Deterministic Latency/Jitter-aware Service Function Chaining over Beyond 5G Edge Fabric

Hao Yu, Tarik Taleb, *Senior Member, IEEE,* and Jiawei Zhang

***Abstract*—Deterministic Networking (DetNet) has recently attracted much attention. It aims at studying the deterministic bounded latency and low latency variation for time-sensitive applications (e.g., industrial automation). To improve the quality of service (QoS) guarantee and make the network management efficient, it is desirable for Internet Service Provider (ISP) to obtain an optimal service function chain (SFC) provision strategy while providing deterministic service performance for the time-sensitive applications. In this paper, we will study the deterministic SFC lifetime management problem in beyond 5G edge fabric with the objective of maximizing the overall profits and ensuring the deterministic latency and jitter of SFC requests. We first formulate this problem as a mathematical model with the maximal profits for ISP. Then, the novel Deterministic SFC Deployment algorithm (Det-SFCD) and SFC Adjustment algorithm (Det-SFCA) due to traffic load variation are proposed to efficiently solve the SFC lifetime management problem. Extensive simulation results show that our proposed algorithms can achieve better performance in terms of SFC request acceptance rates, overall profits and latency variation compared with the benchmark algorithm.**

***Index Terms*—Deterministic Networking (DetNet), Network Function Virtualization (NFV), Service Function Chain (SFC), and Beyond 5G Edge Fabric**

I. Introduction

The emerging applications, e.g., video feeds from cameras that are used to control robots in Industrial Internet of Things (IIoT) environments [1] and autonomous automotive vehicles [2], foster the development of 5G and beyond 5G. For these applications, both high data rates and low communication latency should be ensured by using a dedicated mechanism in the 5G and beyond 5G network environments. Especially, the real-time communication within few milliseconds, needed by industrial applications [3], requires the networks to provide deterministic end-to-end connectivity to users. To this end, IETF Deterministic Networking (DetNet) Working Group [4] [5] defines an architecture focusing on layer 3 routed segments to provide a deterministic data path which aims at supporting deterministic worst-case bounds on latency, packet delay variation (jitter), and extremely low/zero packet loss. In this paper, we extend this concept into the field of softwarized network service management and that is to study the deterministic network service provisioning mechanism.

With the advent of network function virtualization (NFV) and software-defined networking (SDN) technologies, Internet Service Providers (ISPs) can deploy their network services flexibly and efficiently. Upon receiving user requests, ISPs have to configure the required virtual network functions (VNFs) into suitable physical servers and steer the traffic to traverse the VNFs concatenated in a specified order to comply with performance policy, which is defined as service function chain (SFC) [6] [7]. Optimal SFC embedding onto physical networks has drawn much attention in the recent literature. However, most of the existing work in the literature focused on maximizing network throughput/resource efficiency, regardless latency requirements; whereas maximizing network resource efficiency while keeping the service latency and latency variation (jitter) within a deterministic bound has received less attention. As shown in Fig. 1, in traditional networks, the end-to-end latency/jitter curves have a wide probability distribution with a long tail. The main objective of this research is to realize bounded end-to-end latency and delay variation with no long tails in an end-to-end converged network; ultimately supporting deterministic services. Another goal is to increase the profits of ISPs by optimizing the resource allocation and placement of VNF instances while ensuring deterministic latency performance. Moreover, due to the highly dynamic nature of network traffic load, it is a challenge to embed SFC requests with deterministic latency bounds and lower jitter during the SFC lifetime.

As shown in Fig.2, edge nodes $\{V_1, V_2, ..., V_7\}$ are equipped with a certain amount of processing resources and switching ability. At cell sites (CSs) side, user devices generate SFC requests over time. And the SFC requests are featured with latency requirements, which are comprised of communication latency and VNF processing latency. Once a new SFC request arrives, optimal path selection and processing resource allocation should be performed in order to meet its latency requirement. For example, SFC 2 is newly arrived in the networks and its end-to-end latency requirement is $L_{E2E} \leqslant 15\ ms$. After considering the network load status and distance between source and destination nodes, Path 2 is selected with communication latency $L_c = 2\ ms$. Then the VNF processing budget is $13\ ms$ and we need to determine how to allocate appropriate processing resources to the VNF instances of this SFC and to make sure that the processing latency $L_p^1 + L_p^2 + L_p^3 + L_p^4 \leq 13\ ms$ and the total cost for VNF processing is minimized.

In this paper, given the SFC requests and finite physical

Hao Yu is with the Center for Wireless Communications, Oulu University, Oulu 90570, Finland. E-mail: hao.yu@oulu.fi. (*Corresponding Author*)

Tarik Taleb is with the Center for Wireless Communications, Oulu University, Oulu 90570, Finland, and also with the Department of Computer and Information Security, Sejong University, Seoul 05006, South Korea. E-mail: tarik.taleb@oulu.fi.

Jiawei Zhang is with the State Key Laboratory of Information Photonics and Optical Communications, Beijing University of Posts and Telecommunications, 100876, China E-mail: zjw@bupt.edu.cn.

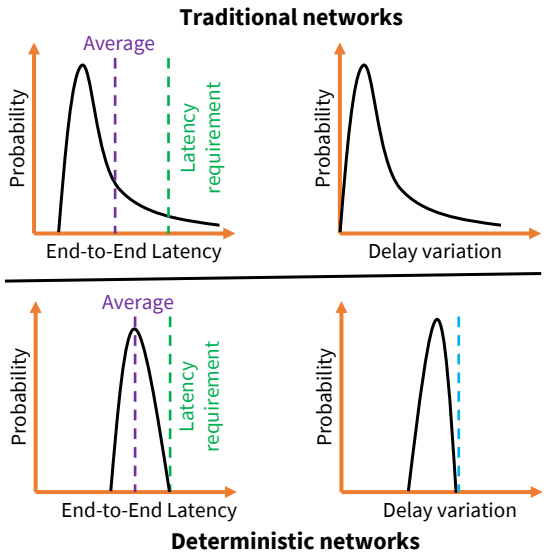


Fig. 1: Comparison of latency performance between traditional networks and deterministic networks.

resources in 5G edge environments, we try to address the deterministic SFC lifetime management (Det-SFCLM) problem. It is easy to keep the QoS of single SFC to be deterministic. However, it is challenging to ensure most SFCs deployed in the network to be deterministic under the restricted resource capacity during their lifetime. Thus, we try to solve this problem from two aspects: (i) how to optimize the resource allocation and path selection for SFC deployment; (ii) how to adjust resource allocation to ensure the bounded service latency and jitter under the traffic variation, which can ultimately maximize the overall incomes for ISP. These two aspects form the whole procedure of lifetime management for SFCs. For the first sub-problem, we try to solve it in two directions: (i) improve the service acceptance ratio (i.e., increase the revenue derived from providing services to users); (ii) reduce the resource consumption by optimizing the resource allocation to VNF instances (i.e., reduce the network cost for ISPs). Given that the propagation, transmission latency are deterministic, we need to bound the non-deterministic VNF processing latency in order to achieve an overall deterministic end-to-end latency and jitter for time-sensitive services. For the second sub-problem, we will investigate the optimal VNF scaling up/down scheme in response to the traffic variation to keep the bounded latency by considering the historical network load, which will help avoid resource bottleneck and reduce network congestion.

We apply the Det-SFCLM problem to the 5G edge environments where an open and smart radio access network (RAN) architecture, as specified by the O-RAN alliance [8], is assumed to be part of the environments. The main idea of O-RAN concept includes RAN function disaggregation via open interfaces and open-source platforms, which facilitates an open and virtualized 5G edge networking ecosystem. By employing NFV concepts in O-RAN architecture, the baseband processing units in the new architecture can be implemented as RAN VNFs instantiated at the edge servers.

We extend our previous work in [9] to further address the problem of provisioning SFC requests with deterministic latency and jitter. First, we formulate it as a mathematical model and the objective is to maximize the overall profits for ISP over a time period. Then, we propose two novel algorithms, named Deterministic SFC Deployment (Det-SFCD) algorithm and Deterministic SFC Adjustment (Det-SFCA) algorithm. Det-SFCD obtains the optimal placement and processing resource allocation for VNF instances by considering defined deployment cost based on an extended shortest path algorithm. Det-SFCA optimizes the processing resource adjustment to make the latency, caused by the traffic variation, stable by considering historical network load. The contributions are listed as follows:

- We study the QoS-based (i.e., deterministic latency and jitter) end-to-end SFC management in 5G edge environments including radio access and core networks. We then formulate the SFC lifetime management problem by a mathematical model with the objective of optimizing the resource allocation and VNF instance placement.
- We propose efficient composition algorithms for the automated deployment of SFCs, which consists of Det-SFCD algorithm which is designed for optimal path selection and resource allocation for VNF instances, and Det-SFCA algorithm which tries to adjust resource allocation of VNF instances on the basis of the traffic variation in order to ensure deterministic latency.

The remainder of this paper is organized as follows. Section II introduces some related work. Section III presents the system model. Section IV formulates the deterministic SFC provisioning problem and our proposed algorithms are presented in Section V. The performance evaluation results are discussed in Section VI. Section VII concludes this paper.

II. Related work

[10] The SFC problem has been widely studied and many solutions have been proposed. From the perspectives of network optimization objectives, some works studied how to maximize the QoS of SFCs under the network resource restriction, e.g., minimizing the total service latencies. The authors in [11] introduced “COLAP”, a predictive framework to place the participating VNFs of a SFC in a cloud environment while optimizing the service latency. In summary, this work has considered the service latency as the main metric while overlooking the VNF instances’ dependencies and availability metrics. Gouareb et al. [12] studied the problem of VNF placement and routing across the physical hosts to minimize overall latency defined as the queuing delay within the edge clouds and in network links. Jinke et al. [13] formulated a joint communication and computation resource allocation problem with the objective of minimizing the weighted-sum latency of all mobile services. Then, a closed-form optimal task splitting strategy is derived as a function of the normalized backhaul communication capacity and the normalized cloud computation capacity. Qu et al. [14] formulated a Mixed Linear Programming Model (MILP) and a heuristic approach to minimize the SFC end-to-end delays while overcoming the scalability of an optimization model. The authors have

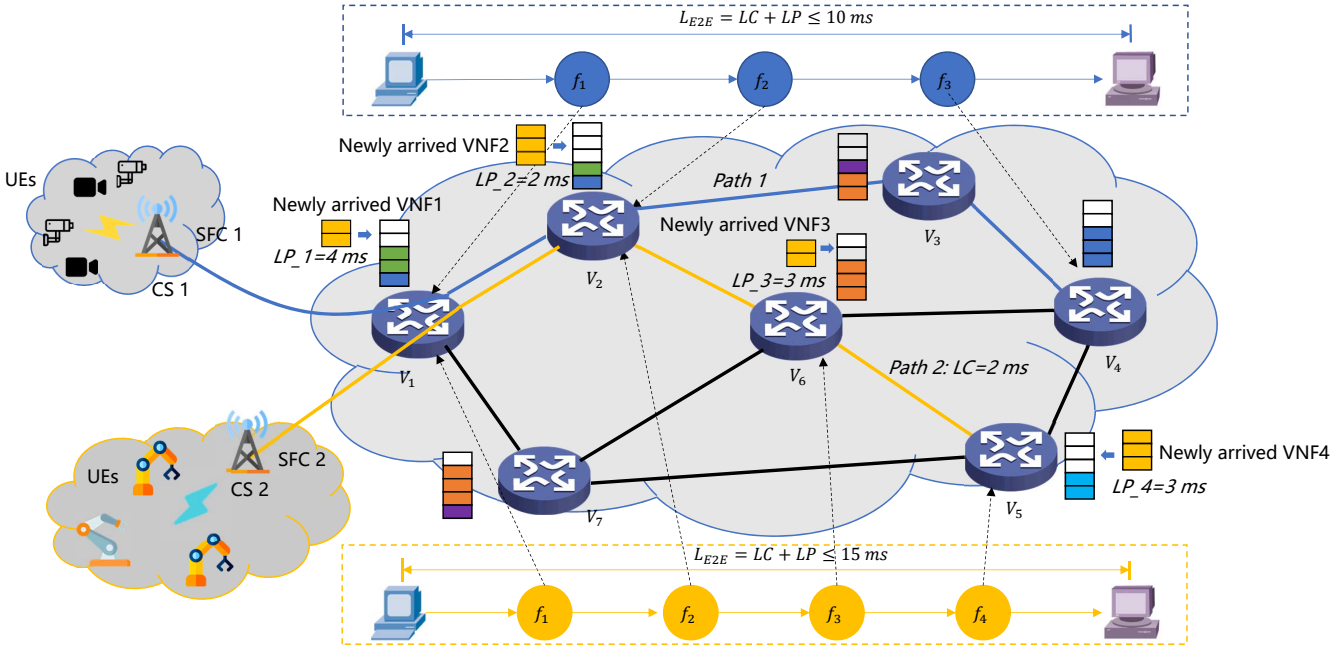


Fig. 2: Embedding SFC requests in 5G edge networks.

proposed an algorithm that selects a subset of VNFs that are needed to generate a SFC and its redundant.

On the other hand, some works considered the network utilization, e.g., computational resources, as the main metric to achieve better resource utilization. The authors in [15] formulated the VNF placement problem as bin-packing and open Jackson network problems to maximize the compute resource utilization. Benkacem et al. [16] formulated the VNF placement problem as two Linear Integer problem models, aiming at minimizing the cost and maximizing the quality of experience (QoE) of the virtual streaming service. They then ensured an optimal tradeoff solution between the cost efficiency and QoE by applying the bargaining game theory. Bari et al. [17] solved the VNF placement problem with a simplified set of constraints, which only considers the deployment cost, the resource requirement, and the processing delay, and discards the placement constraints, such as the VNF chaining, reliability, and delay tolerance constraints. Taleb et al. [18] proposed a VNF placement algorithm to minimize the cost in terms of the total number of instantiated VNFs in a cloud environment. The authors in [19] and [20] solved the VNF placement and routing optimization problem by using mixed integer linear programming (ILP) models. Nevertheless, the ILP and MILP algorithms can only be solved offline. Due to the high complexity, it is usually applied in small-scale network. Otherwise, many works have solved the service function chaining problem by proposing efficient heuristics which can be used to configure and manage SFCs online with higher scalability. Basically, the SFC placement problem can be partitioned into two sub-problems: i) a VNF placement problem and ii) a routing problem. A fast heuristic framework, called Holu, that can efficiently solve the power-aware and delay constrained joint VNF placement and routing (PD-VPR) problem is proposed in [21]. It addresses these two sub-problems sequentially, which improves the system performance in terms of total power consumption and acceptance rate.

In terms of the SFC lifetime management, the SFC operations can be also partitioned into two parts: embedding phase and adjusting phase. The authors in [22] studied the SFC problem in geo-distributed cloud system by proposing SFC eMbedding APproach (SFC-MAP) and VNF Dynamic Release Algorithm (VNF-DRA) to efficiently embed SFC requests and optimize the number of placed VNF instances. Junjie et al. [23] jointly optimized the deployment of SFCs and the readjustment of in-service SFCs while considering the trade-off between resource consumption and operational overhead. They designed a column generation (CG) model for solving the optimization problem. Xincai et al. [24] derived the requested instances with adaptive processing capacities and called two other algorithms for new instance assignment and service chain rerouting, respectively, while achieving good competitive ratios. The problem of dynamic placement reconfiguration of 5G User Plane Functions (UPFs) in a MEC ecosystem was studied in [25], a scheduling technique based on Optimal Stopping Theory (OST) was proposed to adapt to changes in user locations while ensuring QoS and network operator expenditures reduction. The authors in [26] studied the joint SFC deployment and resource management problem (JSDRM) in heterogeneous edge environments to minimize the total system latency and proposed a scheme based on a game model to jointly deploy SFCs and manage resources.

Researchers have been addressing various aspects of SFC. For instance, they propose different optimization models and heuristic solutions for the SFC placement problem and SFC lifecycle management problem. Despite all the significant

literature studies on SFC, SFC deployment and adjustment still need to be further investigated and exploited to satisfy the deterministic latency requirements as shown in Fig. (1). Minimizing network latency or maximizing network utilization, traditionally from a single side or both sides, cannot meet the new requirements exposed by 5G and beyond 5G networks. Deterministic latency performance, rather than latency minimization, for SFCs can fit the network performance requirements of time-critical services and should be the new direction of network resource optimization. Deterministic latency provides a more stable latency distribution which can benefit the time-sensitive tasks from the upper layer. In addition, the work in [27] is the first to use stochastic network calculus (SNC) to study the end-to-end delay bound with given traffic demand and resources. It proposes a solution to find the amount of resources that should be allocated with given traffic distribution and end-to-end delay bound, which is beneficial for the network service provisioning under deterministic end-to-end delay requirements.

Different from the above-mentioned literature, this paper focuses on the optimal SFC lifetime management, i.e., SFC deployment and SFC adjustment, under the deterministic end-to-end latency and jitter requirements in 5G edge environments. Latency minimization is not the objective of this paper. The proposed algorithms try to keep the end-to-end latency experienced by SFCs within a bounded time interval below the latency requirements.

III. System model

A. Network model

We start with a system description that identifies the scope of our study. In the paper, we consider a 5G edge network comprised of edge nodes and cell sites (CSs) forming a multi-cell coverage area for mobile users. Each edge node is equipped with limited computational capacity utilized for running 5G RAN and core network VNFs, where RAN VNFs perform the RAN protocol stack (e.g., baseband processing function). We denote by $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ the physical networks consisting of N physical edge nodes, $\mathcal{N} = \{1, 2, ..., N\}$ and E physical links connecting edge nodes in $\mathcal{N}$, i.e., $\mathcal{E} = \{1, 2, ..., E\}$. We use $n, m \in \mathcal{N}$ to indicate nodes and $(n, m) \in \mathcal{E}$ to represent the link connecting node n and m. The edge nodes are equipped with different amounts of CPUs, RAM, etc. The node capacity of each edge node in terms of processing and memory is given as C_n^{cpu} and C_n^{mem}. For each direct link $(n, m) \in \mathcal{E}$, the bandwidth of (n, m) is expressed as $B_{(n,m)}$.

B. SFC model

A total number of K SFC requests arrive during $\mathcal{T} = \{1, 2, ..., T\}$. We use $\hat{\mathcal{G}}_k = (\mathcal{V}_k, \mathcal{L}_k), k \in \mathcal{K}$ to denote the service function graph of SFC request k. Each SFC $\hat{\mathcal{G}}_k$ is defined as a vector $\{s_k, d_k, a_k, q_k, \lambda_{k,t}, L_k\}$ to dictate the property of this SFC. s_k and d_k correspond to the source and destination nodes of SFC k. a_k and q_k represent the arrival time and departure time. $\mathcal{T}_k = \{t : a_k \leq t \leq q_k\}$ denotes the duration time that SFC k is running in the networks. Since each SFC $k \in \mathcal{K}$ consists of a given number of

TABLE I: Notation and variables

Notation	Description
	Topology
$\mathcal{G}$	Substrate networks
$\mathcal{N}$	Set of nodes in substrate networks
$\mathcal{E}$	Set of links in substrate networks
n, m	physical nodes in the network $\mathcal{G}$
(n, m)	physical links in the network $\mathcal{G}$
C_n^{cpu},C_n^{mem}	CPU/memory capacity of node n
$B_{(n,m)}$	Bandwidth resource capacity of link (n, m)
	Service requests
$\mathcal{T}$	Set of time slots in the system
$\mathcal{K}$	Set of SFC requests
$\hat{\mathcal{G}}_k$	Graph of SFC request k
$\mathcal{V}_k$	Set of VNFs in SFC k
$\mathcal{L}_k$	Set of virtual links in SFC k
$\mathcal{T}_k$	The time duration of SFC k
s_k	Source node of SFC k
d_k	Destination node of SFC k
a_k	Arrival time of SFC k
q_k	Departure time of SFC k
$\lambda_{k,t}$	Data rate of SFC k at time t
L_k	E2E latency requirement of SFC k
$v_{k,\hat{n}}, v_{k,\hat{m}}$	VNFs of SFC k
$l_{k,(\hat{n},\hat{m})}$	Virtual link between VNFs of SFC k
$m_{k,\hat{n}}$	Memory requirement of SFC k
$\rho_{k,\hat{n}}$	The number of CPU cycles for processing one information bit by VNF $v_{k,\hat{n}}$
	Decision Variables
$x_{k,n}^{\hat{n}}$	Whether VNF $v_{k,\hat{n}}$ of SFC k is placed in edge node n
$\pi_{k,t}^{\hat{n}}$	The amount of allocated processing resources for VNF $v_{k,\hat{n}}$ of SFC k at time t
$\hat{\pi}_k^{\hat{n}}$	The amount of allocated memory resources for VNF $v_{k,\hat{n}}$ of SFC k
$y_{k,(n,m)}^{(\hat{n},\hat{m})}$	Whether virtual link $l_{k,(\hat{n},\hat{m})}$ of SFC k traverses physical link (n, m)
$y_{k,n}^{(\hat{n},\hat{m})}$	Whether virtual link $l_{k,(\hat{n},\hat{m})}$ of SFC k traverses physical node n
$\eta_{l,t}^{(\hat{n},\hat{m})}$	The allocated bandwidth for virtual link $l_{k,(\hat{n},\hat{m})}$ of SFC k at time t
$\tau_{k,\hat{n},t}^{p}$	Processing latency of VNF $v_{k,\hat{n}}$ of SFC k at t
$l_{k,t}^{p}$	Processing latency of SFC k at t
$d_{k,(\hat{n},\hat{m}),t}^{prop}$	Propagation latency of virtual link $l_{k,(\hat{n},\hat{m})}$ of SFC k at time t
$d_{k,(\hat{n},\hat{m}),t}^{trans}$	Transmission latency of virtual link $l_{k,(\hat{n},\hat{m})}$ of SFC k at time t
$\tau_{k,(\hat{n},\hat{m}),t}^{c}$	Communication latency of i^{th} virtual link of SFC k
$l_{k,t}^{c}$	Communication latency of SFC k at t
$l_{k,t}$	Experienced E2E latency by SFC k at time t
	System Parameters
ϵ	Coefficient of latency variation bound of SFC
α_1, α_2	Coefficient of CPU/memory resource cost
β	Coefficient of bandwidth cost
δ	Coefficient of revenue from data rate
ω	Coefficient of revenue from latency requirement
N_k	The number of aggregated radio resource blocks (RBs) allocated to the users in SFC k
a_j	RAN Layer 1 computational resource model-specific constant
$i_{MCS,k}$	Indices of the MCSs of SFC k
θ_1	Scaling factors of the Layer 1
θ_2	Scaling factors of the high-layer VNFs

ordered VNFs except source and destination nodes, we use $\mathcal{V}_k$ to denote the set of VNFs in SFC k where $v_{k,\hat{n}}, v_{k,\hat{m}}$ is the n^{th} and m^{th} VNFs in SFC k. We assume that the processing latency of a VNF instance is only related to the processing resource allocated to it. The amount of resources allocated to a certain VNF can be arbitrary provided that the overall processing latency of this SFC can satisfy the latency requirement. We assume the memory resources requested by a VNF is constant, we use a vector $\{m_{k,1}, m_{k,2}, ..., m_{k,\hat{n}}, ...\}$ to denote the memory requirement of SFC k. In addition, there is a set of virtual links connecting the source node s_k, ordered VNFs and the destination node d_k, we denote the link between the n^{th} VNF and the m^{th} VNF in SFC k as $l_{k,(\hat{n},\hat{m})} \in \mathcal{L}_k$. Each SFC supports one service associated with a data rate $\lambda_{k,t}$ at time t which is an aggregated bit rate of multiple users belonging to this SFC.

Note that the processing resources (e.g., CPU cores) are statistically multiplexed if multiple VNF instances are deployed on the same CPU core. The sharing of processing resources is not considered since it cannot ensure deterministic processing latency. As shown in [28], as more VNFs share the same CPU, the CPU access latency experienced by each VNF increases substantially. The processing latency of VNFs that are already mapped will be degraded by the newly instantiated VNF in the same CPU core, which leads to the latency uncertainty of existing SFCs. Thus, in this paper, we assume that different SFCs can not share the same types of VNF instances, even if the VNF instances with the same types exist in an edge node. Each VNF instance occupies the exclusive processing resources and can only belong to a certain SFC.

IV. Deterministic SFC provisioning problem

A. Problem Description

In a NFV-enabled edge system, an ISP should make optimal planning for SFC request deployment to maximize its profit while ensuring the deterministic requirements of SFC requests. The problem can be described as: **Given:** a physical network topology $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ and a set of SFC requests $\mathcal{K}$. For each SFC request, **determine:** 1) how to select path between source and destination nodes and place VNF instances along the path, 2) how to allocate processing and bandwidth resources for corresponding VNFs and traffic, 3) how to adjust resource allocation when traffic load varies, to 4) **maximize:** the overall profits of ISP from running SFC requests, meanwhile, 5) **ensure:** deterministic latency performance. The traffic of SFC request will traverse a series of ordered VNF instances and the path selected on which the ordered VNF instances are mapped will influence the resource consumption on edge nodes and physical links. How to select suitable path and allocate resources for SFC requests remain a challenge for deterministic latency performance and maximum resource efficiency.

B. Problem Formulation

The SFC deployment basically consists of path selection and resource allocation. These two parts are actually interactive to each other and should be coordinated to achieve the objective mentioned above.

VNF instance deployment: We denote by binary variable $x_{k,n}^{\hat{n}}$ the placement of VNF $v_{k,n} \in \mathcal{V}$, $x_{k,n}^{\hat{n}} = 1\ iff$ VNF $v_{k,\hat{n}}$ is placed in edge node $n \in \mathcal{N}$, otherwise, $x_{k,n}^{\hat{n}} = 0$. Based on the observation that one edge node can host multiple VNF instances and one VNF instance can only run on top of one edge node, the placement constraint of VNF instance $v_{k,\hat{n}}$ can be given as follows

$$\sum_{n \in \mathcal{N}} x_{k,n}^{\hat{n}} = 1, \forall k \in \mathcal{K}, \forall v_{k,\hat{n}} \in \mathcal{V}_k, k \in \mathcal{K} \tag{1}$$

Since the computing capability of edge node is shared by all VNF instances that are placed on it, the total CPU and memory capability allocated to VNF instances can not exceed the total capability of the edge node. We then have

$$\sum_{k \in \mathcal{K}, \hat{n} \in \mathcal{V}_k} x_{k,n}^{\hat{n}} \cdot \bar{\pi}_k^{\hat{n}} \leq C_n^{mem}, \forall n \in \mathcal{N}, \tag{2}$$

$$\sum_{k \in \mathcal{K}, \hat{n} \in \mathcal{V}_k} x_{k,n}^{\hat{n}} \cdot \pi_{k,t}^{\hat{n}} \leq C_n^{cpu}, \forall n \in \mathcal{N}, t \in \mathcal{T}_k \tag{3}$$

where we define $\pi_{k,t}^{\hat{n}}$, $\bar{\pi}_k^{\hat{n}}$ to indicate the amount of processing and memory resources allocated to VNF $v_{k,\hat{n}}$ of SFC k at time t. The unit of processing resource is set as the CPU cycles times by the number of CPU cores and the unit of memory resource is GB in the paper, and $\hat{\pi}_k^{\hat{n}} = \{0, m_{k,\hat{n}}\}$. C_n^{cpu}, C_n^{mem} represent the resource capability of CPU and memory in edge node n.

Traffic routing: For SFC k, we define the binary variable $y_{k,(n,m)}^{(\hat{n},\hat{m})}$ and $y_{k,n}^{(\hat{n},\hat{m})}$ to denote whether the $l_{k,(\hat{n},\hat{m})} \in \mathcal{L}_k$ traverses link $(n,m) \in \mathcal{E}$ and the node $n \in \mathcal{N}$, respectively. If $(n,m) \in \mathcal{E}$ is traversed by $l_{k,(\hat{n},\hat{m})} \in \mathcal{L}_k$, $n, m \in \mathcal{N}$ must be traversed as well. Then the following routing constraint must be ensured as

$$y_{k,n}^{(\hat{n},\hat{m})} y_{k,m}^{(\hat{n},\hat{m})} = 1\ if\ y_{k,(n,m)}^{(\hat{n},\hat{m})} = 1 \tag{4}$$

$$\begin{aligned} \sum_{l_{k,(\hat{n},\hat{m})} \in \mathcal{L}_k} \sum_{m \in \mathcal{N}} \left(y_{k,(n,m)}^{(\hat{n},\hat{m})} - y_{k,(m,n)}^{(\hat{n},\hat{m})} \right) \\ = \begin{cases} 1, & n = s_k \\ -1, & n = d_k \\ 0, & otherwise \end{cases}, k \in \mathcal{K} \end{aligned} \tag{5}$$

Constraint (4) ensures the physical nodes and the virtual link that connects them are consistent. Constraint (5) guarantees that the links on the path to embed SFC k are connected head-to-tail. If node $n \in \mathcal{N}$ is selected to serve the VNF $\hat{n} \in \mathcal{V}$ of SFC k, this node must be traversed as follows

$$x_{k,n}^{\hat{n}} \leq y_{k,n}^{(\hat{n},\hat{m})}, \forall n \in \mathcal{N}, \forall v_{k,\hat{n}} \in \mathcal{V}_k, l_{k,(\hat{n},\hat{m})} \in \mathcal{L}_k, k \in \mathcal{K} \tag{6}$$

Also, since the bandwidth resource of physical link (n,m) are shared by the virtual links that are mapped on it, the total bandwidth consumed by these virtual links can not exceed the total bandwidth resource of physical link (n,m). Firstly, we define the following real variable: $\eta_{k,t}^{(\hat{n},\hat{m})}$ to denote the amount of bandwidth allocated to the virtual link between n^{th} and

m^{th} VNF of SFC k. Then, we add the following constraints. Constraints (7) ensures that the sum of bandwidths allocated to virtual links can not exceed the total bandwidth of physical link (n, m) at time t.

$$\sum_{k\in\mathcal{K}, l_{k,(\hat{n},\hat{m})}\in\mathcal{V}_k} y_{k,(n,m)}^{(\hat{n},\hat{m})} \times \eta_{k,t}^{(\hat{n},\hat{m})} \leq B_{(n,m)}, \quad \forall (n,m) \in \mathcal{E}, t \in \mathcal{T}_k \tag{7}$$

Deterministic Latency

For a SFC, latency will be incurred by data processing in edge nodes and data transmission in physical links accordingly, i.e., processing latency and communication latency. The service latency of a SFC is determined by both resource requirement of SFC and the amount of resources allocated to it.

(1) Processing Latency of VNFs: The SFC we consider in the 5G edge fabric contains different types of VNFs, e.g., RAN and core network functions, etc. Different kinds of network functions work in a different way and the processing resource requirements depend on different factors.

• RAN VNF modelling

For RAN functions, the RAN VNF performs layers of RAN protocol stack (e.g., baseband processing). The computational complexity of the specific RAN VNF depends on the user's traffic load (e.g., RAN Layer 3 and Layer 2), while Layer 1 processing is performed per assigned Resource Block (RBs) and is mainly dependent on channel condition. The condition in the channel dictates the appropriate coding rate and modulation for the data to be transmitted successfully which leads to different computational demand on Layer 1. Thus, the computational complexity of Layer 1 functions depend on the amount of RBs assigned and Modulation and Coding Scheme (MCS). The higher layer RAN VNFs (Layer 3 and Layer 2) and other common VNFs (e.g., core network functions) processing are user load dependent and the processing requirements depend on the aggregated users' data rates.

A VNF is usually instantiated by associating with a certain combination of resources (e.g., CPU, RAM and etc.) and the RAN VNF processing latency can be calculated as the function of CPU frequency allocated if the number of RBs and MCS indices are known. Therefore, given the amount of CPU frequency $\pi_{k,t}^{\hat{n}}$ allocated to Layer 1 RAN VNF of SFC k, if we assume $v_{k,\hat{n}}$ to be Layer 1 RAN VNF, according to experimental results running on the general purpose processors (GPPs), the processing time of Layer 1 functions $v_{k,\hat{n}}$ of SFC k regarding with the allocated CPU frequency $\pi_{k,t}^{\hat{n}}$ at time t is given as in [29]

$$\tau_{k,\hat{n},t}^{p} = \frac{\theta_1 N_k}{(\pi_{k,t}^{\hat{n}})^2} \sum_{j=0}^{2} a_j (i_{MCS,k})^j, \quad v_{k,\hat{n}} = RAN\ Layer\ 1, t \in \mathcal{T}_k, k \in \mathcal{K} \tag{8}$$

where N_k denotes the number of aggregated resource blocks (RBs) allocated to the users of SFC k. a_j is the Layer 1 computational resource model-specific coefficient, which is related to the type of RAN functionality, e.g., modulation/demodulation, encoding/decoding, and the corresponding values are given in [29]. $i_{MCS,k}$ is the indices of the MCSs of SFC k as defined in 3GPP TS 38.214 [30]. For the sake of simplicity, we assume that all users within SFC k are assigned with the same MCS indices. θ_1 is the scaling factor of the Layer 1 [31]. Equation (8) provides a closed-form approximation for processing time in RAN function, which can be used to determine the required number of CPUs and their working frequency for provisioning each VNF.

• Other VNFs

For the other common network function types, such as core network, gateway, load balance, etc, the computation resource model is different from the RAN functions, which is user data rate related. Considering each computation task of an SFC k can be processed in the edge servers, we can use a two-field notation $A_{k,\hat{n}} = \{\lambda_{k,t}, \rho_{k,\hat{n}}\}$ to denote the computation task of the VNF $\hat{n}$ of SFC k, where $\lambda_{k,t}$ is the input data-size (in bits) per second at time slot t, which also represents the data rate of this SFC and $\rho_{k,\hat{n}}$ denotes the number of CPU cycles that are required to compute one-bit data by this VNF $v_{k,\hat{n}}$. Note that, the value of $\rho_{k,\hat{n}}$ varies from the type of VNF $v_{k,\hat{n}}$. Similar to [32], the processing latency of the other VNFs $v_{k,i}$ in the SFC k is given as:

$$\tau_{k,\hat{n},t}^{p} = \theta_2 \frac{\rho_{k,\hat{n}} \lambda_{k,t}}{\pi_{k,t}^{\hat{n}}}, \quad \forall v_{k,\hat{n}} \in \mathcal{V}_k / \{RAN\ Layer\ 1\}, t \in \mathcal{T}_k, k \in \mathcal{K} \tag{9}$$

where $\lambda_{k,t}$ is the aggregated users' data rate under this SFC, θ_2 is the scaling factor of the functions on the other layers [31] with regard to RNA Layer 1. Thus, The total processing latency for VNFs of SFC k is:

$$l_{k,t}^{p} = \sum_{v_{k,\hat{n}}\in\mathcal{V}_k} \tau_{k,\hat{n},t}^{p} \tag{10}$$

(2)Communication Latency of Virtual Links: The communication latency of each SFC consists of the propagation latency, transmission latency. Based on the study in literature [22], the communication latency of SFC k on virtual link $l_{k,(\hat{n},\hat{m})} \in \mathcal{L}_k$ can be formulated as:

$$\tau_{k,(\hat{n},\hat{m}),t}^{c} = d_{k,(\hat{n},\hat{m}),t}^{prop} + d_{k,(\hat{n},\hat{m}),t}^{trans} \tag{11}$$

Hereby, the first term of equation (11) indicates the propagation latency on physical links that virtual link $l_{k,(\hat{n},\hat{m})}$ traverses by, which is related with the distance between the adjacent physical edges. The second term indicates the transmission latency, which is calculated by dividing the size of transmitted packet with the bandwidth capacity allocated to the virtual links:

$$d_{k,(\hat{n},\hat{m}),t}^{trans} = \frac{b_k}{\eta_{k,t}^{(\hat{n},\hat{m})}}, k \in \mathcal{K}, l_{k,(\hat{n},\hat{m})} \in \mathcal{L}_k, t \in \mathcal{T}_k \tag{12}$$

Then the communication latency of SFC k is formulated as:

$$l_{k,t}^{c} = \sum_{l_{k,(\hat{n},\hat{m})}\in\mathcal{V}_k} \tau_{k,(\hat{n},\hat{m}),t}^{c} \tag{13}$$

Finally, considering the deterministic E2E latency requirement L_k of SFC k, the E2E latency constraint is given as:

$$L_k(1-\epsilon) \leq l^p_{k,t} + l^c_{k,t} = l_{k,t} \leq L_k(1+\epsilon) \tag{14}$$

where ϵ represents the latency variation (i.e., jitter) that network services can tolerate, $\epsilon \in (0,1)$.

Profit Model:

1) Cost Model: The cost of SFC k can be defined similar to the one in [33] as follows:

$$C_k = \sum_{t\in\mathcal{T}_k} \Bigg(\sum_{l_{k,\hat{n}}\in\mathcal{V}_k} (\alpha_1 \pi^{\hat{n}}_{k,t} + \alpha_2 \bar{\pi}^{\hat{n}}_k) + \sum_{l_{k,(\hat{n},\hat{m})}\in\mathcal{L}_k} \beta \eta^{(\hat{n},\hat{m})}_{k,t} \Bigg), k \in \mathcal{K} \tag{15}$$

where α_1, α_2 denotes the cost factor of allocating one resource unit of CPU and memory, β represents the cost factor of allocating one bandwidth unit.

2) Revenue Model: We define the revenue of each SFC $k \in \mathcal{K}$ within its lifetime as follows:

$$R_k = \sum_{t\in\mathcal{T}_k} (\delta \lambda_{k,t} + \omega / L_k), k \in \mathcal{K} \tag{16}$$

As the E2E latency requirement L_k can be seen as the most important QoS indicator for the safety-critical service and the performance assurance that it can provide to users, so we take into account the L_k as the part of revenue that SFC k can make. Besides E2E latency, data rate $\lambda_{k,t}$ should also been considered as another QoS indicator by which ISP can charge users. It will consume more network resources (i.e., processing and bandwidth resources) to provision a service with higher data rate than lower data rates while ensuring the same E2E latency requirement L_k.

Thus, for each SFC k, the overall profit of SFC k is formulated as:

$$P_k = R_k - C_k \tag{17}$$

3) The total profits of the system: The total profits of the system, denoted by P, is formulated by the summation of profits of all the SFCs deployed as follows:

$$P = \sum_{k\in\mathcal{K}} P_k \tag{18}$$

Thus, the deterministic SFC lifetime management problem in this paper is formulated as an optimization problem which maximizes the overall profits of system:

$$\max P \tag{19}$$

$$s.t. (1-18) \tag{20}$$

The formulated problem is NP-hard as a result of the non-reasonable calculation time. To prove the NP-hardness of this problem, we need to to reduce it to a well-known NP-hard problem. We simplified our problem and considered only one SFC which contains a series of VNFs. The problem can be denoted by P where the grouping of multiple VNFs and placing these VNFs into edge nodes is similar to the knapsack problem, which is known to be NP-hard. In this problem, the edge nodes can be considered as knapsacks while the VNFs are considered as different objects. We try to reduce the resource cost while the total CPU and memory capacity of VNFs within an edge node do not exceed the node capacity, which is equivalent to the cost in the knapsack problem. Besides, we also consider the latency requirement which will increase complexity. Thus, deterministic SFC lifetime management is an NP-hard problem. In addition, in this paper, we consider the CPU resource sharing can not ensure the processing latency, the CPU resource allocation should be discrete, thus it is hard to find an integer solution to make the E2E service latency equal to the latency requirement exactly. We propose a heuristic solution to solve this problem in acceptable timescales.

V. Deterministic SFC Lifetime Management

In general, we can maximize the overall profits for ISP from two aspects: 1) accepting more SFC requests to increase the revenue; 2) reduce the resource cost caused by optimally allocating network resources to SFC requests. In addition, we need to ensure the latency experienced by SFC requests to be deterministic.

We divide the procedures on SFC lifetime management into two phases: SFC deployment and SFC adjustment, which are solved by the Det-SFC deployment (Det-SFCD) algorithm and the Det-SFC adjustment (Det-SFCA) algorithm, respectively. In SFC deployment phase, 1) optimal paths need to be selected to avoid the resource bottleneck when deploying SFCs, ultimately increase the SFC acceptance rate; 2) VNF instances need to be created optimally to minimize the resource costs while ensuring the latency requirements. In SFC adjustment phase, optimal VNF instance scaling up/down scheme should be designed in order that the latency variation is controlled within a small range.

A. Deterministic SFC Deployment (Det-SFCD) algorithm

1) Path Calculation based on Deployment Cost: To derive optimal paths for deploying SFCs in the network, we define "deployment cost" for physical links and nodes which will indicate overloaded edge/node, so that we balance the network load and ultimately reduce resource bottlenecks. We use $c^{cpu}_{k,n}$, $c^{mem}_{k,n}$ and $c^{bw}_{k,n,m}$ to represent the deployment costs of CPU and memory on node n and bandwidth on link (n,m) when deploying SFC k as follows.

$$c^{cpu}_{k,n} = \phi_n \cdot \frac{\underset{n\in\mathcal{N}}{max} C^{cpu}_n}{C^{cpu}_n r^{cpu}_{k,n}} \tag{21}$$

$$c^{mem}_{k,n} = \phi_n \cdot \frac{\underset{n\in\mathcal{N}}{max} C^{cpu}_n}{C^{mem}_n r^{mem}_{k,n}} \tag{22}$$

$$c^{bw}_{k,(n,m)} = \phi_{(n,m)} \cdot \frac{\underset{(n,m)\in\mathcal{E}}{max} B_{(n,m)}}{B_{n,m} r^{bw}_{k,(n,m)}} \tag{23}$$

where $r^{cpu}_{k,n}$, $r^{mem}_{k,n}$ and $r^{bw}_{k,(n,m)} \in [0,1]$ denote the residual CPU ratio (i.e., the amount of residual CPU cores divided by the the total amount of CPU cores) on node n, residual memory ratio on node n and remaining bandwidth ratio on

link (n, m). In Eqs. (21)-(23), when the network load is low, the values of $c^{cpu}_{k,n}$, $c^{mem}_{k,n}$ and $c^{bw}_{k,(n,m)}$ are relatively small and increase slowly. Whereas if the resource consumption are comparative to resource capabilities, the values will be very large and increase quickly. Besides network load, we also take into account the node or link capacity itself. With $\underset{n\in\mathcal{N}}{max} C^{cpu}_n / C^{cpu}_n$, the nodes with higher capacities and lower network load are more likely to be selected. Thus, $c^{cpu}_{k,n}$, $c^{mem}_{k,n}$ and $c^{bw}_{k,n,m}$ can be used to facilitate the path selection. In addition, we use the coefficients ϕ_n and $\phi_{(n,m)}$ to set the criticalities of different nodes and links, since some nodes and links are located in critical places, which may have much background traffic. Furthermore, we define the load status of a path by summing up the deployment costs along the path. The deployment cost of the path l when deploying SFC request k is defined as:

$$R_{k,l} = \sum_{n\in\mathcal{N}_l} max\{c^{cpu}_{k,n}, c^{mem}_{k,n}\} + \sum_{(n,m)\in\mathcal{E}_l} c^{bw}_{k,n,m} \qquad (24)$$

where $\mathcal{N}_l$ and $\mathcal{E}_l$ denote the nodes and links along the selected path l. According to Eq.(24), we can infer that there are no bottleneck at edge nodes, nor at links along the selected path if the cost $R_{k,l}$ is small when deploying SFC k. Conversely, if the cost $R_{k,l}$ of the selected path is very large, there must exist some bottleneck at the edge nodes or at some links which are overloaded along the path l. We therefore need to choose other paths with smaller cost to deploy SFC k. We then sort the candidate paths in the ascending order according to $R_{k,l}$.

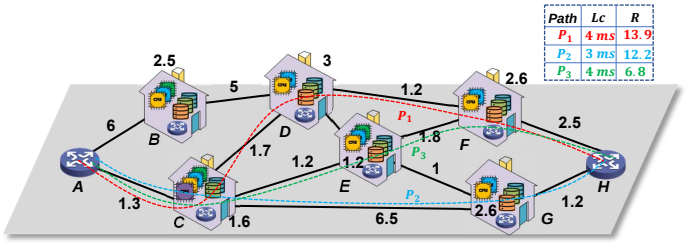


Fig. 3: Weighted topology with deployment costs.

Leveraging the traditional shortest path algorithm, we extend it to obtain the shortest paths using as weight the deployment cost $R_{k,l}$ from the source node s to the destination node d on the graph $\mathcal{G}$. As shown in Algorithm 1, we first set the weight of this topology as link length (distance between two adjacent edge nodes) and use KSP algorithm [34] to get Γ shortest paths. Then we update the deployment cost of each physical node and link. For deployment cost of physical nodes, we set the value of deployment with $max\{c^{cpu}_{k,n}, c^{mem}_{k,n}\}$. As we aim at finding the bottleneck nodes, we consider a node being a bottleneck node when its CPU load $c^{cpu}_{k,n}$ and memory load $c^{mem}_{k,n}$ are high. We set deployment cost of physical links with $c^{bw}_{k,(n,m)}$. After updating the deployment cost of the whole topology, we calculate the $R_{k,l}$ for each candidate path. As shown in Fig.3, even if the latency path 2 is lowest (three hops) among all the candidate paths between node A and H, the deployment cost of path 2 is higher compared with the cost of path 3, which means path 2 is overloaded on some link

Algorithm 1: Path calculation with Extended Dijkstra's Algorithm.

Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E}), s_k, d_k$
Output: Γ *candidate paths with* $R_{k,l}$

```
Perform KSP algorithm to get Γ shortest paths between s_k and d_k
Update the deployment cost of physical nodes with max{c^cpu_{k,n}, c^mem_{k,n}}
Update the deployment cost of physical links with c^bw_{k,n,m}
for l ∈ [1, Γ] do
    Calculate the deployment cost R_{k,l} of path l
Sort the paths in ascending order according to R_{k,l}
```

(e.g., link(C,G)). Therefore, when deploying an SFC request between nodes A and H, path 3 should be selected even it is not the shortest path, which is the best path for SFC k with least deployment cost. In this case, the deployment cost has the higher priority to be considered than path latency. The benefits of deploying the SFC into the path with least deployment costs are twofold: 1) it can exclude the paths with bottleneck nodes or links to increase the acceptance ratio; 2) it can avoid making nodes or links to be the bottlenecks in the networks.

2) SFC Deployment Scheme with Minimal Resource Cost: Next, we need to decide each VNF instance's size for SFC k, i.e., the processing resource allocation to each VNF instance, based on the chosen path. As generally known, each VNF has its specified resource demand according to the user load and VNF type. The latency of a VNF instance is decided by the amount of resource demand and the resources allocated to it. It is obvious that allocating processing resources to different VNF instances can lead to different processing latency, which will in turn result in different resource costs. Thus, given a SFC request k, creating VNF instances with minimal processing resources while ensuring a certain E2E service latency remains a question to be solved. After a suitable path is selected for SFC k, the propagation and transmission latency are known, which can be calculated according to this route. Note that, to focus on the processing allocation, we set the bandwidth allocation in line with the data rate of SFC k. Given a total E2E service latency requirement, in other words, latency budget, we need to determine the latency distribution on each VNF instance of SFC k. Fig.4 gives an example. We assume the total latency budget of SFC k is 15 ms, the communication latency on the selected path 3 is 4 ms, thus the remaining latency budget for VNF1/VNF2/VNF3 are 11 ms. For VNF 1 (e.g., Layer 1 RAN VNF), the required processing resources are relatively larger than that of VNF 2&3 (e.g., Layer 2&3 RAN VNF). As we can see, the resulted latency of allocating 1 CPU core to VNF instance 1/2/3 are 8.5/4.5/2.5 ms, respectively. To derive the CPU core allocation combination of VNF instances within SFC k, we will calculate l^p_k for all the combinations of CPU core allocation options as shown in CPU-latency-cost table of Fig.4. We then choose a combination of CPU core allocation with minimal resource cost among the combinations in which the resulted latency is also near to L_k. In this case,

the CPU core allocation (2 cores, 2 cores, 2 cores) for VNF instance 1/2/3 results in the minimal resource cost 3.9 $ with a VNF processing latency $l_k^p = 10.1\ ms < 11\ ms$. It shall be noted that the CPU resource allocation is discrete in terms of the amount of CPU cores; thus it can not usually derive a latency value exactly equal to latency requirement.

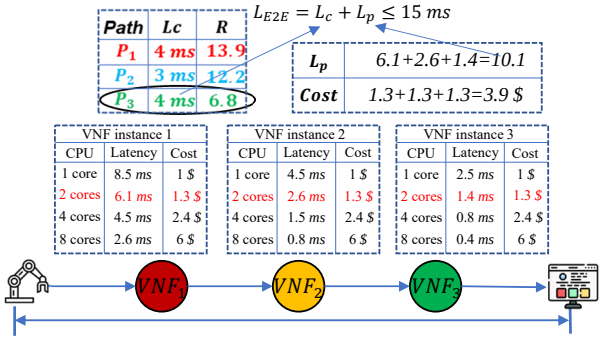


Fig. 4: CPU core allocation to VNF instances

The pseudocode of the Det-SFCD algorithm is shown in Algorithm 2 which is designed to solve the joint SFC placement and resource allocation problem. For a newly arrived SFC request k, we first calculate the deployment cost of CPU, memory and bandwidth according to the network load status and update the topology with the deployment costs in Line 1-5. With Algorithm.1, we then obtain Γ available paths with deployment costs in Line 6. Based on the path p_k, optimal VNF processing resource allocation scheme will be applied into SFC k in Line 8. In Line 9, the system will check if the selected path can meet the resource requirements in terms of CPU, memory and bandwidth demand. If the path p_k has enough resources for SFC k, we next need to determine the physical nodes to instantiate the VNFs on along the path. Given the allocated CPU core and memory of each VNF, we try to map the VNF instances into physical nodes in a load-balancing way. Specifically, we place the VNF instance with higher CPU demand in the node with more CPU resources, while keeping the VNF order along the SFC. After embedding the SFC k successfully, we update the network status.

Algorithm 2: Det-SFCD

Input: SFC set k, weighted topology $\widetilde{\mathcal{G}}$
Output: SFC deployment scheme
1 **for** *each* $n \in \mathcal{N}, (n,m) \in \mathcal{E}$ **do**
2 $c_{k,n}^{cpu} \leftarrow$ The deployment cost on CPU of n at current *network status*
3 $c_{k,n}^{mem} \leftarrow$ The deployment cost on memory of n at current *network status*
4 $c_{k,n,m}^{bw} \leftarrow$ The deployment cost on bandwidth of (n,m) at current *network status*
5 Update the topology $\widetilde{\mathcal{G}}$ with $c_{k,n}^{cpu}$, $c_{k,n}^{mem}$ and $c_{k,n,m}^{bw}$
6 $P_k \leftarrow \Gamma$ candidate paths
7 **for** p_k *in* P_k **do**
8 $Q(\mathcal{V}_k) \leftarrow$ Optimal resource allocation scheme along the selected path p_k
9 $Bool \leftarrow$ Check whether p_k satisfies SFC k in terms of CPU, memory and bandwidth allocation
10 **if** $Bool == true$ **then**
11 Embed the VNF instances (along with virtual links) into the nodes in a load-balancing way
12 Update the *network status*
13 **Return True**
14 **Return False**

B. Deterministic SFC Adjustment (Det-SFCA) algorithm

In this section, we try to solve the SFC adjustment problem caused by the traffic variation during the SFC lifetime. To this end, we propose a deterministic SFC adjustment algorithm with the objective of minimizing the latency variation. First, we need to adjust the resource allocation (i.e., CPU core allocation) to follow the traffic load variation of SFCs, so that the E2E latency requirements are met. Second, we consider the historical network load when adjusting the SFCs, to avoid network congestion and reduce adjusting failure ultimately.

As the traffic load of SFCs change over time, the resource demand also change according to the traffic load which, in turn, affects the E2E service latency. If the changed service latency cannot satisfy the constraint (14), the corresponding resource adjustment should be performed to control the latency variation, i.e., scaling up/down VNF instances. To achieve VNF instance scaling up/down optimally, we need to solve two problems: 1) which VNF instance(s) should be scaled up/down?, and 2) how much processing resources should be provisioned to this VNF instance(s)? For the first problem, we need to take into account the current residual resources of edge nodes and historical network load status to decide VNF instance(s) to be scaled up/down. Basically, when network load is relatively high, we need to design the scaling scheme carefully to avoid network congestion. The network load indicator is defined to describe the variation of network load. We use $\psi_n(t)$ to denote the past network load of physical node n from time t.

$$\psi_n(t) = \sum_{k \in \mathcal{D}_n} (\lambda_{k,t} - \lambda_{k,t-T}) \tag{25}$$

where $(\lambda_{k,t} - \lambda_{k,t-T})$ represents the traffic variation of SFC k during $(t-T, t]$. D_n denotes the set of SFCs that are embedded in physical node n. Since the network throughput could not change rapidly in a short time interval, $\psi_n(t)$ can indicate the trend of the network load in the next time units. If the value of $\psi_n(t)$ is positive, the network load in physical node n is supposed to increase in the future. While the value of $\psi_n(t)$ being around 0 means that the network load in physical node n keeps stable and decreasing network load leads to $\psi_n(t) < 0$. When the service latency of SFC k increases beyond the latency requirement, that is, $l_{k,t} > L_k(1+\epsilon)$, we should scale up this SFC k according to the network load indicator $\psi_n(t), n \in \mathcal{D}_k$. We choose the VNF instance(s) located in a physical node with lower $\psi_n(t)$ to scale up. For example, as shown in Fig.(5), the three SFCs are featured with different traffic profiles and are all partly deployed on node n. Taking

SFC 1 as an example, the VNFs of this SFC are deployed on node m, n, o. If it needs to be scaled up at time t, we will first calculate the ψ_m, ψ_n, ψ_o for each node, we then find that ψ_n at time t is the minimum among the three nodes, which means the network load will decrease in the following time slots in node n. Thus we should scale up VNF 1.2 to fit the latency requirement of SFC 1 in a load-balancing way. With network load indicator $\psi_n(t)$, we can avoid overloading some nodes and reduce the network bottleneck by using complementary traffic profiles between different SFCs.

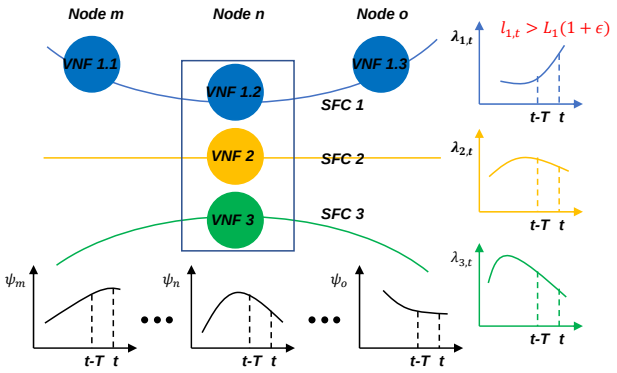


Fig. 5: SFC adjustment.

As shown in Algorithm 3 (scaling up case), the actual service latency $l_k(t)$ of SFC k and $\psi_n(t)$ are updated based on the network status at time t at Line 1. If the service latency of SFC k exceeds service latency $L_k(1+\epsilon)$ (i.e., traffic load of SFC k increases, $(\lambda_{k,t} > \lambda_{k,t-1})$), we need to determine which VNF instance(s) to scale up. Let D_k denote the physical nodes in which SFC k is embedded. Since VNF instance scaling up/down will consume extra time and resources [35], scaling as fewer VNF instances as possible is beneficial for service continuity. Det-SFCA will first sort the nodes in D_k with ascending order in terms of $max\{c^{cpu}_{k,n}, c^{mem}_{k,n}\}$. We set a threshold ξ to control the scaling of VNFs. We scale up the VNFs in the descending order of residual resources of nodes in $\mathcal{D}_k$, that is, we try to scale up the VNFs in the node with more residual capacity.

To derive the new CPU core allocation scheme for the VNF instance(s) in node $n \in \mathcal{D}_k$ with minimal $\psi_n(t)$, the CPU-latency table for SFC k is updated according to the traffic load at time t. Based on the CPU-latency value, new CPU core allocation should be applied to satisfy the latency constraints: $L_k(1-\epsilon) < l_k(t) \leqslant L_k(1+\epsilon)$. We assume that there is a maximum of CPU cores for VNF instance, if it fails to make $l_k(t)$ within the latency range above, iterate this procedure until the latency constraint is met in Line 6-14. If there is not node n whose $max\{c^{cpu}_{k,n}, c^{mem}_{k,n}\} \leq \xi$, we need re-sort the nodes in $\mathcal{D}_k$ in terms $\psi_n(t)$ and scale up VNFs of SFC k until the latency constrains is satisfied. For the scaling down case, the principle is to scale down the VNFs in the nodes whose residual capacity is relatively small or network load is increasing.

Algorithm 3: Det-SFCA (scaling-up case)

Input: SFC k at time t, weighted topology $\widetilde{\mathcal{G}}$
Output: SFC adjustment

1 For SFC k at t,
2 Update $l_k(t)$, $max\{c^{cpu}_{k,n}, c^{mem}_{k,n}\}$ and $\psi_n(t)$
3 **if** $l_k(t) > L_k(1+\epsilon)$ **then**
4 Update the path latency and re-calculate the latency-CPU table for SFC k
5 Sort the nodes in $\mathcal{D}_k$ with ascending order in terms of $max\{c^{cpu}_{k,n}, c^{mem}_{k,n}\}$
6 **for** $n \in \mathcal{D}_k$ **do**
7 **if** $max\{c^{cpu}_{k,n}, c^{mem}_{k,n}\} \leq \xi$ **then**
8 Increase CPU core allocation for VNFs iteratively and calculate new E2E latency according to new CPU core scheme
9 **if** $L_k(1-\epsilon) < l_k(t) \leqslant L_k(1+\epsilon)$ **then**
10 Scale up this VNF instance(s) with new CPU core allocation option
11 Update the *network status*
12 return **True**
13 **else**
14 Scale up VNF instance(s) with highest CPU cores
15 **else**
16 **Break**
17 Re-sort the nodes in $\mathcal{D}_k$ with ascending order in terms of $\psi_n(t)$
18 **for** $n \in \mathcal{D}_k$ **do**
19 Increase CPU core allocation for VNFs iteratively and calculate new E2E latency according to new CPU core scheme
20 **if** $L_k(1-\epsilon) < l_k(t) \leqslant L_k(1+\epsilon)$ **then**
21 Scale up this VNF instance(s) with new CPU core allocation option
22 Update the *network status*
23 return **True**
24 **else**
25 Scale up VNF instance(s) with highest CPU cores
26 return **False**

C. Deterministic SFC Lifetime Management (Det-SFCLM)

Finally, we will introduce the deterministic SFC lifetime management solution, as shown in Algorithm 4. We set the total time period as $\mathcal{T}$. At time t, the system will detect the events that happen in the networks. We divide the events into three types: arrival events, departure events and traffic load variation events. The SFCs which are associated with these events will be put into $\mathcal{K}_{i,t}$, $\mathcal{K}_{o,t}$ and $\mathcal{K}_{v,t}$, respectively. For arrival events, we sort the incoming SFC requests with descending order in terms of latency requirements to let the SFC request with stricter latency requirement be deployed first in Line 5. We then use the Det-SFCD algorithm to perform the

Algorithm 4: Det-SFCLM

Input: SFC set $\mathcal{K}$, weighted topology $\widetilde{\mathcal{G}}$
Output: Resource allocation during SFC lifetime

1 **for** $t \in \mathcal{T}$ **do**
2 Update weight (deployment cost) of topology $\widetilde{\mathcal{G}}$
3 Collect events at time t
4 Divide the events into three classes: arrival events $\mathcal{K}_{i,t}$, departure events $\mathcal{K}_{o,t}$, traffic load variation events $\mathcal{K}_{v,t}$.
5 $\mathcal{K}_{i,t} \leftarrow$ Sort SFCs with arrival events in terms of latency requirements in descending order
6 **for** $k \in \mathcal{K}_{i,t}$ **do**
7 Perform SFC deployment with **Det-SFCD** algorithm for SFC k
8 $\mathcal{K}_{o,t} \leftarrow$ Sort SFCs with traffic load variation events in terms of latency requirements in descending order
9 **for** $k \in \mathcal{K}_{v,t}$ **do**
10 Perform SFC deployment with **Det-SFCA** algorithm for SFC k
11 $\mathcal{K}_{o,t} \leftarrow$ Sort SFCs with departure events
12 **for** $k \in \mathcal{K}_{o,t}$ **do**
13 Release corresponding resource for SFC k

TABLE II: Notations used in the proposed algorithms.

Notation	Description
$r^{cpu}_{k,n}$, $r^{mem}_{k,n}$, $r^{bw}_{k,(n,m)}$	CPU, memory and bandwidth remaining rate of node n and link (n,m)
$c^{cpu}_{k,n}$, $c^{mem}_{k,n}$, $c^{bw}_{k,(n,m)}$	Deployment cost of CPU and memory on node n, bandwidth on link (n,m)
$R_{k,l}$	The overall deployment cost of embedding SFC k in path l
D_n	The set of SFC requests embedded in node n
D_k	The set of nodes that the VNFs of SFC k are deployed on
$l_k(t)$	The actual experienced latency of SFC k at t
$\psi_n(t)$	Historical network load of node n
$\mathcal{P}_k$	Current path of SFC k
T	Traffic sample period

SFC deployment in Line 7. For traffic variation events, we also deal with the SFC requests with stricter latency requirements first using the Det-SFCA algorithm in Line 9-10. For departure events, we just need to release the corresponding network resources in Line 12-13.

D. Complexity Analysis

In Det-SFCD, the complexity of calculating the costs of nodes and links is no more than $O(|\mathcal{V}| + |\mathcal{E}|)$. Executing the shortest path algorithm in physical topology $G = (\mathcal{V}, \mathcal{E})$ involves of a complexity in the order of $O(|\mathcal{E}| + |V|log|V|)$. Given that Γ candidate paths and I_k VNFs in a SFC, the total complexity of Det-SFCD is in the order of $O(\Gamma I_k(|\mathcal{E}| + |V|log|V|))$.

In Det-SFCA, for a SFC k that needs to be scaled up/down based on the latency violation, and assuming that the path length is $|P_k|$ and the SFCs that are embedded on each node along this path are $s_n, n \in P_k$, the time complexity of calculating the network load indicator is $O(|P_k|s_n), n \in P_k$. As Det-SFCA needs to scale this SFC until its service latency meets the latency constraints, the maximum time complexity for VNF instance scaling is $O(|P_k|(I_k + s_n))$.

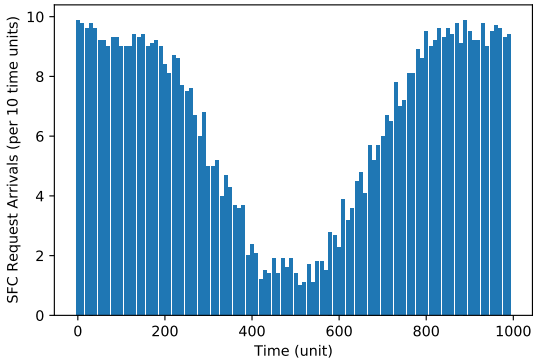


Fig. 6: SFC request arrival rate over time.

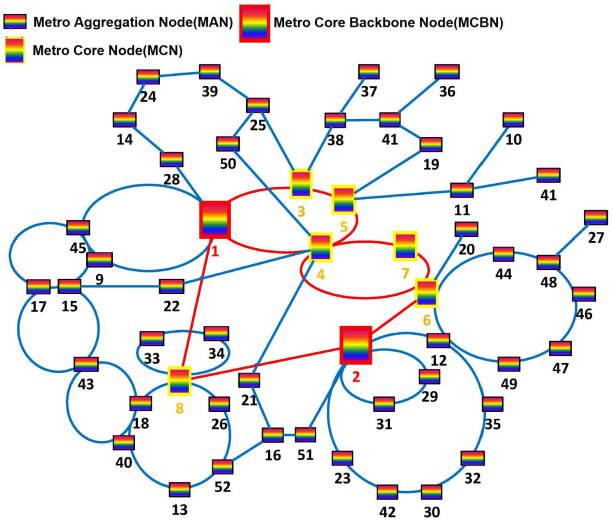
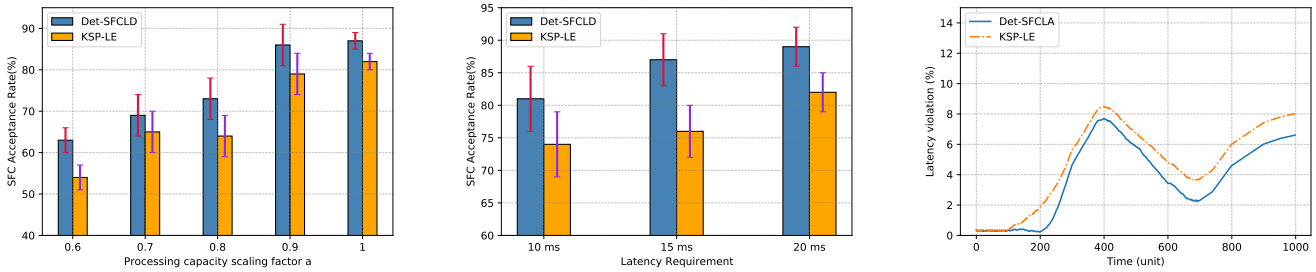


Fig. 7: Network Topology.

VI. Performance Evaluation

A. Simulation Setup

For the performance evaluation, We consider a reference metro-regional network with 52 nodes, comprised of 2 Metro Core Backbone Nodes (MCBNs), 6 Metro Core Nodes (MCNs), and 44 Metro Aggregation Nodes (MANs) and 72 bidirectional links as shown in Fig.7. In the topology, we select the 44 MANs as edge nodes and the other nodes act as switching nodes. Each edge node is associated with three cell sites (not shown in the figure for the sake of clarity). The memory capacity and CPU capacity of each edge node are set differently according to different node types, as shown in Table III. The maximum number of CPU cores permitted to be allocated per VNF instance is set to eight. The bandwidth capacity per link is 100 Gbps (e.g., 10 * optical wavelengths



(a) (b) (c)

Fig. 8: (a) The comparison of SFC request acceptance rate vs. scaling factor a; (b) The comparison of SFC request acceptance rate vs. latency requirement; (c) Latency violation over time.

at 10Gbps). The radio configuration of CSs are in line with the RAN VNF parameters specified in [31].

In the simulation, the source nodes of SFC requests are selected randomly from the edge nodes (44 MANs), while the destination nodes of SFC requests are randomly set from the switching nodes (eight MCNs&MCBNs). In order to capture the dynamic load, the arrival rate of SFC requests follows the distribution [22] as shown in Fig.6. The lifetime of SFC requests obeys an exponential distribution with an average of 100 time units. We assume that all SFCs are running with 4 VNFs (i.e., Layer 1 RAN VNF, Layer 2&3 RAN VNFs, 5G core VNF, and common VNF) [36]. For each SFC request, the resource block N_k, mean data rate $\lambda_{k,t}$(Mbps), memory (MB) are set as randomly distributed between [50, 100], [10, 100] and [100,500], respectively [31]. The E2E latency requirement for each SFC request is set as 10 ms, 15 ms and 20 ms [37].

We set the simulation period as 1000 time units and repeat the simulations in 20 epochs to eliminate contingency, in each epoch a set of SFCs arrive and leave the environment. After an existing SFC leaves, the corresponding network resources are released and the $network\ status$ is accordingly updated. The coefficients ϕ_n, $\phi_{n,m}$ are set randomly between [1,2] among the nodes and links.

B. Algorithm to use for comparison

Since there is no existing work studying the SFC configuration with deterministic latency and jitter, here-under we briefly introduce a straightforward SFC configuration algorithm that we use as a comparison term against our proposed algorithms.

- K Shortest Path-Latency Equalization (KSP-LE): This algorithm uses ksp algorithm without considering deployment to obtain k available shortest paths for the SFC deployment. For the CPU core allocation, it distributes the processing latency budget on VNF instances equally, which does not consider the resource cost on processing latency. In the adjustment phase, it scales up/down the VNF instances with the internal order until the E2E service latency is met.

C. Result analysis

In Fig.8(a), the mean acceptance rate of SFC requests achieved in each algorithm is plotted and that is for different processing capacity scaling factor. The acceptance rate of Det-SFCD is higher. Since the Det-SFCD algorithm considers the deployment cost when selecting suitable paths for SFC requests, this reduces the resource bottleneck and enhances the SFC request acceptance rate. Compared to our proposed algorithm, KSP-LE pays no attention to the remaining resources in the network. It simply chooses the shortest paths. This increases the probability of embedding failures. In Fig.8(a), we also investigate on how the network capacity affects the SFC acceptance rate by setting the scaling factor a of CPU resources. From the figure, we observe that as the scaling factor a increases from 0.6 to 0.9, the SFC acceptance rate

TABLE III: Simulation Parameter Settings.

Description	Value
System Parameters	
Network Topology	TIM Metro-Regional network
Number of edge nodes	44
Number of physical links	72
CPU capacity of MAN/MCN/MCBN	32/64/128 $\times 10^2$cores
Memory capacity of MAN/MCN/MCBN	16/32/64 $\times 10^2$GB
Bandwidth capacity of links	100 Gbps
Maximum CPU core for VNF instance	8
CPU frequency	2 GHz
RAN VNF Computational Model	
Upper layer scaling factor, θ_2	2
Layer 1 scaling factor, θ_1	1
Model [29]-specific constant, a_0	32.583
Model [29]-specific constant, a_1	1.072
Model [29]-specific constant, a_2	0.03
Average MCS index, $i_{MSC,k}$	16
SFC Parameters	
SFC arrival rate	Fig.6
Lifetime of each SFC	$X \sim E(\frac{1}{100})$
Packet size	64 Bytes
Number of VNFs per SFC request	4
Resource block N_k	[50, 100]
Memory demand of VNF instance	$[100, 500]$MB
Mean user rate, $\lambda_{k,t}$	[10, 100] Mbps
Maximum tolerated latency, L_k	$\{10, 15, 20\}ms$
ρ_2 for Layer 2&3 functions	0.2
ρ_3 for 5G core functions	0.2
ρ_4 for common functions	0.1
Simulation Parameters	
Execution period of Det-SFCLM	1000 time units
Jitter threshold ϵ	10 %
T	5 time units

increases from the range about 58-64% to the range about 96-100%. This is because if the edge nodes are equipped with more resource, SFC requests are more likely to be served in a load balancing way. Moreover, when the scaling factor a exceeds 0.9, the impact of network capacity on the acceptance rate is not obvious. We also present the influence of latency requirement L_k of SFC on the acceptance rate of algorithms. As shown in Fig.8(B), in case of the Det-SFCD algorithm, as the latency requirement becomes less strict, the acceptance rate increases from a range of 77-83% to 83-93%. This is attributable to the fact that lower latency requirements lead to higher CPU resource requirement, which decreases, in turn, the overall acceptance rate.

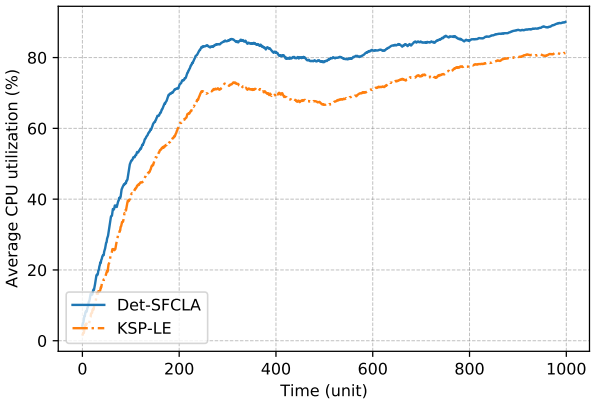


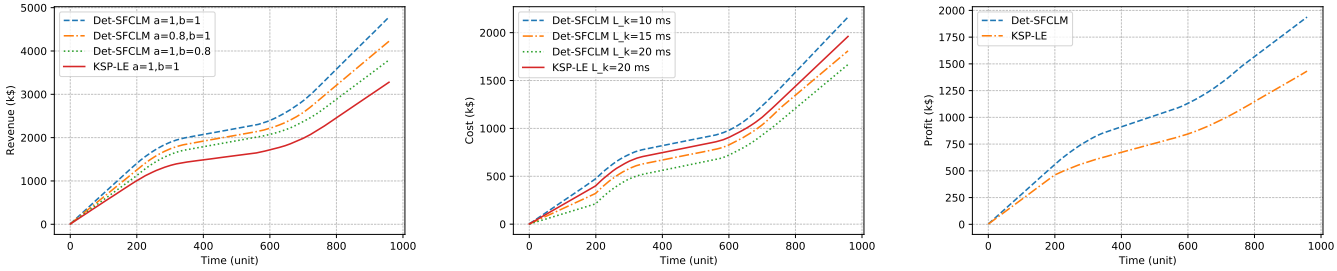
Fig. 9: Average CPU utilization over time.

We evaluate the performance of Det-SFCA in terms of the latency violation over time. As shown in Fig.8(c), the service latency violation of SFCs happens earlier with KSP-LE than with Det-SFCA. The latency violation is low since the available resources are adequate in the beginning of the simulation and the acceptance ratio is high. After $t = 400$, the latency violation decreases and that is due to the reduction of SFC arrivals. Based on this, we also show the average CPU utilization over time in Fig.9. As discussed above, the average CPU utilization increases rapidly along with the deployment of new SFCs, due to the adequate node and link capacities. After $t = 300$, the CPU utilization increases slowly and even decreases. On the one hand, the SFC arrivals at $t = 300$ decreases, and on the other hand, the node capacities are almost occupied by the existing SFCs in the networks. New SFCs will be simply rejected at this time. Compared to KSP-LE, Det-SFCA can result in a higher utilization. Without considering deployment cost, KSP-LE exhibits a lower utilization of CPU resources by rejecting more SFC requests due to resource bottleneck. Indeed, during the adjustment phase, if the resource adjustment is performed without taking into account the history of network load dynamics. For example, if we try to scale up all SFCs when traffic load within an edge node increases, it is more likely that we fail in scale up some SFCs. This may, in turn, result in lower CPU resource utilization. Finally, Det-SFCA keeps about 85% mean CPU utilization rate under heavy network load during the 800-1000 time units.

We evaluate the performance of Det-SFCLM and KSP-LE in terms of the overall profits in Fig.10. Fig.10(a) shows the evaluation of accumulated network revenues derived by accepting SFC requests over time units. Since high data traffic rates directly result in high bandwidth consumption and long latency, affecting the CPU processing resource allocation, we jointly consider the data rate and latency requirement of a SFC (i.e., which can be seen as the indicators of service level agreement – SLA – with ISP) to define the revenue beneath accepting and deploying SFCs into the networks. According to the SFC request arrival rate shown in Fig.6, the traffic load decreases between 200-400 time units and increases between 600-800 time units. The growth rates of accumulated revenue vary in these two time periods. In addition, we add another scaling factor b of bandwidth, and set 80% CPU capacity of physical nodes (a=0.8,b=1) and 80% bandwidth capacity of physical links (a=1,b=0.8), respectively, and that is in order to investigate the impact of network capacity on the revenue. As shown in the figure, when we scale down the network capacity (i.e., bandwidth and CPU capacity), the overall revenue decreases, and that is since less SFC requests are accepted. Furthermore, compared to scaling down the CPU capacity (a=0.8), decreasing the network bandwidth capacity (b=0.8) leads to more network bottleneck when deploying SFC requests. Once a suitable path is selected, the communication channel for the traffic steering of SFC requests is fixed, whereas, there are more candidate localities for embedding VNF instances. If the bandwidth capacity of one of the links along the path is not sufficient, the SFC will lose the opportunity to be embedded. Thus, network bandwidth capacity is the main reason that will cause network congestion for SFC deployment.

In Fig.10(b), we evaluate the resource cost of embedding SFC requests. Since the Det-SFCD algorithm considers the optimal resource cost when allocating latency budgets on the VNF instances of a SFC, it achieves a lower overall resource cost compared with latency equalization scheme. Next, we investigate the impact of latency requirement L_k on the resource cost. We set the latency requirement L_k as 10 ms, 15 ms and 20 ms. The result shows the SFC requests with stricter latency requirements consume more network resources and the case with $L_k = 10$ ms obtains about 20% higher resource cost than the case with $L_k = 15$ ms. Det-SFCLM outperforms KSP-LE in terms of revenue and cost. As a result, it gains higher overall profits. As shown in Fig.10(c), we set the $L_k = 20$ ms and $a = 1, b = 1$, Det-SFCLM obtains about 35% higher accumulated overall profits that KSP-LE.

Finally, we evaluate the performance of Det-SFCLM in terms of the mean latency and jitter and depict the results in Fig.11. We set the latency threshold $\epsilon = 10\%$. We then observe that the mean latency experienced by SFCs is close to the latency requirement. Although the average latencies of Det-SFCLM and KSP-LE are close, the jitter of Det-SFCLM is less the one of KSP-LE, as the latency violation of KSP-LE is much more compared with Det-SFCLM. Furthermore, the jitter increases along with the data rate. On the other hand, the probability of failing to scale up a SFC increases as the data rate increases. Indeed, when the data rate is low, the required



(a) (b) (c)

Fig. 10: (a) The comparison of revenue over time; (b) The comparison of cost over time; (c) Comparison of overall profits over time.

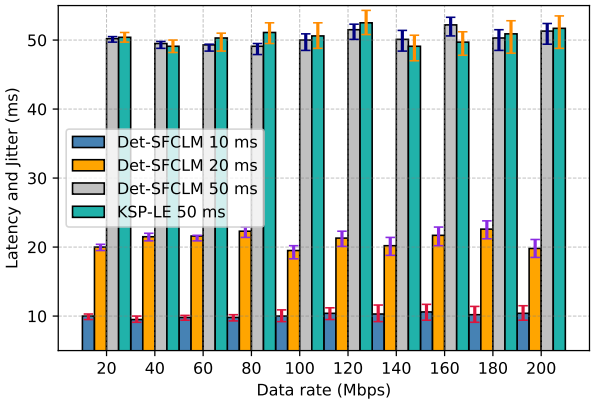


Fig. 11: The mean latency and jitter vs. data rate.

CPU resources are much lower. Thus, the system is more likely to scale up SFCs demanding fewer processing resources during the adjustment phase.

VII. Conclusion

In this paper, we studied the deterministic SFC lifetime management problem in 5G edge fabric and that is with the objective of maximizing the overall profits for ISPs and ensuring the bounded E2E service latency and jitter. For this purpose, the Det-SFCD and Det-SFCA algorithms were proposed. Det-SFCD selects optimal paths and determines processing resource allocation with minimal resource cost for SFC deployment, while keeping the service latency within the latency requirement. Det-SFCA adjusts the processing resource allocation for VNF instances of SFC due the traffic variation in order to ensure a lower jitter (latency variation). These two algorithms jointly solve the deterministic SFC lifetime management efficiently. The conducted performance evaluation showed that the proposed algorithms achieved more than 15% enhancement in SFC acceptance rate and an average 35 % more overall profits in comparison to the baseline solution. In the future, we will study SFC migration and investigate how to ensure the deterministic latency and jitter requirements during the SFC migration. We also plan to design a proactive resource management framework that efficiently carries SFC deployment, migration, and adjustment as per prior and accurate prediction of traffic and network load dynamics.

Acknowledgment

This work was partially supported by the European Union's Horizon 2020 Research and Innovation Program under the CHARITY project with grant agreement No. 101016509 and the MonB5G Project under Grant No. 871780. It was also supported in part by the Academy of Finland 6Genesis project under Grant No. 318927 and IDEA-MILL with grant number 33593.

References

[1] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial internet of things (iiot): An analysis framework," *Computers in industry*, vol. 101, pp. 1–12, 2018.

[2] S. Samii and H. Zinner, "Level 5 by layer 2: Time-sensitive networking for autonomous vehicles," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 62–68, 2018.

[3] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0," *IEEE industrial electronics magazine*, vol. 11, no. 1, pp. 17–27, 2017.

[4] N. Finn, P. Thubert, B. Varga, and J. Farkas, "Deterministic networking architecture," *draft-ietf-detnet-architecture-03 (work in progress)*, 2017.

[5] E. Grossman, C. Gunther, P. Thubert, P. Wetterwald, J. Raymond, J. Korhonen, Y. Kaneko, S. Das, Y. Zha, B. Varga *et al.*, "Deterministic networking use cases," *IETF draft*, 2018.

[6] H. Hantouti, N. Benamar, and T. Taleb, "Service function chaining in 5g & beyond networks: Challenges and open research issues," *IEEE Network*, vol. 34, no. 4, pp. 320–327, 2020.

[7] H. Hantouti, N. Benamar, T. Taleb, and A. Laghrissi, "Traffic steering for service function chaining," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 487–507, 2018.

[8] O. R. Alliance, "O-ran: towards an open and smart ran," *White Paper*, 2018.

[9] H. Yu, T. Taleb, and J. Zhang, "Deterministic service function chaining over beyond 5g edge fabric," *IEEE Globecom 2021*.

[10] H. Yu, F. Musumeci, J. Zhang, M. Tornatore, L. Bai, and Y. Ji, "Dynamic 5g ran slice adjustment and migration based on traffic prediction in wdm metro-aggregation networks," *Journal of Optical Communications and Networking*, vol. 12, no. 12, pp. 403–413, 2020.

[11] L. Gupta, M. Samaka, R. Jain, A. Erbad, D. Bhamare, and C. Metz, "Colap: A predictive framework for service function chain placement in a multi-cloud environment," in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2017, pp. 1–9.

[12] R. Gouareb, V. Friderikos, and A.-H. Aghvami, "Virtual network functions routing and placement for edge cloud latency minimization," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2346–2357, 2018.

[13] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5031–5044, 2019.

[14] L. Qu, C. Assi, K. Shaban, and M. J. Khabbaz, "A reliability-aware network service chain provisioning with delay guarantees in nfv-enabled enterprise datacenter networks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 554–568, 2017.

[15] Q. Zhang, Y. Xiao, F. Liu, J. C. Lui, J. Guo, and T. Wang, "Joint optimization of chain placement and request scheduling for network function virtualization," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 731–741.

[16] I. Benkacem, T. Taleb, M. Bagaa, and H. Flinck, "Optimal vnfs placement in cdn slicing over multi-cloud environment," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 616–627, 2018.

[17] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating virtualized network functions," *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 725–739, 2016.

[18] T. Taleb, M. Bagaa, and A. Ksentini, "User mobility-aware virtual network function placement for virtual 5g network infrastructure," in *2015 IEEE International Conference on Communications (ICC)*. IEEE, 2015, pp. 3879–3884.

[19] T. Lin, Z. Zhou, M. Tornatore, and B. Mukherjee, "Optimal network function virtualization realizing end-to-end requests," in *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.

[20] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*. IEEE, 2015, pp. 171–177.

[21] A. Varasteh, B. Madiwalar, A. Van Bemten, W. Kellerer, and C. Mas-Machuca, "Holu: Power-aware and delay-constrained vnf placement and chaining," *IEEE Transactions on Network and Service Management*, 2021.

[22] J. Pei, P. Hong, K. Xue, and D. Li, "Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 10, pp. 2179–2192, 2018.

[23] J. Liu, W. Lu, F. Zhou, P. Lu, and Z. Zhu, "On dynamic service function chain deployment and readjustment," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 543–553, 2017.

[24] X. Fei, F. Liu, H. Xu, and H. Jin, "Adaptive vnf scaling and flow routing with proactive demand prediction," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 486–494.

[25] I. Leyva-Pupo, C. Cervelló-Pastor, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic scheduling and optimal reconfiguration of upf placement in 5g networks," in *Proceedings of the 23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2020, pp. 103–111.

[26] Y. Liu, X. Shang, and Y. Yang, "Joint sfc deployment and resource management in heterogeneous edge for latency minimization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 8, pp. 2131–2143, 2021.

[27] Q. Xu, J. Wang, and K. Wu, "Learning-based dynamic resource provisioning for network slicing with ensured end-to-end performance bound," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 28–41, 2018.

[28] C. Xu, S. Gamage, P. N. Rao, A. Kangarlou, R. R. Kompella, and D. Xu, "vslicer: Latency-aware virtual machine scheduling via differentiated-frequency cpu slicing," in *Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing*, 2012, pp. 3–14.

[29] S. Khatibi, K. Shah, and M. Roshdi, "Modelling of computational resources for 5g ran," in *2018 European Conference on Networks and Communications (EuCNC)*. IEEE, 2018, pp. 1–5.

[30] G. T. RAN, "Ts 38.214, nr; physical layer procedures for data," *V15.3.0*, Sept. 2018.

[31] J. Janković, Ž. Ilić, A. Oračević, S. A. Kazmi, and R. Hussain, "Effects of differentiated 5g services on computational and radio resource allocation performance," *IEEE Transactions on Network and Service Management*, 2021.

[32] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5031–5044, 2019.

[33] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 17–29, 2008.

[34] B. Y. Chen, X.-W. Chen, H.-P. Chen, and W. H. Lam, "Efficient algorithm for finding k shortest paths based on re-optimization technique," *Transportation Research Part E: Logistics and Transportation Review*, vol. 133, p. 101819, 2020.

[35] Z. Luo and C. Wu, "An online algorithm for vnf service chain scaling in datacenters," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1061–1073, 2020.

[36] H. Yu, F. Musumeci, J. Zhang, M. Tornatore, and Y. Ji, "Isolation-aware 5g ran slice mapping over wdm metro-aggregation networks," *Journal of Lightwave Technology*, vol. 38, no. 6, pp. 1125–1137, 2020.

[37] L. Pantel and L. C. Wolf, "On the impact of delay on real-time multiplayer games," in *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, 2002, pp. 23–29.

Hao Yu received the B.S. and Ph.D degree in communication engineering from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2015 and 2020. He was also a Joint-Supervised Ph.D. Student with the Politecnico di Milano, Milano, Italy. He is currently a Postdoctoral Researcher with the Center of Wireless Communications, Oulu University, Oulu, Finland. His research interests include network automation, SDN/NFV, time sensitive networks, deterministic networking.

Tarik Taleb is currently a Professor at the Center of Wireless Communications, The University of Oulu, Finland. He is the founder and director of the MOSA!C Lab (www.mosaic-lab.org). Between Oct. 2014 and Dec. 2021, he has been a Professor at the School of Electrical Engineering, Aalto University, Finland. Prior to that, he was working as Senior Researcher and 3GPP Standards Expert at NEC Europe Ltd, Heidelberg, Germany. He was then leading the NEC Europe Labs Team working on R&D projects on carrier cloud platforms. Before joining NEC and till Mar. 2009, he worked as assistant professor at the Graduate School of Information Sciences, Tohoku University, Japan, in a lab fully funded by KDDI, the second largest mobile operator in Japan. From Oct. 2005 till Mar. 2006, he worked as research fellow at the Intelligent Cosmos Research Institute, Sendai, Japan. He received his B. E degree in Information Engineering with distinction, M.Sc. and Ph.D. degrees in Information Sciences from Tohoku Univ., in 2001, 2003, and 2005, respectively.

Prof. Taleb's research interests lie in the field of telco cloud, network softwarization & network slicing, AI-based software defined security, immersive communications, mobile multimedia streaming, and next generation mobile networking. Prof. Taleb has been also directly engaged in the development and standardization of the Evolved Packet System as a member of 3GPP's System Architecture working group 2. Prof. Taleb served on the IEEE Communications Society Standardization Program Development Board. As an attempt to bridge the gap between academia and industry, Prof. Taleb founded the "IEEE Workshop on Telecommunications Standards: from Research to Standards", a successful event that got awarded "best workshop award" by IEEE Communication Society (ComSoC). Based on the success of this workshop, Prof. Taleb has also founded and has been the steering committee chair of the IEEE Conf. on Standards for Communications and Networking.

Prof. Taleb served as the general chair of the 2019 edition of the IEEE Wireless Communications and Networking Conference (WCNC'19) held in Marrakech, Morocco. He was the guest editor in chief of the IEEE JSAC Series on Network Softwarization & Enablers. He is/was on the editorial board of the IEEE Transactions on Wireless Communications, IEEE Wireless Communications Magazine, IEEE Journal on Internet of Things, IEEE Transactions on Vehicular Technology, IEEE Communications Surveys & Tutorials, and a number of Wiley journals. Till Dec. 2016, he served as chair of the Wireless Communications Technical Committee, the largest in IEEE ComSoC. He also served as Vice Chair of the Satellite and Space Communications Technical Committee of IEEE ComSoc (2006 - 2010).

Prof. Taleb is the recipient of the 2021 IEEE ComSoc Wireless Communications Technical Committee Recognition Award (Dec. 2021), the 2017 IEEE ComSoc Communications Software Technical Achievement Award (Dec. 2017) for his outstanding contributions to network softwarization. He is also the (co-) recipient of the 2017 IEEE Communications Society Fred W. Ellersick Prize (May 2017), the 2009 IEEE ComSoc Asia-Pacific Best Young Researcher award (Jun. 2009), the 2008 TELECOM System Technology Award from the Telecommunications Advancement Foundation (Mar. 2008), the 2007 Funai Foundation Science Promotion Award (Apr. 2007), the 2006 IEEE Computer Society Japan Chapter Young Author Award (Dec. 2006), the Niwa Yasujirou Memorial Award (Feb. 2005), and the Young Researcher's Encouragement Award from the Japan chapter of the IEEE Vehicular Technology Society (VTS) (Oct. 2003). Some of Prof. Taleb's research work have been also awarded best paper awards at prestigious IEEE-flagged conferences.

Jiawei Zhang received the Ph.D. degree from the State Key Laboratory of Information Photonics and Optical Communications, Beijing University of Posts and Telecommunications (BUPT), China. He currently is an associate Professor with BUPT. Dr. Zhang has authored and co-authored more than 30 OFC/ECOC papers and top journal papers in optical communication and networks. His research interests include the collaboration of optical networks with IP, wireless and cloud/edge, currently with an emphasis on the advanced technologies for providing deterministic connections for future network applications. He served on the Technical Program Committees for the IEEE DRCN 2018-2020, IEEE ICNC 2017-2018, ACP2020, and for the Workshop on Cloud Computing Systems, Networks and Applications at the IEEE GLOBECOM 2014-2016, ICC 2015-2016, and INFOCOM 2017-2018 conferences. He also severed as a Guest Editor of the special issue on Resilience in future 5G Photonic Networks of Photonic Network Communications journal (Springer).