system. It has two sub-components, the *VNF Registry* — which maintains the definition of each VNF type, including their required CPU and memory, and associated VNF image — and the *SFC Registry* — which contains the definition of each SFC, including required VNFs.

The *SFC Manager* subsystem is responsible for the management tasks related to the placement and migration of SFCs and consists of four components. The *SFC Lifecycle Manager* is responsible for requesting the instantiation of SFCs at the underlying infrastructure and handling the termination of SFC instances by interacting with the NFV Orchestrator (NFVO) and the VNF Manager. The *SFC Monitor* monitors all SFC Instances in execution, periodically acquiring information related to the service delay, to check whether the SLA is being met. It interacts with the VNF Monitor to acquire processing latency at the VNFs and the Virtualization Middleware to get communication latency information. The *Mobility Management* component keeps track of the users' current locations and predicts future mobility to support the SFC migration process. The *SFC Migration* component is responsible for handling the migration process. It interacts with the Mobility Management and SFC Monitor components to estimate service latency and future user location and trigger the migration process. When a SFC Migration is necessary, the component interacts with the NFVO to orchestrate the migration of the required VNFs. The *VNF Manager* subsystem consists of two components that support the management of VNF instances. The *VNF Lifecycle Manager* supports the instantiation and termination steps of VNF Instances, while the *VNF Monitor* periodically monitors metrics related to the resource utilization and packet processing latency for each VNF executing at the system.

Our architecture relies on components — owned by the Infrastructure Provider — that manage the underlying infrastructure where services will be instantiated. Most of the components described below are part of the ETSI architecture [19], except the Virtualization Middleware. The *Virtualized Infrastructure Manager (VIM)* is responsible

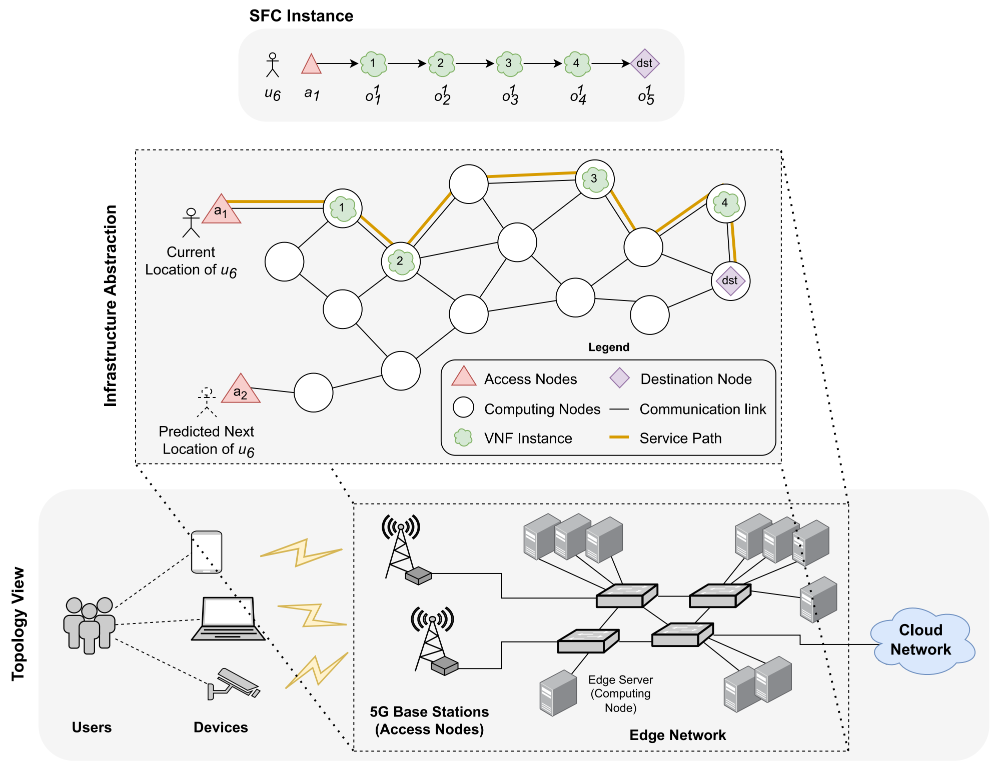


Fig. 1. Illustration of an SFC Instance at the underlying infrastructure.

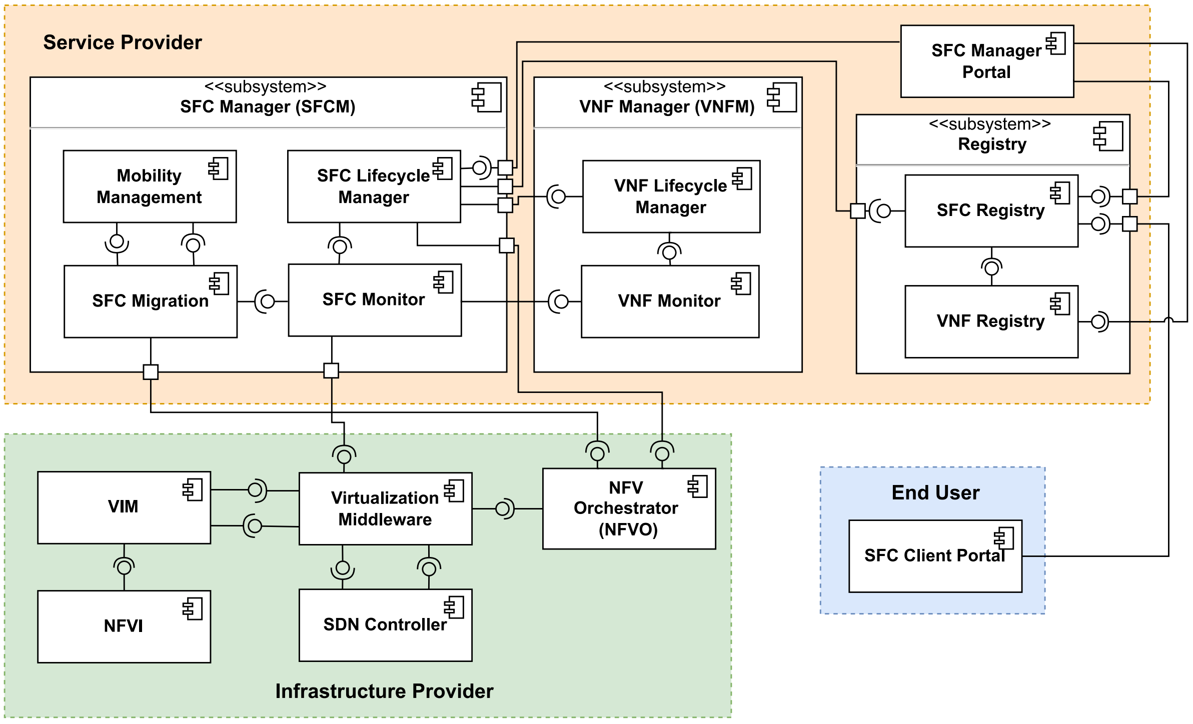


Fig. 2. Components of the proposed architecture.

for managing and monitoring NFVI computing, storage, and network resources. It exposes this functionality to the Virtualization Middleware so that it can translate the commands for VNF deployment and management. The *NFV Infrastructure (NFVI)* encompasses the set of hardware and software elements that form the environment where VNFs are deployed, executed, and managed. The *NFV Orchestrator (NFVO)* supports the execution of SFC placement and migration algorithms, interacting with the SFC Manager. The *SDN Controller* component provides interfaces to accommodate the demands for network resources, which is translated to the creation of virtual links and route installations. Finally, the *Virtualization Middleware* provides a common interface for the request and monitoring of the virtualized and physical resources in the infrastructure, translating abstracted commands to different resource managers and SDN controllers that might be available at the infrastructure.

3.3. Mobility prediction

According to Abu-Ghazaleh et al. [20], ensuring the availability of the network services anytime can only be feasible if it were feasible to predict, at any time, where a user will likely demand network usage. In 5G networks, a provider's service area is divided into small geographic areas called "cells". Each cell provides wireless coverage managed by a single access node or base station.

Mobile users may require handoff from one cell or base station to another while using a service. For the service provider to offer a successful handoff to its mobile users, it is necessary to ensure that the new resources demanded by the service are granted to the session in progress by the access point to which the user is being transferred. Otherwise, the session will be terminated or prematurely discarded due to insufficient resources in the new cell.

As presented by [20], one way of minimizing the dropping of handoff requests is to reserve enough resources at all neighboring cells for the handoff request. However, such an approach leads to a waste of resources and the possibility of significantly increasing the blocking rate for new connection requests. A more streamlined approach is to predict user mobility to identify the most likely user base stations to migrate and perform the necessary actions. In this context, in this work, we adopt a mobility prediction mechanism to identify the moments when an SFC migration might be necessary.

The migration strategy follows a proactive approach focused on anticipating performance issues that can affect executing service. Therefore, the migration relies on the mobility prediction mechanism to predict the next movement of the user and estimate if this movement will incur higher service delays, anticipating the impact caused by the mobility of a user.

In our approach, we use Markov chains [21] for modeling and predicting user mobility. A Markov chain is a probabilistic model that describes a sequence of events that might occur according to a given probability. State changes are called transitions, which have an associated probability. Markov chains in which the future state depends only on the current state are called first-order chains, whereas chains in which the future state depends on n past states are called chains of n-order. In our case, the Markov chain represents user mobility in the network. Since each cell is accessible through an access node, we abstract the fine-grained location of the user and represent it as the access node that is responsible for connecting the user to the network. Therefore, each state of the chain represents an access node that a user might use to access the services, whereas transitions represent the mobility event in which the user moves from one node to another.

Fig. 3 shows a representation of a second-order chain and a first-order chain. The main difference is that the main chain only considers the current state of the system (i.e., current access node), whereas the second-order chain considers both the current and previous states. The transition probabilities for the second and first-order chains are calculated according to Eqs. (1) and (2), respectively.

$$P_{f,c,p} = \frac{number\ of\ movements\ from\ node\ p\ to\ node\ c,\ then\ to\ node\ f}{number\ of\ movements\ from\ node\ p\ to\ c,\ then\ to\ any\ adjacent\ node} \tag{1}$$

$$P_{f,c} = \frac{number\ of\ movements\ from\ node\ c\ to\ node\ f}{number\ of\ movements\ from\ node\ c\ to\ any\ other\ adjacent\ node} \tag{2}$$

Instead of relying on only one chain for the prediction, we propose an approach where multiple chains are used. In this approach, three types of chains are built as described below. These chains are explored from the most specific (chain 1) to the most generic (chain 3), depending on the availability of the mobility data of the user whose next movement prediction is required. The number of access nodes returned by the mobility predictor (i.e., the N locations that the user is most likely to move) is a configurable parameter, referred to here as N.

1. Second Order Markov chain for a given user — Built using the historical mobility data of a specific user. Return access nodes that are most probable to be visited by the specific user in the next movement, considering the user's current and previous locations.
2. Generic Second Order Markov chain — Built using the historical mobility data of all users seen in the system. Return access nodes that are most probable to be visited by a user in the next movement, considering the user's current and previous locations.
3. Generic First Order Markov chain — Built using the historical mobility data of all users seen in the system. Return access nodes that are most probable to be visited by a user in the next movement, considering only the user's current location.

As displayed in Fig. 4, the prediction algorithm starts with the most specific Markov chain, the second order chain, in which probabilities are calculated based on the mobility of the specific user so that previous mobility data from other users do not impact the prediction of the current user. When a prediction request for the next location of a given user arrives, the algorithm checks if the chain contains any previous data regarding the mobility of this specific user. If data is available, the algorithm returns as the result of the prediction N locations that the user is most likely to move. If data of the current user is not available, then the algorithm uses the generic second-order chain, allowing historical data from other users to be used to calculate the transition probabilities. Considering data from all users, if the data is still unavailable, the algorithm falls back to the generic first-order chain that only considers the current state when calculating the probabilities of the next node. In the case that the data is still not available (e.g., the user is in a node that no other user has used), then the algorithm chooses N nodes that are adjacent to the current user node as the prediction result.

3.4. Proposed solution for SFC migration

This subsection describes the proposed solution to manage SFC migrations. In this paper, we do not detail the architectural aspects of the solution, but we assume the existence of a centralized entity with components responsible for both SFC migration and resource usage monitoring tasks. The strategies for the migration triggering, migration decision algorithm – where suitable nodes to receive the migrating VNFs and migration paths are chosen – and the migration execution are detailed.

3.4.1. Migration triggering

An important aspect regarding the migration of SFCs is deciding when the migration should occur. Migrations can be triggered using a reactive approach in which the SFC is migrated to solve performance issues that are currently affecting a service SLA or proactively migrating SFCs before the SLA is degraded. Also, different metrics can be considered to detect potential performance issues. A load-based approach can migrate overloaded VNFs whenever the usage of the hosting node resources gets close to its maximum capacity. In contrast, a delay-based migration approach requires the monitoring of packet processing and forwarding times delay, verifying whether the packets are taking too much time to reach the service destination.

Our migration triggering mechanism follows a proactive approach based on the mobility predictor mentioned in Section 3.3. Since our primary goal is to reduce performance issues caused by user mobility,

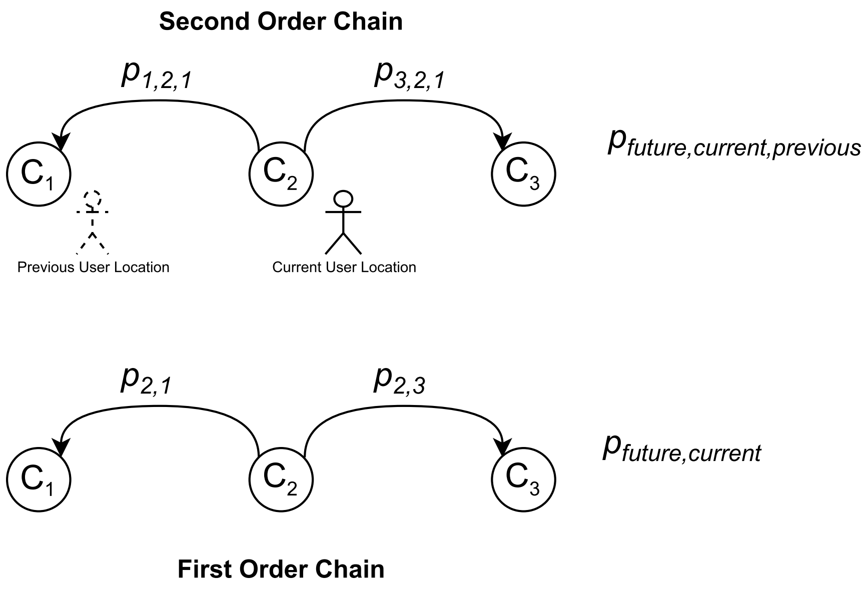


Fig. 3. User mobility represented in Markov chains.

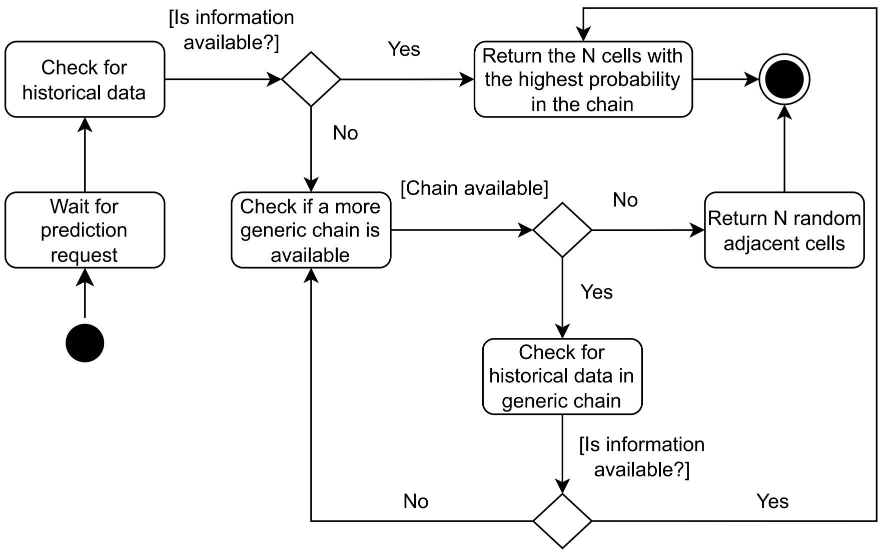

Check for historical data
[Is information available?]
Yes
Return the N cells with the highest probability in the chain
No
Wait for prediction request
Check if a more generic chain is available
[Chain available]
No
Return N random adjacent cells
Yes
Check for historical data in generic chain
[Is information available?]
No
Yes

Fig. 4. Mobility prediction procedure.

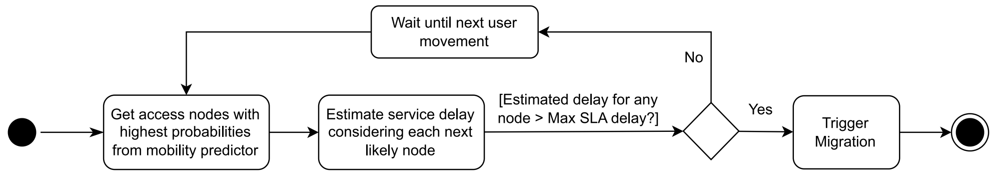
a load-based approach would not be effective when the distance of the user from the SFC is the root cause of the performance issues. Therefore, we adopt a delay-based approach, focusing on keeping the service delay within the maximum delay specified in its SLA.

Due to the proactive approach of the triggering mechanism, it is necessary to know beforehand if a migration will result in better performance. Therefore, we consider that a monitoring component will keep track of the service, nodes, and link characteristics (such as packet processing time in each VNF, link transmission delay, and average packet generation rate of the service) to allow making estimates of a future service delay, considering the next movement of the user. Fig. 5 depicts the activities involved in the SFC migration triggering procedure, which integrates the SFC migration process. Initially, the N access nodes that the user is most likely to use as an access point in its next movement are retrieved from the mobility predictor. Then, an estimate of the service delay is calculated considering each node returned by the predictor. If the estimated service delay for any of these nodes is above the maximum allowed delay defined in the SLA, the migration is triggered. If the delay is below the maximum for each node, the mechanism waits for the next user movement.

3.4.2. Iterative migration decision

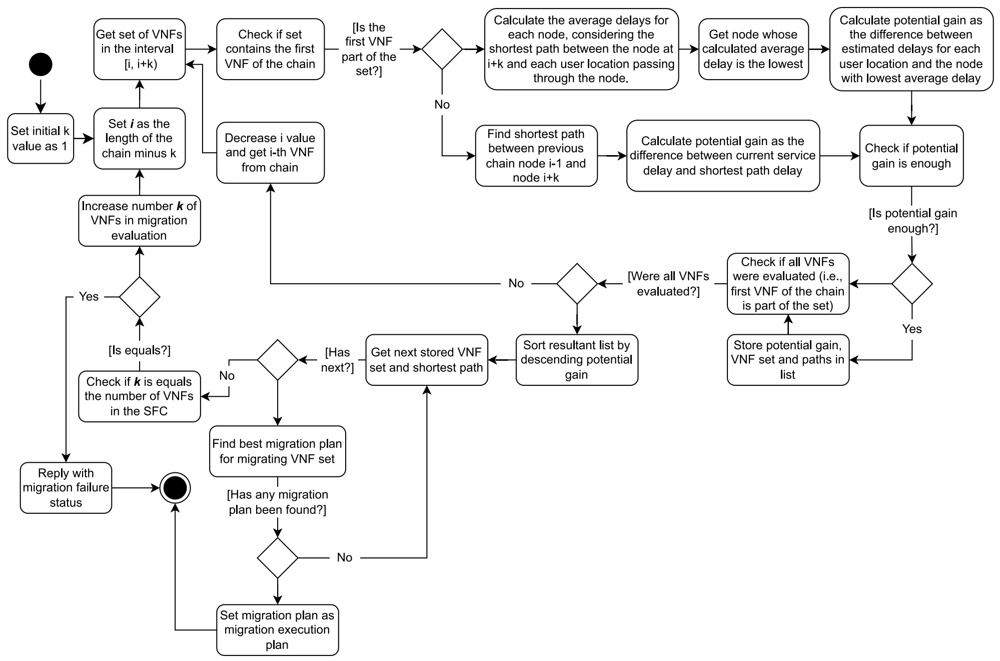
After the migration algorithm detects that an SFC must be migrated, the following step is to decide which VNFs that compose the SFC should be migrated to solve the identified performance issue. To reduce the amount of utilized resources and the time required by the SFC migration procedure, our solution follows an iterative approach that identifies the smallest number of VNFs of the chain that must be migrated to improve service performance and avoid SLA violations.

In each iteration, our solution estimates the performance gains of migrating k VNFs of the SFC. The gain metric considered is the reduction of the service delay. As shown in Fig. 6, the algorithm starts with a k value of one, which means that at the initial iteration, only one VNF of the chain is considered to be migrated. Then, it is necessary to identify which VNF of the chain can be migrated so that the performance issue can be solved with a single VNF migration.



Wait until next user movement
Get access nodes with highest probabilities from mobility predictor
Estimate service delay considering each next likely node
[Estimated delay for any node > Max SLA delay?]
No
Yes
Trigger Migration

Fig. 5. Migration triggering mechanism.



Set initial k value as 1
Set *i* as the length of the chain minus k
Get set of VNFs in the interval [i, i+k)
Check if set contains the first VNF of the chain
[Is the first VNF part of the set?]
Calculate the average delays for each node, considering the shortest path between the node at i+k and each user location passing through the node.
Get node whose calculated average delay is the lowest
Calculate potential gain as the difference between estimated delays for each user location and the node with lowest average delay
No
Find shortest path between previous chain node i-1 and node i+k
Calculate potential gain as the difference between current service delay and shortest path delay
Check if potential gain is enough
[Is potential gain enough?]
Yes
Store potential gain, VNF set and paths in list
Check if all VNFs were evaluated (i.e., first VNF of the chain is part of the set)
[Were all VNFs evaluated?]
No
Decrease i value and get i-th VNF from chain
Sort resultant list by descending potential gain
Get next stored VNF set and shortest path
[Has next?]
No
Check if *k* is equals the number of VNFs in the SFC
[Is equals?]
Yes
Increase number *k* of VNFs in migration evaluation
Reply with migration failure status
Find best migration plan for migrating VNF set
[Has any migration plan been found?]
No
Set migration plan as migration execution plan

Fig. 6. Procedure to select the VNFs to be migrated.

Therefore, the algorithm iterates over the chain of VNFs, estimating the gains that can be achieved if the ith VNF of the chain is migrated. To calculate the gain of migrating the ith VNF, the algorithm finds the shortest available path – in terms of delay – between the ordered node o of index $i-1$ and the ordered node $i+k$ of the chain. If a path shorter than the current path between these nodes exists, then the gain is calculated as the difference between the service delay of the current path and the estimated delay of the shortest path. Eq. (3) shows the gain calculated for an SFC Instance s, migrating the VNFs between $i-1$ and $i+k$. Note that a different calculation is made if the first VNF of the chain is being considered for migration. In this case, the algorithm analyzes the delay gain considering the set X_u, which contains the current user location and the predicted next locations. The number of predicted nodes/locations that the user is most likely to move in his/her next mobility event depends on the N parameter used in the mobility predictor. The current delay $\alpha^{a_z}_{i,k,u}$ and the shortest delay $\beta^{a_z}_{c,i,k,u}$ passing through an intermediate node c and considering a location a_z are respectively calculated as described in Eqs. (4) and (5). An intermediate node c is considered as an intersection node between each location $a_z \in X_u$ and node o with index $i+k$. The pseudocode related to this step can be seen in Algorithm 1.

$$g^s_{i,k} = \begin{cases} \sum_{l^k_{n,m} \in p_{curr}(o^s_{i-1}, o^s_{i+k})} d^s_{n,m,k} - \sum_{l^k_{n,m} \in p_{short}(o^s_{i-1}, o^s_{i+k})} d^s_{n,m,k}, i > 1 \\ \max_{c \in H}\left(\frac{\sum_{a_z \in X_u} (\alpha^{a_z}_{i,k,u})}{|X_u|} - \frac{\sum_{a_z \in X_u} (\beta^{a_z}_{c,i,k,u})}{|X_u|} \right), i = 1 \end{cases} \tag{3}$$

$$\alpha^{a_z}_{i,k,u} = \left(\sum_{l^k_{n,m} \in p_{short}(a_z, o^s_i)} d^s_{n,m,k} + \sum_{l^k_{n,m} \in p_{curr}(o^s_i, o^s_{i+k})} d^s_{n,m,k} \right) \tag{4}$$

$$\beta^{a_z}_{c,i,k,u} = \left(\sum_{l^k_{n,m} \in p_{short}(a_z, c)} d^s_{n,m,k} + \sum_{l^k_{n,m} \in p_{short}(c, o^s_{i+k})} d^s_{n,m,k} \right) \tag{5}$$

$$d^{a_n}_s - g^s_{i,k} < d^{max}_s, \forall a_z \in X_u, i > 1 \tag{6}$$

$$d^{a_z}_s - (\alpha^{a_z}_{i,k,u} - \beta^{a_z}_{c,i,k,u}) < d^{max}_s, \forall a_z \in X_u, i = 1 \tag{7}$$

If the calculated gain is enough to solve the identified performance issue (i.e., the current service delay minus the calculated gain is below the maximum delay in the service SLA for each user location), as shown in Eqs. (6) and (8), the gain value, shortest path, and index i are stored for further evaluation. After that, the algorithm checks whether all VNFs were evaluated (*i.e.*, if a VNF was considered for migration at some point). If not, the algorithm decreases index i and continues the analysis. An example is shown in the left part of Fig. 7. If all VNFs were evaluated, then the algorithm sorts the stored sets of migrating VNFs by a decreasing gain value and tries to find the best migration plan, choosing appropriate destinations for the set of migrating VNFs in the nodes that are part of the shortest path. This procedure is explained in Section 3.6. The best migration plan will be used to perform the migration. If a migration plan cannot be found, then the algorithm continues the analysis with an increased k value – *i.e.*, the analysis continues considering the migration of a higher number of VNFs. Fig. 7 shows an example of each VNF being evaluated with different k values.

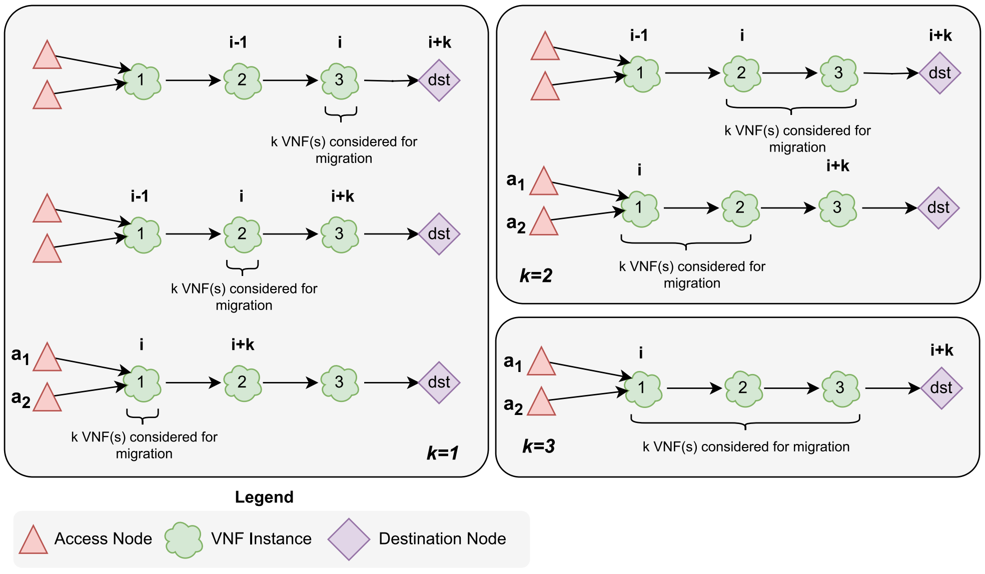


Fig. 7. Example of the iterative migration evaluation with an SFC composed of 3 VNFs.

Algorithm 1 Iterative procedure to identify potential gains when migrating a subset of VNFs.

```
1: for $k$ from 1 to $|s|$ do ▷ k is the number of VNFs being considered
2:     for $i$ from $|s| - k$ to 1 do
3:         if i=1 then ▷ If the first VNF is being considered...
4:             for $c \in H$ do ▷ For each intermediate node
5:                 for $a_{z_s} \in X_u$ do ▷ For each user location
6:                     $d_s^{a_z} \leftarrow$ ServicePathDelay($a_z$,$o^s_{|s|+1}$)
7:                     $\alpha^{a_z}_{i,k,u} \leftarrow$ ShortestPathDelay($a_z$,$o^s_i$) + ServicePathDelay($o^s_i$,$o^s_{i+k}$)
8:                     $\beta^{a_z}_{c,i,k,u} \leftarrow$ ShortestPathDelay($a_z$,$c$) + ShortestPathDelay($c$,$o^s_{i+k}$)
9:                     if $d_s^{a_z}$ - ($\alpha^{a_z}_{i,k,u}$ - $\beta^{a_z}_{c,i,k,u}$) $\geq d_s^{max}$ then:
10:                        continue ▷ Skip this, since it will violate the SLA
11:                    $potential\_avg\_gain\_c \leftarrow \frac{\sum_{a_z \in X_u}(\alpha^{a_z}_{i,k,u})}{|X_u|} - \frac{\sum_{a_z \in X_u}(\beta^{a_z}_{c,i,k,u})}{|X_u|}$
12:                    if $potential\_avg\_gain\_c > g^s_{i,k}$ then
13:                        $g^s_{i,k} \leftarrow potential\_avg\_gain\_c$
14:                        if $g^s_{i,k} > 0$ then
15:                            $potential\_vnfs\_to\_mig.add$(GetVNFsInInterval($i$,$i+k$))
16:        else
17:            $g^s_{i,k} \leftarrow$ ServicePathDelay($o^s_{i-1}$,$o^s_{i+k}$) − ShortestPathDelay($o^s_{i-1}$, $o^s_{i+k}$)
18:            for $a_{z_s} \in X_u$ do
19:                $d_s^{a_z} \leftarrow$ GetServicePathDelay($a_{z_s}$, $o^s_{|s|}$)
20:                if $g^s_{i,k} > 0$ and $d_s^{a_z} - g^s_{i,k} < d_s^{max}$ then
21:                    $potential\_vnfs\_to\_mig.add$(GetVNFsInInterval($i$,$i+k$))
```

3.4.3. Migration paths definition

After finding which VNFs should be migrated using a shorter service chain path that can lead to better latency performance, the next step of the migration process is to find nodes that are suitable for receiving the VNFs with a reasonable migration cost. However, considering all available nodes as candidates to receive the migrating VNFs would significantly increase the number of possibilities for the allocations of the migrating VNFs, resulting in poor scalability. To reduce the complexity, our heuristic considers as candidates only the nodes that appear in the shortest path that incurs the highest gain. Also, migrating VNFs should be mapped to these nodes in the order defined by the SFC — *i.e.*, if a VNF is mapped to a node that appears first in the path, another VNF with a higher index cannot be mapped to a node that precedes the other node. In this process, different allocation possibilities are generated, mapping the set of migrating VNFs to the nodes available in the shortest path. These possibilities may also include allocating multiple VNFs to the same node if resources are available. In this case, we consider that the delay between two VNFs allocated to the same node is negligible and, thus, zero. These possibilities must also comply with the resource and service delay constraints discussed in Section 3.5.

As shown in Fig. 8, this part of the algorithm begins setting as the subpath – whose nodes can be used for allocation – the shortest path stored at the previous step. Then, the algorithm gets the next VNF of the set of candidate VNFs for migration – sorted according to their order in the SFC – and tries to allocate this VNF in the first node of the subpath whose resources constraints are satisfied – i.e., the node that will receive the VNF must have enough available resources to satisfy the VNF demands. If the node has the necessary resources, the algorithm considers these resources as reserved and finds a migration path between the current VNF location and the node being evaluated. At this phase, the algorithm considers that this migration path is the shortest path in terms of the delay to migrate that VNF, considering the consumption of $limit_{bw}$ — as defined in Eq. (8) — of the available bandwidth of the links that compose the migration path.

$$limit_{bw} = \frac{1}{remainingVNFs} \tag{8}$$

The $limit_{bw}$ threshold is necessary to avoid making the solution unfeasible by taking all the bandwidth of a link that would be necessary for the migration of other VNFs of the set whose node and path are still unknown. For instance, if the algorithm considers the consumption of all available bandwidth of links that compose the chosen migration path of a VNF when deciding where to migrate the next VNF of the SFC, the only feasible migration path might be the one that shares a specific link with the previous migration path, whose resources have been already taken for the migration of the previous VNF, rendering the migration plan unfeasible. Therefore, the algorithm avoids this situation by considering the consumption of only $limit_{bw}$ of the available bandwidth so that if the next VNFs migration paths require the same link, enough link resources will be available to migrate each one of them.

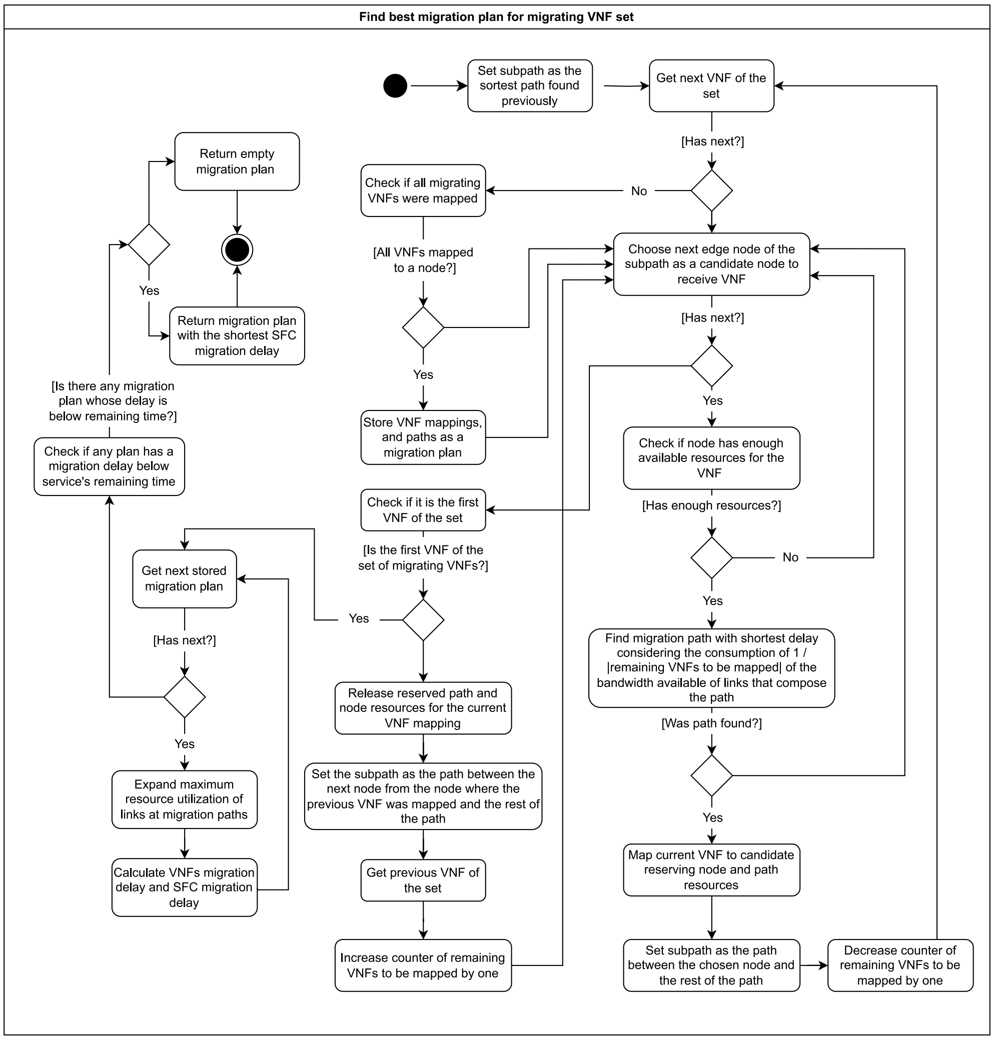

Find best migration plan for migrating VNF set
Set subpath as the sortest path found previously
Get next VNF of the set
[Has next?]
No
Check if all migrating VNFs were mapped
[All VNFs mapped to a node?]
Yes
Store VNF mappings, and paths as a migration plan
Choose next edge node of the subpath as a candidate node to receive VNF
[Has next?]
Yes
Check if node has enough available resources for the VNF
[Has enough resources?]
No
Yes
Find migration path with shortest delay considering the consumption of 1 / |remaining VNFs to be mapped| of the bandwidth available of links that compose the path
[Was path found?]
Yes
Map current VNF to candidate reserving node and path resources
Set subpath as the path between the chosen node and the rest of the path
Decrease counter of remaining VNFs to be mapped by one
Check if it is the first VNF of the set
[Is the first VNF of the set of migrating VNFs?]
Yes
Release reserved path and node resources for the current VNF mapping
Set the subpath as the path between the next node from the node where the previous VNF was mapped and the rest of the path
Get previous VNF of the set
Increase counter of remaining VNFs to be mapped by one
Return empty migration plan
Return migration plan with the shortest SFC migration delay
Yes
[Is there any migration plan whose delay is below remaining time?]
Check if any plan has a migration delay below service's remaining time
Get next stored migration plan
[Has next?]
Yes
Expand maximum resource utilization of links at migration paths
Calculate VNFs migration delay and SFC migration delay

Fig. 8. Target nodes and migration paths definition.

Then, the destination node and the migration path for the current VNF are stored as a possible migration plan. After that, the subpath is updated to contain only the links and nodes starting from the last mapped destination node forward. Because VNFs must follow the order established in the SFC, allowing the next VNF to be allocated in a node farther in the chain than the node in which the previous VNF was mapped would incur additional delay overhead.

Next, the algorithm tries to find the recipient node (in the updated subpath) and the migration path for the next VNF candidate for migration of the set, repeating the same aforementioned process. When the last node of the subpath is evaluated as a possible candidate to receive a VNF, the algorithm continues to explore the possibilities of allocation of the previous VNF. The main idea of this part of the algorithm is to find the different migration possibilities for the VNFs in the set, storing the destination nodes and migration path for each VNF as a possible migration plan for each possibility that satisfies the constraints.

As previously mentioned, the $limit_{bw}$ is applied for all links that compose a migration path. However, some links might be shared between multiple migration paths, while others might be used to migrate only one VNF – i.e., used by only one migration path. Thus, after the phase of selecting possible combinations of recipient nodes and migration paths for migrating VNFs, the algorithm optimizes the amount of resources of a link to be used to migrate a VNF. For each stored migration plan, the amount of bandwidth resources of the links in the migration paths that the migration is allowed to use is expanded to the link's maximum bandwidth, so that the migration can occur as fast as possible. If a link is used for the migration of more than one VNF of the SFC, then each VNF migration will be allowed to use the link's available bandwidth equally, as defined by Eq. (9).

$$max^{l}_{bw} = \frac{1}{\#\ of\ migration\ paths\ that\ use\ link\ l} \tag{9}$$

After that, the best plan will be selected according to a migration delay criterion. The best migration plan will be the one with the shortest migration delay. The SFC migration delay is considered as the greater migration time of all migrating VNFs of the SFC, as shown in Eq. (10). The migration possibility with the fastest SFC migration delay will be used as the migration plan during the execution.

$$SFC_{mig_{delay}} = \max_{v^i_k \in M} (d^{v^i_s, h_n}_{mig}) \tag{10}$$

3.5. Candidate node and migration cost constraints

When finding a destination node for a migrating VNF Instance $v_{i,j,k}$, a suitable candidate must have sufficient available resources to allocate the demands of the migrating VNF. Eq. (11) represents that h_n must have enough CPU resources available considering the demand of the VNF. Likewise, Eq. (12) shows that available memory resources at the node must be greater than the VNF demand.

$$ca_{h_n} \geq D_c(v^{type}_{i,s}) \tag{11}$$

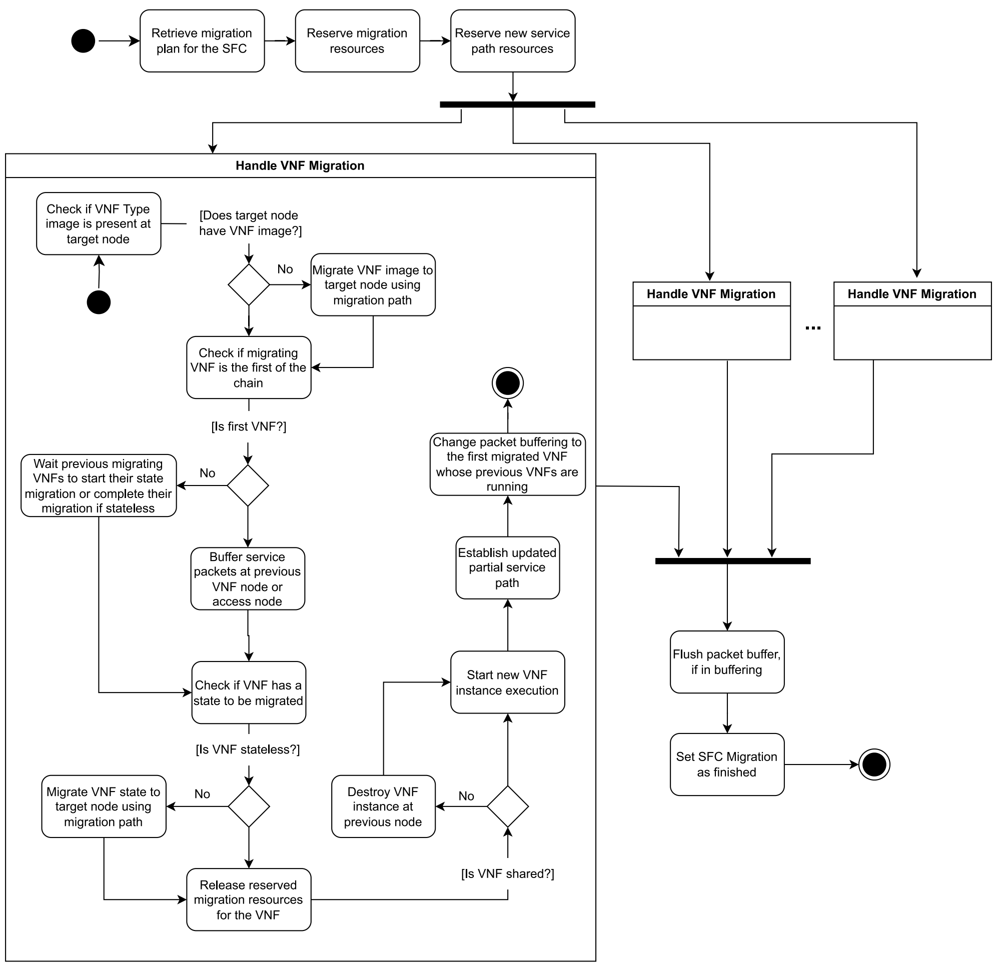

Retrieve migration plan for the SFC
Reserve migration resources
Reserve new service path resources
Handle VNF Migration
Check if VNF Type image is present at target node
[Does target node have VNF image?]
No
Migrate VNF image to target node using migration path
Check if migrating VNF is the first of the chain
[Is first VNF?]
No
Wait previous migrating VNFs to start their state migration or complete their migration if stateless
Buffer service packets at previous VNF node or access node
Check if VNF has a state to be migrated
[Is VNF stateless?]
No
Migrate VNF state to target node using migration path
Release reserved migration resources for the VNF
[Is VNF shared?]
No
Destroy VNF instance at previous node
Start new VNF instance execution
Establish updated partial service path
Change packet buffering to the first migrated VNF whose previous VNFs are running
Handle VNF Migration
...
Handle VNF Migration
Flush packet buffer, if in buffering
Set SFC Migration as finished

Fig. 9. VNF migration execution.

$$ma_{h_n} \geq D_m(v_{i,s}^{type}) \tag{12}$$

Another constraint is related to the time required to migrate a VNF. The migration delay is calculated according to Eq. (14). As shown in Eq. (13), a migration must have a delay below the remaining service time – i.e., migrating a VNF cannot take more time than the remaining duration of the service. Otherwise, the service would not be necessary anymore by the end of the migration. The total size of the VNF is composed of its image and its state, as described in Eq. (15). If the VNF image is already available at the destination node, it is not necessary to transfer its image. Thus, its size is not considered in the delay calculation. Also, if a migrating VNF is stateless, it is not necessary to make a state transfer. Therefore, the state size is zero during the VNF migration delay calculation.

$$d_{mig}^{v_s^i,h_n} < R_{Time}(SFC_s^{inst}) \tag{13}$$

$$d_{mig}^{v_s^i,h_n} = \sum_{l \in path(node(v_k^i),h_n)} \frac{size(v_{i,s}^{type})}{bw_l * max_{bw}^l} \tag{14}$$

$$size(v_{i,s}^{type}) = size(v_{i,j,s}^{state}) + size(v_{i,j,s}^{img}) \tag{15}$$

3.6. Migration execution

After finding the best migration plan for the SFC, containing the VNFs that will be migrated, the destination nodes, the migration paths, and the new service path – considering the VNFs at their new location – the next step is to execute the migration procedure.

As shown in the diagram of Fig. 9, after retrieving the migration plan, the next step is to reserve the necessary resources for each VNF migration at the underlying infrastructure. This pre-reservation is necessary to avoid the required node and link resources for migration becoming unavailable due to the placement of other SFCs or other concurrent SFC migrations that might occur during the migration. The reserved resources include (i) the bandwidth of links that compose the path that will be used for the VNFs migrations, and (ii) the CPU and memory resources at the nodes that will receive the migrating VNFs. Then, the bandwidth of links that will form the new service path must also be reserved. However, since a link might be used for the migration and to establish the new service path, the step of reserving the service path resources will only reserve resources of links that are not used by any migration path or whose reserved resources for migration are below the required for the service – i.e., if the reserved amount of a link for the migration is sufficient for the service demands, only the reservation for the migration will occur. In this way, resource utilization is optimized, avoiding competition for the link resources by the migration and service paths and reducing the migration duration.

Once the required resources are reserved, a parallel procedure begins for each migrating VNF. In this procedure, it is checked whether it is necessary to migrate the VNF image. The VNF image transfer occurs only when the destination node does not have the required VNF image. After the image transfer, the next step is to migrate the VNF state, in the cases of stateful VNFs. Some constraints must be satisfied to begin the state migration phase. If the VNF is the first of the set of migrating VNFs – considering the VNF order in the SFC – then the packets of the SFC are buffered at the host of the previous non-migrating VNF or access

Algorithm 2 Procedure to execute the Iterative SFC Migration.

```
1: function handleIterativeSFCMigration($s$)
2:     $mp_s$ ← GetSFCMigrationPlan($s$)
3:     for vnf, origin, target, mig_path, service_path in $mp_s$ do
4:         r ← reserveMigAndServiceRes(vnf, target, mig_path, service_path)
5:         callThread(migrateVNF($s$, vnf, origin, target, mig_path, service_path, r))
6:     waitForMigrationsToComplete($s$)
7:     stopPacketBuffering($s$)
8:     return success
9:
10: function migrateVNF($s$, vnf, origin, target, mig_path, service_path, r)
11:     if not target.isVNFImageAvailable(vnf.type) then
12:         executeImageMigration(vnf, target, mig_path, r)
13:     if not isFirstVNFToBeMigrated($s$,vnf) then
14:         waitForPreviousVNFsMigrations()
15:     else
16:         if vnf.index() = 1 then
17:             bufferPackets(GetAccessNode($s$, vnf))
18:         else
19:             bufferPackets(GetPrevVNFNode($s$, vnf))
20:     if vnf.isStateful() then
21:         executeStateMigration(vnf, mig_path, r)
22:     releaseMigrationResources(vnf, r)
23:     if not vnf.isShared() then
24:         destroyVNFInstance(origin, vnf)
25:     startVNFInstance(target, vnf)
26:     updateServicePath(vnf, service_path)
27:     if arePreviousVNFsExecuting() then
28:         updatePacketBuffering(target, vnf)
```

node. This is done to avoid changing the state of the VNF instance at its original location, which would require several state transfers to update the state at the destination node. Before starting the state migration, the VNF Instance cannot be processing any packets, and the virtual link — that represents the reserved resources at physical links that connect two VNFs — that connects previous VNFs (or access node) to the migrating VNF Instance must be idle, to avoid the arrival of packets – which would change the VNF state at its original location – while the state migration is in place. An additional constraint is that all previous migrating VNFs need to have started the VNF state migration procedure or finished their migration. Note that this constraint is not valid for the first migrating VNF of the set since there is no other migrating VNF before it. With these requirements satisfied, the VNF state – if stateful – is transferred to the destination node.

After the steps described previously, the migration path resources used by the VNF migration are released, except the ones that will be used to establish the service path. If the VNF Instance is not shared with multiple SFCs, the VNF Instance at its original location is destroyed, and the VNF at the destination node is instantiated. Then, the partial service path – which connects the migrated VNF to the previous and next VNFs of the SFC – is established. If the required resources for the service path establishment are being used for the migration of another VNF of the SFC, then the process waits for the migration of the other VNF – and the following release of the migration resources – to establish the path. Next, the node in which the packets were being buffered is changed to the node with the first migrating VNF that is already running at its new location, allowing packets to be processed by the running VNFs until reaching the buffering node and avoiding making the packets wait for the migration of all migrating VNFs. Finally, when all migrating VNFs have been migrated to their destinations, the packet buffering is disabled, and the SFC is set as migrated. The pseudocode for the iterative migration execution can be seen in Algorithm 2.

Table 2
Simulation Parameters. CIs calculated considering a confidence of 95%.

Parameter	Values
Users	Variable, based on dataset sample (Avg.: 24.5 ± 0.43)
Access nodes	Variable, based on dataset sample (Avg: 32.3 ± 2.43)
Computing nodes	Variable, based on dataset sample (Avg: 148.8 ± 0.45)
Number of VNF types	10
Number of SFC types	10
Interval of placement executions	100 ms
Node CPU resource	50 MIPS
Node RAM resource	512 MB
Link bandwidth	2 Gbps
VNF images in a node	1, 2, 3, 4 or 5
VNF CPU demand	5 MIPS
VNF RAM demand	32, 64 or 128 MB
VNF image size	256, 512 or 1024 MB
SFC maximum delay	20 ms
SFC required bandwidth	30 Mbps
VNFs per SFC	4
Packet size	3750 bytes

4. Performance evaluation

In this section, we detail the experiments held to evaluate our proposed solution. We developed a proof-of-concept implementation using the SimPy [22] environment to simulate the system under consideration and evaluate the impact of the Iterative Migration proposal. We also compared our proposed method with an approach that considers the migration of all VNFs of an SFC, which is referred to as Whole SFC Migration.

To obtain the data related to user mobility, we relied on a mobility dataset [23] that contains the association events of users and the location of each access point in a wireless network. We considered each access point to be an access node in the edge environment and a mobility event was generated each time that a user changes its association between access points. After a filtering process, in which reassociation events were removed, the final dataset contains information regarding the mobility of 13888 users moving between 602 access nodes. Computing nodes were generated in a grid-like topology and can be connected given a predetermined probability that decreases with the number of links of each node. The number of computing nodes depends on the calculated area, considering the positions of access nodes in the dataset sample.

We also considered that each user will request an SFC the first time they appear in the system. Arriving SFC requests will be considered for placement at the next execution of the placement procedure based on the specified time interval of 100 ms. We considered a placement algorithm that allocates VNFs following a greedy shortest-delay approach. Regarding the migration decision, we consider that a migration of a service should only occur if its estimated migration time is less than half of the remainder service time. Each placed request will generate packets to be processed by the SFC. The computing nodes and link capacities, VNF requirements, and other simulation parameters are described in Table 2. Some parameters' values – such as the VNF RAM and image size – were randomly selected among the possible values displayed in Table 2. The experiment was executed 1000 times. In each execution, a different topology is generated based on the sampled data from the dataset, and the user quantity and mobility are based on the number of users that were present in the sample.

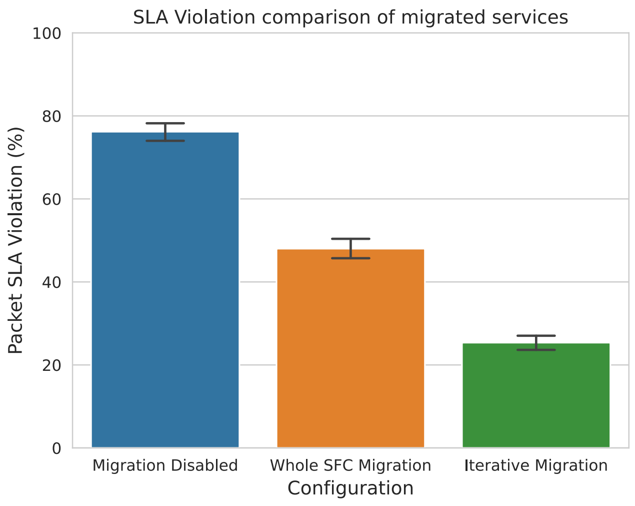


Fig. 10. Comparison of the average packet SLA violation of services that undergo a migration when enabled.

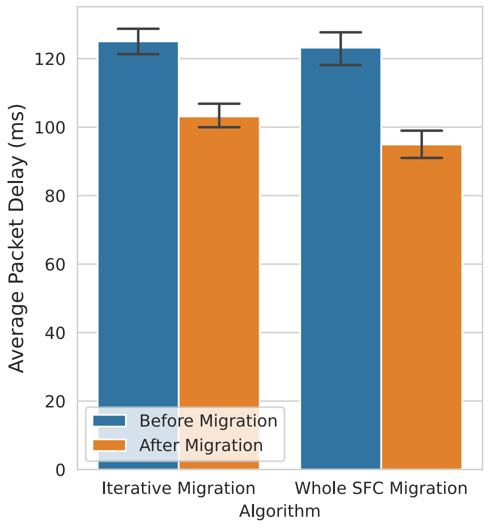


Fig. 12. Comparison of the average packet delay.

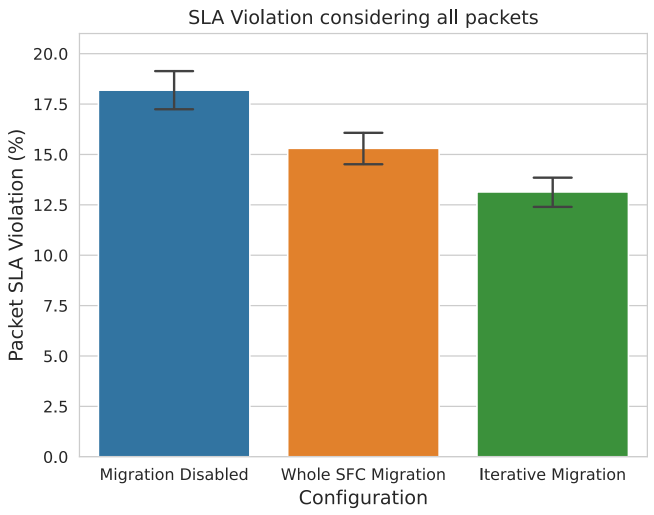


Fig. 11. Comparison of the average packet SLA violation of all services.

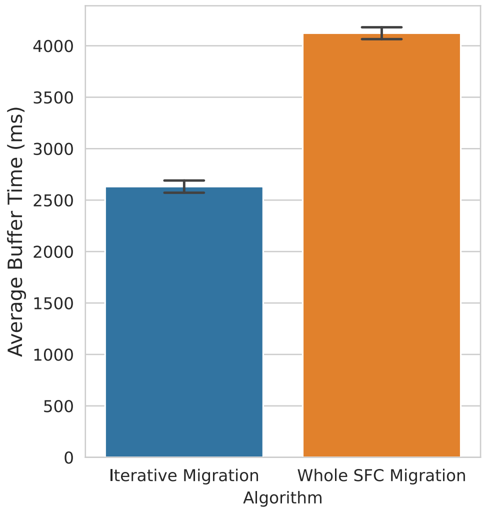


Fig. 13. Comparison of the average packet buffer time.

The first set of results assessed the impact of our migration proposal on the performance of ongoing services, measured in terms of the average percentage of packet SLA violation and packet delay. We assume that the packet delay is the interval between the time a packet takes to arrive at the system and the time to reach the destination node of the service, including the processing delays at each VNF and the transmission delays of the SFC path. A packet violates the service SLA if its delay is above the maximum tolerated delay (a QoS parameter) described in the SFC request.

For this analysis, we executed experiments considering three different scenarios, namely (i) Iterative migration; (ii) Whole SFC migration; and (iii) Migration Disabled. In the Iterative Migration scenario, our proposal is used to migrate a subset of VNFs of an SFC. In the Whole SFC Migration scenario the algorithm tries to find a migration plan for all VNFs of the SFC. Both are also compared with a scenario where no migration mechanism is used, referred to as Migration Disabled. Fig. 10 shows the average percentage of packets that violate the service SLA, considering only the packets of services that were migrated in any scenario. In this case, the proposed migration approach significantly decreased the average percentage of packet violation from 76.2% to 25.4%, which represents a reduction of approximately 66.7%, while the whole SFC migration approach reduced packet violation to 48%, a reduction of 37%. Fig. 11 shows the impact of the migration for all services, migrated or not. In this case, the proposed migration procedure reduced the average percentage of packet violation by around 27.7% (from 18.2% to approximately 13.1%), while the whole SFC migration reduced violations to 15.3%, which represents a reduction of 15.9%. These results can be explained by the blocking rate of each migration approach. Under the evaluated scenarios, the proposed iterative migration algorithm had a blocking rate of 36.4% – i.e., in 36.4% of the cases in which the migration was triggered, the proposed migration approach could not find a suitable migration plan, whereas for the Whole SFC migration blocking rate was of 64.3%. In these cases, VNFs of the services were not migrated from their current nodes, leading to SLA violations depending on how far the users moved from their requested service. Therefore, because the Iterative Migration could migrate more services than the Whole SFC migration, its impact on reducing SLA violations is greater.

To further analyze the impact of our migration proposal, we considered the processing delay of packets that were generated and processed through the SFC before a migration event and the delay of packets that were generated and processed at a time after the migration has finished. Fig. 12 shows that the Iterative Migration procedure reduced

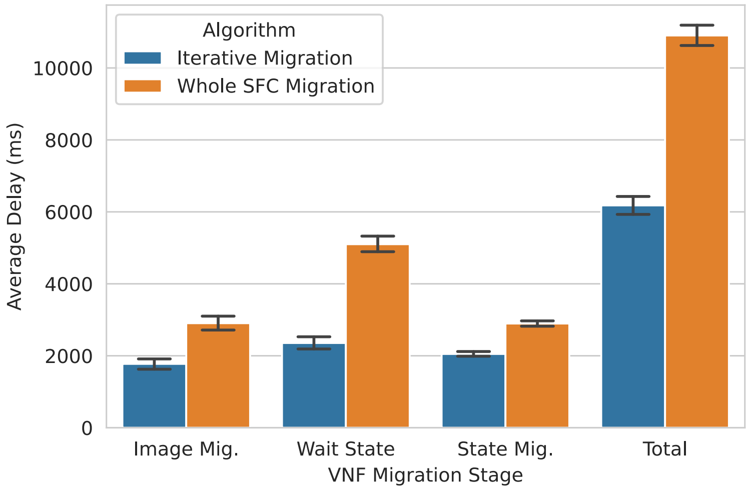


Fig. 14. Average delay for each stage of VNF migration.

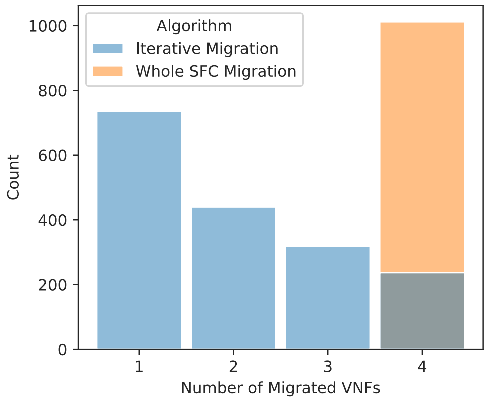


Fig. 15. Analysis of the number of migrated VNFs per SFC.

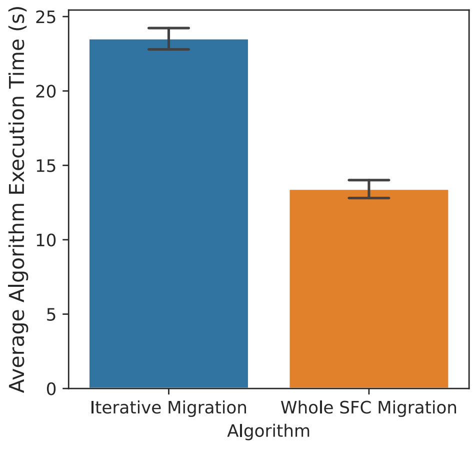


Fig. 16. Analysis of the average execution time of the migration approaches.

the average packet delay from 120.0 ms to 103.1 ms, while the Whole SFC Migration reduced packet delay from 123.2 ms to 94.9 ms, on average. Despite the higher blocking rate, the Whole SFC Migration achieves slightly better gains due to the fact that, by migrating all VNFs of the chain, the algorithm has more freedom to decide where to locate the VNFs. We also analyzed the average packet buffer time, which is the time that a packet had to wait until the VNFs migrations of an SFC were complete to be processed by those VNFs and traverse the chain. We only considered packets that had any buffer time in the average, excluding the ones that did not have to wait for VNF state migration, and, thus, would have a zero buffer time. As shown in Fig. 13, the average packet buffer time for the Iterative Migration was around 2.6 s, whereas, in the Whole SFC Migration, the average buffer time is 4.1 s, which has a greater buffer time due to the time required to execute the VNF state data transfer procedure for all VNFs of the SFC.

We also analyzed the average delay of each stage of VNF migration, as shown in Fig. 14. The results show that the Iterative Migration approach is significantly faster than the Whole SFC Migration in all migration stages. Despite having the same VNF configuration in both scenarios, the Whole SFC migration takes longer because more VNFs must be transferred simultaneously in comparison to the Iterative Migration, which leads to increased bandwidth overhead in the infrastructure.

The number of migrated VNFs per SFC was also analyzed to evaluate the effectiveness of the Iterative Migration approach. As shown in Fig. 15, the majority of SFC migrations only required the migration of a single VNF, reducing significantly the resource utilization in comparison with the approach that migrates the whole SFC. Only in 13.7% of the migrations the whole VNF chain – composed of all 4 VNFs – had to be migrated, which shows that the iterative approach is effective in saving migration resources by migrating only the required VNFs. Regarding the execution time, our proposed migration approach took, on average, around 23.5 s to find a migration plan, as shown in Fig. 16. The Whole SFC Migration approach takes less time since it only considers the migration of the whole set of VNFs, as opposed to the Iterative Migration, which must also look for the migration of subsets of VNFs of the SFC.

The experiment results showed that our proposed migration algorithm is capable of reducing SLA violation and packet processing caused by user mobility in the edge environment. The iterative approach significantly reduces resource usage by finding a suitable migration plan that mitigates performance problems by migrating the fewest amount of VNFs of the SFC.

The reduction in resource consumption is particularly relevant for the context being considered in this work, namely, Edge Computing environments. Resources available at the network edge, both communication and processing, are typically scarcer than those available at the network core and in data centers located in the cloud. Therefore, every solution aimed at such environments needs to be resource-aware and seek to optimize its usage. Our solution was motivated by this premise of resource scarcity and, therefore, aims to satisfy the user's quality requirements while efficiently using available resources.

Due to the heuristic nature of the algorithm, our proposal is also able to find suboptimal but effective solutions promptly, even with over 100 computing nodes. It is also important to mention that we consider the process of migrating VNFs from an SFC taking place in parallel to the execution of the service, in a similar way to soft handover in cellular networks. Therefore, the migration time results obtained in our experiments denote the maximum time it takes for the system properly meeting again the SLA for a given service, but the service continues to be provided, although with degraded quality, while the migration occurs. There is no interruption of the service provisioning, only a decrease in the quality of service delivered, in several cases to the point of violating the established contract (SLA).

5. Conclusion

In this work, we proposed a solution for SFC migration that addresses the challenges posed by dynamic and mobility-prone 5G-Edge environments. The solution includes a proactive method for estimating performance issues caused by user mobility and a novel, iterative algorithm for SFC migration. The algorithm aims at ensuring the user/application is provided with the required quality of service while simultaneously reducing the migration cost by moving the fewest number of VNFs of the SFC. It deals with the dynamism of resource availability at the edge by reserving the required resources before starting the migration. The proposed migration mechanism also manages the resource concurrency that might occur between VNF migrations of the same SFC. The algorithm supports the migration of both stateless and stateful VNFs. It identifies when the migration of the state of a VNF should start, establishing conditions to synchronize the state migrations of the VNFs and reduce the buffering overhead of packets.

It is important to mention that our approach is proactive only to some extent since it only predicts the next hop in the user movement. Using Markov chains, if the estimated service delay in a user's next movement is greater than the maximum delay established in the SLA, the migration algorithm will be triggered. In this sense, some other enhanced method, which better anticipates the user movement in his/her complete path (e.g., N movements before reaching the movement in which the SLA is violated), can improve the proactiveness of the method. In this initial version of our solution, we used Markov chains. In future work, we intend to incorporate and assess different methods for mobility prediction, for instance, based on machine learning techniques.

We also intend to extend the evaluation of our solution by comparing it to other SFC migration approaches in the literature. Finally, we plan to specify a full-fledged architecture encompassing all the software entities, such as monitoring and orchestration components, required to enable the SFC migration in 5g-Edge environments.

CRediT authorship contribution statement

Juan Lucas Vieira: Writing – review & editing, Writing – original draft, Validation, Software, Investigation, Conceptualization. **Evandro L.C. Macedo:** Writing – review & editing, Writing – original draft, Software, Conceptualization. **Anselmo L.E. Battisti:** Writing – review & editing, Writing – original draft, Software, Conceptualization. **Julia Noce:** Writing – review & editing, Writing – original draft. **Paulo F. Pires:** Writing – review & editing, Supervision, Conceptualization. **Débora C. Muchaluat-Saade:** Writing – review & editing, Supervision, Conceptualization. **Ana C.B. Oliveira:** Supervision, Project administration. **Flavia C. Delicato:** Writing – review & editing, Writing – original draft, Supervision, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Flavia Delicato reports financial support was provided by State of Sao Paulo Research Foundation. Flavia Delicato reports financial support was provided by Carlos Chagas Filho Foundation for Research Support of Rio de Janeiro State. Debora Muchaluat- Saade reports financial support was provided by Carlos Chagas Filho Foundation for Research Support of Rio de Janeiro State. Debora Muchaluat-Saade reports financial support was provided by National Council for Scientific and Technological Development. Flavia Delicato reports financial support was provided by National Council for Scientific and Technological Development. Paulo Pires reports financial support was provided by National Council for Scientific and Technological Development. Juan Lucas Vieira reports financial support was provided by Coordination of Higher Education Personnel Improvement. Debora Muchaluat-Saade reports financial support was provided by State of Sao Paulo Research Foundation. Juan Lucas Vieira has patent #Mobility-aware Iterative SFC Migration in a Dynamic 5G Edge Environment pending to Dell EMC. Anselmo Battisti has patent #Mobility-aware Iterative SFC Migration in a Dynamic 5G Edge Environment pending to Dell EMC. Evandro Macedo has patent #Mobility-aware Iterative SFC Migration in a Dynamic 5G Edge Environment pending to Dell EMC. Ana Oliveira has patent #Mobility-aware Iterative SFC Migration in a Dynamic 5G Edge Environment pending to Dell EMC. Julia Noce has patent #Mobility-aware Iterative SFC Migration in a Dynamic 5G Edge Environment pending to Dell EMC. Paulo Pires has patent #Mobility-aware Iterative SFC Migration in a Dynamic 5G Edge Environment pending to Dell EMC. Debora Muchaluat-Saade has patent #Mobility-aware Iterative SFC Migration in a Dynamic 5G Edge Environment pending to Dell EMC. Flavia Delicato has patent #Mobility-aware Iterative SFC Migration in a Dynamic 5G Edge Environment pending to Dell EMC. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

Acknowledgments

This work was partially funded by DELL EMC, CAPES, CNPq, FAPERJ, and FAPESP. Professors Débora Muchaluat-Saade, Flavia Delicato and Paulo Pires are CNPq Fellows. Professors Débora Muchaluat-Saade and Flavia Delicato are also FAPERJ Fellows.

Funding

This study was partially funded by DELL EMC (ID FEC 4717), FAPESP (grant number 2015/24144-7), and FAPERJ (grant number 200.972/2022).

References

[1] G. 5G Infrastructure Association, et al., The 5G infrastructure public private partnership: The next generation of communication networks and services, 2015, https://5g-ppp.eu/wp-content/uploads/2015/02/5G-Vision-Brochure-v1.pdf.
[2] I. Recommendation, E. 800: Terms and definitions related to quality of service and network performance including dependability, in: ITU-T August 2008, 2008, https://www.itu.int/rec/T-REC-E.800-200809-I/en.
[3] S. Yang, F. Li, S. Trajanovski, R. Yahyapour, X. Fu, Recent advances of resource allocation in network function virtualization, IEEE Trans. Parallel Distrib. Syst. 32 (2) (2020) 295–314.
[4] I. Benkacem, T. Taleb, M. Bagaa, H. Flinck, Optimal VNFs placement in CDN slicing over multi-cloud environment, IEEE J. Sel. Areas Commun. 36 (3) (2018) 616–627.

[5] N. Reyhanian, H. Farmanbar, S. Mohajer, Z.-Q. Luo, Joint resource allocation and routing for service function chaining with in-subnetwork processing, in: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, IEEE, 2020, pp. 4990–4994.

[6] A.L.É. Battisti, E.L.C. Macedo, M.I.P. Josué, H. Barbalho, F.C. Delicato, D.C. Muchaluat-Saade, P.F. Pires, D.P.d. Mattos, A.C.B.d. Oliveira, A novel strategy for vnf placement in edge computing environments, Future Internet 14 (12) (2022) 361.

[7] D. Zhao, G. Sun, D. Liao, S. Xu, V. Chang, Mobile-aware service function chain migration in cloud–fog computing, Future Gener. Comput. Syst. 96 (2019) 591–604.

[8] W. Liang, L. Cui, F.P. Tso, Low-latency service function chain migration in edge-core networks based on open Jackson networks, J. Syst. Archit. 124 (2022) 102405.

[9] Y. Ma, W. Liang, J. Li, X. Jia, S. Guo, Mobility-aware and delay-sensitive service provisioning in mobile edge-cloud networks, IEEE Trans. Mob. Comput. 21 (1) (2020) 196–210.

[10] B. Yi, X. Wang, M. Huang, K. Li, Design and implementation of network-aware VNF migration mechanism, IEEE Access 8 (2020) 44346–44358.

[11] N. Toumi, M. Bagaa, A. Ksentini, Machine learning for service migration: A survey, IEEE Commun. Surv. Tutor. 25 (3) (2023) 1991–2020, http://dx.doi.org/10.1109/COMST.2023.3273121.

[12] Y.-T. Chen, W. Liao, Mobility-aware service function chaining in 5G wireless networks with mobile edge computing, in: ICC 2019-2019 IEEE International Conference on Communications, ICC, IEEE, 2019, pp. 1–6.

[13] T.-M. Pham, Optimizing service function chaining migration with explicit dynamic path, IEEE Access 10 (2022) 16992–17002.

[14] B. Yi, X. Wang, M. Huang, A. Dong, A multi-criteria decision approach for minimizing the influence of VNF migration, Comput. Netw. 159 (2019) 51–62.

[15] Y. Qin, D. Guo, L. Luo, J. Zhang, M. Xu, Service function chain migration with the long-term budget in dynamic networks, Comput. Netw. 223 (2023) 109563.

[16] J. Chen, J. Chen, K. Guo, R. Hu, T. Zou, J. Zhu, H. Zhang, J. Liu, Fault tolerance oriented SFC optimization in SDN/NFV-enabled cloud environment based on deep reinforcement learning, IEEE Trans. Cloud Comput. (2024).

[17] A. Karim, J. Ema, T. Yasmin, P. Roy, M.A. Razzaque, Latency and cost-aware deployment of dynamic service function chains in 5 g networks, in: 2023 International Symposium on Networks, Computers and Communications, ISNCC, IEEE, 2023, pp. 1–6.

[18] Y. Yue, X. Tang, Z. Zhang, X. Zhang, W. Yang, Virtual network function migration considering load balance and SFC delay in 6 g mobile edge computing networks, Electronics 12 (12) (2023) 2753.

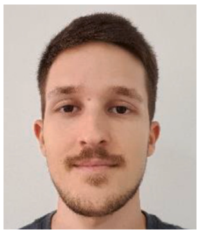
[19] ETSI, ETSI network functions virtualisation (NFV) release 2; management and orchestration; architectural framework specification, 2021, https://www.etsi.org/deliver/etsi_gs/nfv/001_099/006/02.01.01_60/gs_nfv006v020101p.pdf.

[20] H. Abu-Ghazaleh, A.S. Alfa, Application of mobility prediction in wireless networks using markov renewal theory, IEEE Trans. Veh. Technol. 59 (2) (2009) 788–802.

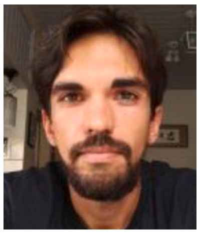
[21] J.R. Norris, Markov Chains, Cambridge University Press, 1997, http://dx.doi.org/10.1017/CBO9780511810633.

[22] Simpy - discrete event simulation for python, 2023, https://simpy.readthedocs.io/. (Accessed in 04 January 2023).

[23] D. Kotz, T. Henderson, I. Abyzov, J. Yeo, CRAWDAD Dartmouth/campus (v. 2009-09-09), http://dx.doi.org/10.15783/C7F59T.



Juan Vieira received his B.Sc. and M.Sc. degrees in Computer Science from Universidade Federal Fluminense (UFF), Rio de Janeiro, Brazil in 2017 and 2019, respectively. He is currently a doctorate student at the same university. His research interests include wireless networks, the Internet of Things and Software Defined Networks.



Evandro L.C. Macedo received his B.S. degree in Computing and Information Technology from State University of Rio de Janeiro (UERJ), Brazil in 2011. He acquired his Master's degree in Systems Engineering and Computing from the Systems Engineering and Computer Science Program (PESC), Federal University of Rio de Janeiro (UFRJ), Brazil in 2015. He is a Ph.D. student at PESC, UFRJ. His research interest includes modeling and performance analysis, information and computer security, Internet of Things, 5G, and Edge Computing.

Anselmo Luiz Éden Battisti holds a degree in Computer Science (2007) from Universidade Estadual do Oeste do Paraná, Brazil, and a master's degree in strategic business at the Pontifical Catholic University of Paraná, Brazil, and a master's degree in computer science from Fluminense Federal University in the area of multimedia systems. He is a Ph.D. student at Fluminense Federal University (UFF) in Brazil. His main research interests are 5G, Edge Computing and IoT.

Julia Noce holds a degree in Computer Science (2019) from Universidade Federal Fluminense and a master's degree in Computational Intelligence and Data Science (2022) from Pontifical Catholic University of Rio de Janeiro (PUC-Rio). She is currently a Data Science advisor at Dell EMC working on different Machine Learning and Artificial Intelligence projects.

Paulo F. Pires (D.Sc. COPPE/UFRJ 2002) is an Associate Professor at Fluminense Federal University, Brazil. His main research interests are at the intersection of Software Engineering and Distributed Systems. He has published more than 200 papers from internationally renowned journals, conference articles and book chapters, has twelve patents registered with the USPTO (United States) and coordinated several university-industry R&D partnerships projects. Dr. Pires has co-authored two books: "Middleware Solutions for the Internet of Things" (Springer, 2013) and "Resource Management for Internet of Things" (Springer, 2017). He is currently a member of the IEEE TC on Cybermatics, Associate Editor of the IEEE Transactions on Services Computing and IEEE Open Journal of the Communications Society and member of the editorial board of the International Journal of Computer Networks (CSC Journals). Dr. Pires has a CNPq research productivity scholarship since 2010 and is a member of IEEE and the Brazilian Computer Society (SBC).

Débora Christina Muchaluat Saade is a Full Professor at the Department of Computer Science of Fluminense Federal University (UFF) in Brazil. She received her Ph.D. in Computer Science in 2003 from Pontifical Catholic University of Rio de Janeiro (PUC-Rio). She is one of the founders and heads of the MidiaCom Research Lab (www.midiacom.uff.br). Her main research interests are multimedia systems, computer networks, IoT, smart grids and e-health. She has contributed to the design and development of the Ginga-NCL middleware, used in the Brazilian Digital TV Standard (ABNT NBR 15606-2) and IPTV services (ITU-T H.761).

Ana Cristina Bernardo de Oliveira holds a degree in Systems Engineering (1992), a degree in Electronic Engineering (1993) and a master's degree in Theory of Controls and Statistics (1997), from the Department of Electrical Engineering at the Pontifical Catholic University of Rio de Janeiro, Brazil. She is currently a Research Director and a Data Scientist at Dell EMC. She has experience in the areas of Machine Learning, Big Data and Applied Statistics.

Flavia C. Delicato is an Associate Professor at the Fluminense Federal University, Brazil. Her main research interests are Edge/Fog Computing, Internet of Things, sensor and network virtualization. She has 2 books and over 200 published papers. She coordinates and participates in R&D projects funded by companies such as EMC, Dell and Petrobras. Her research has also received funding from Brazilian and international development agencies, including CNPq, RNP, FAPESP, FAPERJ, CENPES, MCT, Fundacion Carolina and The Australian Agency for International Development (AusAID). She has served as General and Program Chairs at several national and international conferences. She currently serves as Associate Editor of the following journals: Ad hoc Networks, ACM Computer Surveys, IEEE Transactions on Service Computing, IEEE Open Journal of the Communications Society, ITU Journal on Future and Evolving Technologies and Journal of Interconnection Networks. Since 2017 she is a member of the IEEE Technical Committee on Smart World — SWTC (http://www.cybermatics.org/swtf/indexTC.html). Since 2020 she has been one of the Chairs of the IEEE Special Committee on Hyper-Intelligence (Hyper-Intelligence Technical Committee – HITC – https://ieee-hyperintelligence.org/member).