Dual Timescale Orchestration System for Elastic Control of NextG Cloud-Integrated Networks

Quirino Pagliuca*, Luciano Jerez Chaves†, Pasquale Imputato*, Antonia Tulino*‡, Jaime Llorca*‡
*University of Naples Federico II, Italy
†Universidade Federal de Juiz de Fora, Brazil
‡New York University, USA

***Abstract*—The confluence of advanced networking (5G/6G) and distributed cloud technologies (edge/fog computing) are rapidly transforming next-generation networks into highly distributed computation platforms, especially suited to host emerging resource-intensive and latency-sensitive services (e.g., smart transportation/city/factory, real-time computer vision, augmented reality). In this paper, we leverage the recently proposed Cloud Network Flow (CNF) modeling and optimization framework to design a novel two-timescale orchestration system for the joint control of communication and computation resources in cloud-integrated networks. The *Long-Term Controller* solves a properly constructed CNF optimization problem at a longer timescale that determines i) the end-to-end CNF routes (defining data paths and processing locations) for each service chain and ii) the associated allocation of communication and computation resources. The *Short-Term Controller* uses a local control policy to adjust the allocation of communication and computation resources based on queue state observations at a shorter timescale. Driven by the lack of proper simulation tools, we also develop new ns-3 features that allow modeling and simulation of cloud-integrated networks equipped with both communication and computation resources hosting arbitrary service chains. Finally, we integrate the proposed orchestration system into ns-3 to evaluate and analyze the dynamic orchestration of a set of representative service chains over a hierarchical cloud-integrated network.**

***Index Terms*—5G, 6G, cloud networks, edge computing, service orchestration, network modeling, network simulation, ns-3**

I. Introduction

Next-generation (NextG) networks are expected to support a wide range of service classes, each with its own set of stringent quality of service/experience (QoS/QoE) requirements [1]. While 5G/6G technologies are evolving to enable high throughput, low latency, and reliable access networks, distributed cloud networking has emerged as a preeminent paradigm for enabling a flexible cloud-integrated core network. Built upon the foundations of Software-Defined Networking (SDN) and Network Function Virtualization (NFV), it enables the integrated evolution of network and cloud infrastructure into a general-purpose, highly-distributed compute platform capable of hosting next-generation resource-intensive and latency-sensitive services (e.g., industrial automation, smart transportation, mixed reality) in the form of elastic and disaggregated software functions. These functions can be instantiated at multiple locations across the device-edge-cloud continuum, elastically scaled in response to changing conditions, and flexibly interconnected through a programmable network fabric, as illustrated in Fig.1.

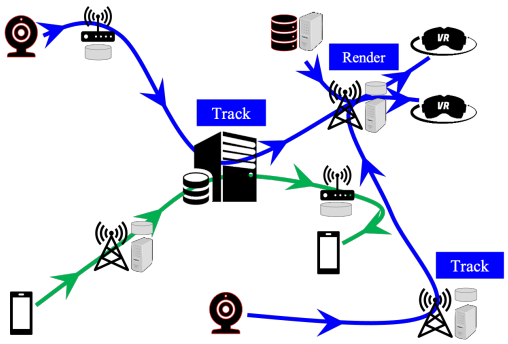


Fig. 1: Cloud-integrated network hosting a mixed reality service with tracking and rendering functions running over the device-edge-cloud continuum.

A. Joint cloud-network orchestration

While distributed cloud networking offers a significant advancement in the industry, the complexity and dynamic nature of managing network and cloud resources within this paradigm necessitates a *unified approach* [2]. The co-design and orchestration of cloud and network resources are fundamental to i) optimizing system performance, ii) enhancing flexibility, and iii) reducing operational costs. Operating in isolation, these resources may be underutilized or overtaxed, leading to inefficiencies and reduced QoS. A unified orchestration system harmonizes the use of cloud and network capabilities to support the communication and computation requirements of NextG services, accommodating dynamic network conditions and traffic demands. Ultimately, using a joint design and orchestration approach can enable optimal resource allocation, leading to increased cost-effectiveness and improved QoS.

To this end, in this paper we leverage Cloud Network Flow (CNF), a unified mathematical framework designed for the modeling and optimization of integrated compute-communication systems such as cloud-integrated networks [3]–[5]. As described in Section III, CNF allows modeling the end-to-end service distribution problem (function placement, flow routing, resource allocation) as a single flow problem over a properly constructed cloud network graph.

B. Dual timescale control

While CNF provides a common framework for the joint orchestration of cloud and network resources, the resulting optimization problems do not come without complexity and scalability challenges. In this paper, our goal is to address the complexity around end-to-end optimization and control of NextG cloud-integrated networks via a dual-timescale orchestration system that leverages the advantages of centralized and distributed control policies, cooperating at different timescales. The rationale behind this approach is that the centralized controller can leverage global system information, compute optimal configurations, and establish consistent policies across the cloud-network continuum albeit slower reactions and scalability limitations. On the other hand, local controllers can provide faster reactions in response to dynamic changes in network conditions and service demands albeit possibly converging to sub-optimal global configurations. A proper combination of centralized optimization algorithms and distributed control policies is expected to strike a desirable balance between optimal and responsive configurations.

C. Cloud-network simulation tools

Accurate performance evaluation is also a challenge. Indeed, cloud network orchestration solutions need to be properly evaluated to identify performance gains, limitations, and directions for future investigations. While numerical evaluation help in assessing the goodness of proposed solutions, it does not take into account the complexity of the whole network stack. On the other hand, experimentation on real networks presents higher complexity and costs. In this context, simulation or emulation tools supporting SDN/NFV realistic modeling and QoS performance evaluation are lacking or have limited support. Indeed, tools like Mininet or ns-3, which allow a full stack network performance evaluation, have only limited support for SDN networks [6].

D. Contributions

In this work, we present a practical proposal for a dual-timescale SDN/NFV orchestration system along with its modeling and evaluation via the ns-3 network simulator. Key contributions can be summarized as:

- the design of a CNF Orchestration System (OS) to jointly optimize end-to-end service flow routing and processing along with the associated allocation of communication and computation resources;
- the design of a dual-timescale control strategy combining long-term optimization and short-term control policies;
- the design of extensions to ns-3 to support both SDN and NFV that allows simulating arbitrary service chains running over a distributed cloud-integrated network;[1]
- the ns-3 implementation of the proposed CNF OS;
- and a preliminary performance evaluation to validate the models and get insights into the CNF OS behavior.

[1]The ns-3 extensions will be incorporated into the OFSwith13 module. We are in the process of releasing the new OFSwitch13 module through the ns-3 app store.

The paper is organized as follows. Section II presents related work. Section III briefly describes the CNF framework. Section IV presents the proposed CNF OS. Section V presents the dual-timescale control strategies. Section VI describes the ns-3 implementation of the CNF OS. Section VII presents simulation results illustrating the benefit of the new CNF OS. Finally, Section VIII states concluding remarks.

II. Related Work

In the context of future softwarized 5G/6G networks, the work in [2] presents a comprehensive discussion on QoE management. It introduces a generalized QoE provisioning ecosystem for services in softwarized 5G/6G and beyond networks. While QoE is supported by network QoS and complex queueing systems [7], the key technologies enabling QoE management are SDN and NFV which play a critical role in modern communication systems [8]–[10]. SDN/NFV networks offer highly flexible and dynamic architectures that can adapt to changing network conditions and requirements. This flexibility enables intelligent strategies to support QoE, as network resources can be efficiently allocated and managed based on user demands and application needs.

The need for advanced management through orchestration systems for SDN/NFV networks is discussed in [11]. The work also discusses the need for proper modeling and evaluation of proposed solutions. Authors in [8] propose a cost-efficient optimized orchestration system that addresses the whole life-cycle management of different Service Function Chains (SFCs). The system considers QoS parameters, such as end-to-end delay, bandwidth, and jitter, along with actual capacities of network functions deployed across multiple cloud/edge locations, taking into account availablecomputation resources. In [12], authors propose a mathematical model for optimizing the embedding of SFCs (implemented in P4), while considering functional and QoS requirements associated with embedding requests. The model accounts for various types of processing devices with different properties, including processing delay and supported features. In [13], authors propose an online orchestration methodology for a multi-user edge service. The orchestrator's goal is to simultaneously maximize QoS and minimize resource consumption. They provide a mathematical formulation to compute an optimal offline policy and derive an online approach using a model-free Deep Reinforcement Learning (DRL) framework.

At the time of writing, there are no readily available models for simulating SDN/NFV-baed cloud-integrated networks or orchestration systems. For example, simulators like Mininet or ns-3 offer limited support for SDN's main functionalities [6]. Our work lies within ongoing efforts in research and development to enhance QoE management in SDN/NFV networks. Specifically, we focus on the modeling and performance evaluation of SDN/NFV cloud-integrated networks and their complex orchestration systems.

III. The underlying Cloud Network Flow Framework

The proposed CNF OS leverages the recently introduced CNF modeling, optimization, and control framework that allows jointly

optimizing function placement, flow routing, and computation-communication resource allocation by solving a single flow problem on a properly constructed cloud network graph [3]–[5]. Key to the CNF framework is the definition of two graphs, the *Service Graph (SG)*, and the *Cloud Network Graph (CNG)*, representing the services' requirements and the cloud network infrastructure capabilities, respectively.

The SG vertices represent service functions and the SG edges represent data streams (or commodities) that are input/output to/from the corresponding service functions. Service functions impose computation rate requirements (e.g., CPU, memory) and service commodities communication rate requirements (e.g., bandwidth). An example of a service graph for an augmented reality service is shown in Fig. 2.

On the other hand, the CNG vertices and edges represent cloud network nodes and links, respectively. Importantly, cloud network links include communication links, representing the capability to send information from one node to another, as well as production, consumption, and computation links, representing the capability to produce, consume, and process information at a given node, respectively. Each cloud network link is characterized by its associated capacity and cost. A generic CNG node is represented in Fig. 3.

Given the proper definition of SG and CNG, CNF allows solving the end-to-end service distribution (function placement, flow routing, and associated allocation of computation and communication resources) as a single flow problem. CNF is in essence a generalization of Multi-Commodity-Flow (MCF) for the optimization of arbitrary services over cloud-integrated networks. Several CNF optimization algorithms have been designed to solve what is referred to as the service distribution problem, whose goal is to find the function placement, flow routing, and associated allocation of cloud and network resources that guarantees QoS requirements while minimizing overall resource cost [3]–[5]. Alternatively, CNF control policies have been designed to dynamically adjust flow routing, scheduling, and resource allocation in response to dynamic changes in network conditions and service demands [14]–[16]. Our goal is to design a dual timescale orchestration system that exploits CNF optimization algorithms at the centralized long-term controller and CNF dynamic control policies at the distributed short-term controllers.

IV. The CNF Orchestration System

In the following section, we present the CNF OS and its main components, as illustrated in Fig. 4. The *Application Layer (AL)* extracts the service demands, in terms of the sequence of service

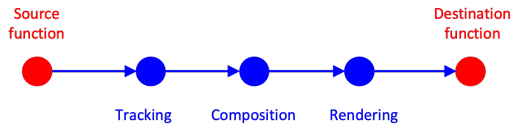


Fig. 2: Example of a service graph (SG) for a mixed reality service with source, processing and destination functions.

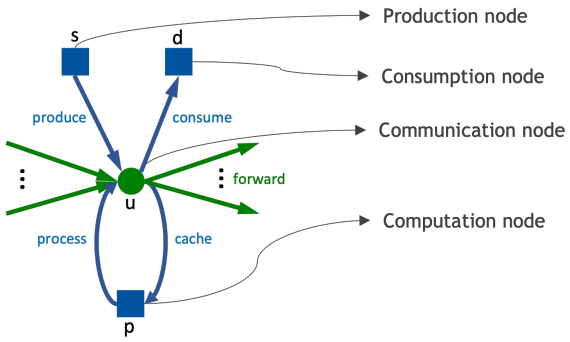


Fig. 3: Generic cloud network graph (CNG) node with production, consumption computation, and communication links representing the potential capabilities of producing, consuming, computing, and sending data over the network.

functions and associated resource requirements for each service, encodes them into an SG, and sends it the the *Control Layer (CL)*. The CL also receives as input the CNG, which is a description provided by the *Infrastructure Layer (IL)* of the cloud network topology and associated resource capacities and costs. The CL is responsible for providing the necessary inputs to both *Long-Term Control (LTC)* and *Short-Term Control (STC)* agents. Operating over a longer timescale, the LTC agent takes as input the SG and the CNG and computes an optimal solution to the underlying CNF optimization problem that includes end-to-end CNF routes (data paths and service processing locations) for each service chain and the associated allocation of communication and computation resources. The LTC output is then used by the IL to configure the end-to-end CNF routes and overall resource allocation on the CN infrastructure. On the other hand, the STC agent interacts with the IL on a shorter timescale. The STC is responsible for adjusting the allocation of communication and computation resources based on near real-time network state information (queue states, available resource capacities, and costs). We highlight that the CNF solution computed by the LTC determines the CNF routes and resource allocation needed to support the required service rates (as given in the SG) with minimum overall resource cost. However, the LTC is constrained to work at a timescale that allows collecting global system information. On the other hand, the STC is designed to work at a shorter timescale, using local but more granular system information such as queue states, to provide fast reactions that are more suitable to address end-to-end latency requirements.

V. Optimization and Control Strategies

In the following, we describe the specific choices of long-term optimization algorithm and distributed control policy for our first implementation of the proposed CNF OS.

A. Long-term optimization

For the long-term controller, we choose to use a mixed-integer linear programming (MILP) solver to directly solve the underlying CNF optimization problem. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote the CNG, with

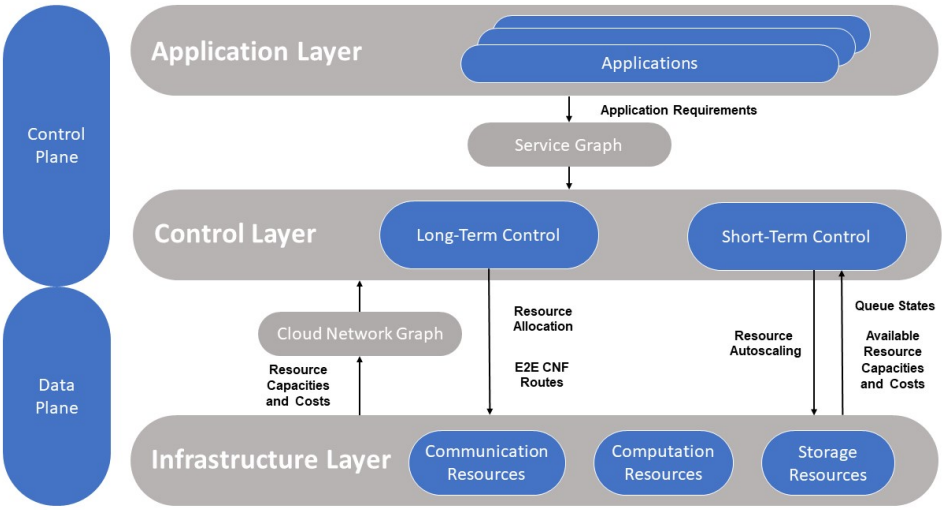


Fig. 4: Architecture diagram for the proposed dual-timescale CNF-based orchestration system.

$\mathcal{V}$ and $\mathcal{E}$ representing the set of nodes and links, respectively. Recall that each link $e = (u, v) \in \mathcal{E}$ can represent production, consumption, communication, or computation resources, each characterized by its capacity c_e (in resource blocks), its resource block capacity c_e^b (in flow units), and its cost w_e (in cost per resource block). Note that depending on the nature of link e (e.g., communication, computation) the flow and resource block capacity units may be of different dimensions [3], [4]. Finally, we use $\mathcal{V}^c \subset \mathcal{V}$ to specify the set of communication nodes within the CNG, and $\delta^-(u) \subset \mathcal{V}$ and $\delta^+(u) \subset \mathcal{V}$ to denote the set of incoming and outgoing neighbors of node u, respectively.

The SG is denoted by $\mathcal{R} = (\mathcal{I}, \mathcal{K})$, with $\mathcal{I}$ and $\mathcal{K}$ denoting the set of functions and commodities (or data streams), respectively. Furthermore, $\mathcal{X}(k)$ denotes the set of input commodities required to produce commodity k, $\mathcal{K}^s$ the set of source commodities, and $\mathcal{K}^d$ the set of destination commodities. Finally, $s(k) \in \mathcal{V}, k \in \mathcal{K}^s$ denotes the CN node hosting source commodity k and $d(k) \in \mathcal{V}, k \in \mathcal{K}^d$ the CN node requesting destination commodity k. Importantly, each commodity $k \in \mathcal{K}$ is characterized by its communication rate requirement (in communication flow units, e.g., Mbps). In addition, each commodity $k \in \mathcal{K} \backslash \mathcal{K}^d$ is also characterized by its computation rate requirement (in computation flow units, e.g., CPU cycles per second). Accordingly, we denote by R_e^k the rate of commodity k when it goes over edge e. This way, R_e^k takes value equal to the communication rate of k when e is a communication link, the computation rate of commodity k when e is a computation link, and zero otherwise. Note also that the rates (communication or computation) of a given commodity in a service chain can be derived by multiplying the rate of the source commodity times appropriate *communication and computation scaling factors*. In Sec. VI, such scaling factors play an important role in properly implementing service functions in ns-3.

Let's now define the following decision variables:

- f_e^k: fraction of commodity k traversing link e
- y_e: number of resource blocks allocated to link e

Then, the CNF optimization problem can be formulated as follows:

$$\min \sum_{e \in \mathcal{E}} y_e w_e \tag{1a}$$

$$\text{s.t.} \sum_{v \in \delta^-(u)} f_{uv}^k = \sum_{v \in \delta^+(u)} f_{uv}^k \qquad \forall u \in \mathcal{V}^c, k \in \mathcal{K} \tag{1b}$$

$$f_{pu}^k = f_{up}^l \qquad \forall u \in \mathcal{V}^c, k \in \mathcal{K}, l \in \mathcal{X}(k) \tag{1c}$$

$$f_{su}^k = 1 \qquad \forall k \in \mathcal{K}^s, s = s(k) \tag{1d}$$

$$f_{ud}^k = 1 \qquad \forall k \in \mathcal{K}^d, d = d(k) \tag{1e}$$

$$\sum_{k \in \mathcal{K}} f_e^k R_e^k \leq y_e c_e^b \leq c_e \qquad \forall e \in \mathcal{E} \tag{1f}$$

$$f_e^k \in [0, 1], y_e \in \mathbb{Z}^+ \qquad \forall e \in \mathcal{E}, k \in \mathcal{K} \tag{1g}$$

In (1), (1a) defines the resource allocation cost function to be minimized, (1b) denotes generalized flow conservation constraints (where incoming and outgoing neighbors can include communication nodes, as well as local production, consumption, and computation nodes), (1c) service chaining constraints (stating the need to have every input commodity in $\mathcal{X}(k)$ going into the local computation node p when producing commodity k at node u), (1d)-(1e) service source and destination locations, and (1f) capacity constraints. Finally, we note that the domain of the flow variables can be $[0, 1]$ or $\{0, 1\}$ depending on the splittable or unsplittable nature of the service flows. We also remark that the combinatorial nature of the resulting MILP is one of the reasons it is used at the LTC, which only runs at periodic long-term intervals. Nonetheless, efficient CNF approximation algorithms can also be

used to compute close-to-optimal solutions to the above MILP in polynomial time in order to match the LTC timescale to the network scale [4], [5].

B. Short-Term Control

The solution to the CNF optimization problem that the LTC computes is designed to minimize the total resource cost while keeping queuing delays bounded via the satisfaction of the capacity constraints 1f (where allocated resources always cover average service rates). However, actual queuing delays will depend on the actual distribution of packet arrivals and associated queue lengths. Hence, in order to control end-to-end service delays, we introduce distributed STC agents designed to adjust the initial resource allocation solution in response to real-time variations of queueing states.

While we plan to investigate advanced control policies, e.g., leveraging data-driven reinforcement learning techniques, in this first implementation of the CNF OS, we choose a simple threshold-based control policy. Local STC agents constantly monitor the TX queue usage on outgoing links. If the queue usage exceeds a predefined threshold value, the STC agent increases the resource allocation of the respective link by one resource block (up to the maximum number of resource blocks available for that link). With this granular heuristic policy, we expect to quickly reduce queuing delays with a controlled impact on total cost. From an implementation point of view, the increase of resource allocation by the STC agents can be achieved through SDN link aggregation. Finally, we highlight that while the STC agents work locally at a shorter time scale, the LTC agent operates at a central controller and evaluates at a longer timescale the goodness of the actions decided by the STC agents.

VI. The CNF OS implementation in ns-3

To model the CNF OS into ns-3, we first describe the CNG and SG graphs through a JSON file. Then, a custom JSON reader provides the input to ns-3 simulation script to properly configure the ns-3 simulation environment. Starting at the IL, we model the CNG vertices and edges as ns-3 nodes and links. Production, computation, and consumption CNG nodes are configured as standard ns-3 host nodes. In turn, communication CNG nodes, which are responsible for packet forwarding, are configured as ns-3 OpenFlow switches, allowing traffic routing at a flow-based granularity.

At the AL, we model the SG vertices (service functions) as custom ns-3 stateless applications running over UDP/IP protocols. These applications can generate, process, and consume network traffic (service commodities). The choice of instantiating applications over the UDP protocol is motivated by the implementation of stateless functions and the possibility of scaling the traffic flow in each processing function to reflect different communication/computation commodity demands. In particular, we model each processing function as a forwarding application that uses the communication and computation scaling factors described in Sec. V to adjust output traffic throughput according to the properties of each service function. The JSON file also brings information on function availability, describing in which CNG nodes we can install instances of the SG functions. As cloud network resources are modeled into CNG links, it is possible to have copies of SG functions on different nodes ready for usage. Besides, to model service function isolation, we use dedicated computation links to connect multiple computation host nodes to the same communication switch. For the CL, we leverage the capabilities of the OFSwitch13 module for ns-3 [6] to implement a custom SDN controller that interacts with the LTC agent and globally configures infrastructure resources. Precisely, the LTC agent described in Sec. V-A is implemented in Python with support of the PuLP library. The interaction between the SDN controller and LTC happens with the support of the ns3-ai module [17], which facilitates the data exchange between ns-3 and Python-based frameworks. When the LTC agent generates a new output, the SDN controller sends OpenFlow messages to the communications switches to (re)configure CNF routes and (re)allocate cloud network resources. On the other hand, the STC is implemented at every communication node, monitoring local outgoing queues and increasing resource allocation locally when necessary. We recall that the LTC agent is able to configure CNF routes and allocate resources through proper configuration of SDN rules from the CL, while distributed STC agents update resource allocation based on real-time local queue observations.

VII. Performance evaluation

A. Simulation setting

We conducted a preliminary performance evaluation of the proposed CNF OS through ns-3 simulations. The goal was to validate the implemented CNF OS along with the optimization strategies and the new features implemented in ns-3. For the cloud network, we considered the CNG represented in Fig. 5, consisting of five communication nodes (in black), eleven computation nodes (in yellow), two production nodes (in red), and two consumption nodes (in blue). Communication nodes include one core cloud node (node 3), two edge cloud nodes (nodes 1 and 2), and two access nodes (nodes 4 and 5). Each computation node represents a cluster of computation resources reserved for the execution of one or more service functions at the given communication node, with the capacity and cost of its incoming link denoting the maximum number of available resource blocks and the cost per resource block, respectively. We set the capacity of all links to 100 resource blocks, except for computation link $(1, 11)$ which has a capacity of 6 resource blocks in order to observe possible congestion-driven rerouting solutions.The capacity per resource block is set to 10 Mbps for all communication links and to 10 Mhz for all computation links. Finally, the cost per resource block is set to 10 for all communication links and 20 for all computation links.

For the set of services, we consider the SG represented in Fig. 6, consisting of five service chains. The i-th service is identified through a source function having ID i-0 and a destination node having ID i-9. Intermediate processing functions have ID i-n, where n denotes the function type. Functions of types 1 and 2 have computation and communication scaling factors of 1.2. Functions of type 3 have computation scaling factor 1.5 and communication

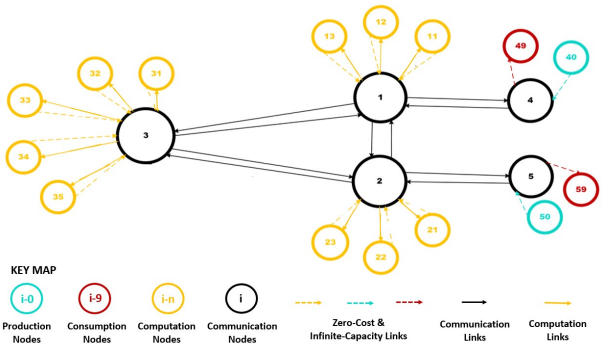


Fig. 5: Representation of the CNG used in the performance evaluation.

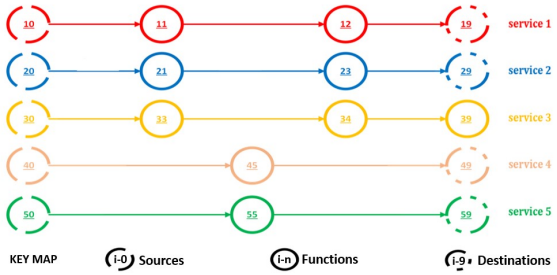


Fig. 6: Representation of the SG used in the performance evaluation.

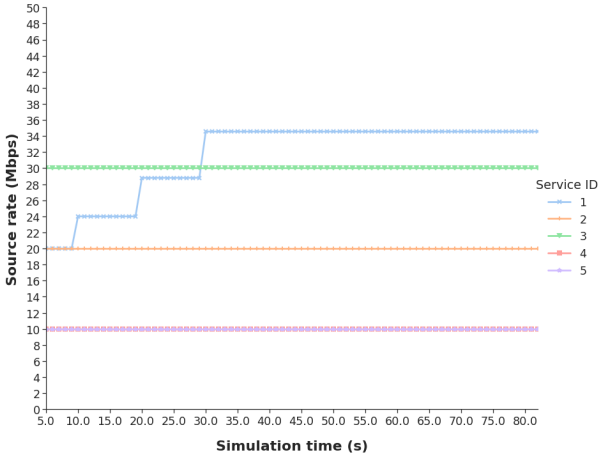


Fig. 7: Service source rates during the simulation time.

scaling factor 0.8. Functions of types 4 and 5 have scaling factors of 0.4 and 1.2 for computation and communication, respectively.

It is important to note that the source and destination functions of each service chain need to be mapped to CNG production and consumption nodes, respectively. In our evaluation, source functions 10, 20 and 40 are mapped to production node 40, source functions 30 and 50 to production node 50, destination functions 19, 39 and 49 to consumption node 49, and destination functions 29 and 59 to consumption node 59. In addition, we assume processing functions with ID i-n for any i can be instantiated at computation nodes j-n for any j.

Each service chain is characterized by its source traffic rate during the simulation period (lasting 80 s) as shown in Fig. 7, with service 1 having non-constant source rate while the other services keep a constant source rate. Services 4 and 5 are representative of background traffic.

We evaluated two scenarios:

- *LTC-only*, where the centralized LTC agent is prompted to evaluate potential resource reallocation and service rerouting at every *LT-interval* of 25 s;
- *Joint LTC-STC*, where, in addition to the LTC agent, distributed STC agents operate at every *ST-slot* of 2 s to dynamically adjust resource allocation to match real-time service demand variations.

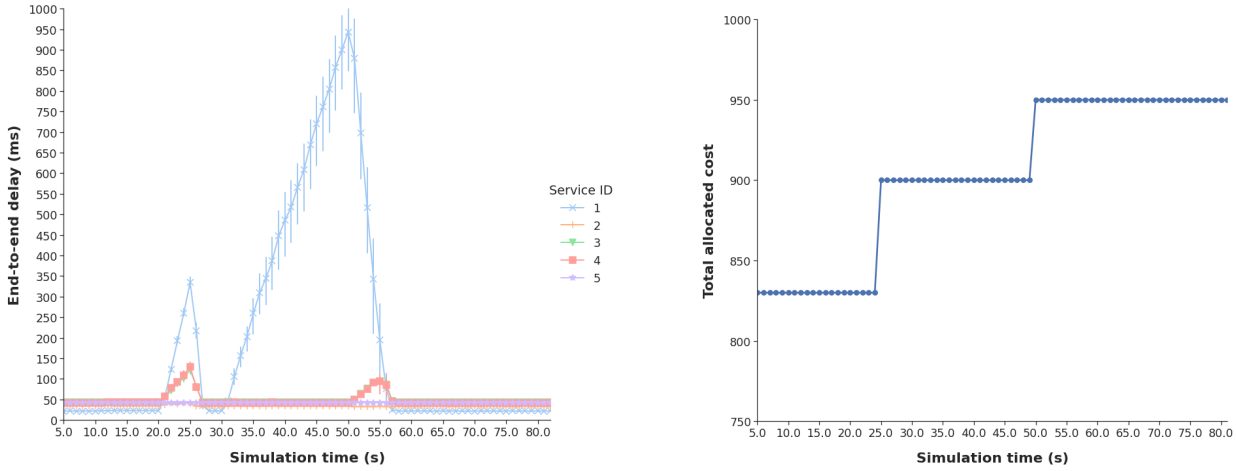
At the beginning of each simulation scenario, the LTC agent solves the CNF optimization problem for the cloud network initial state, providing the initial end-to-end routes for each service chain and the initial total resource allocation. Results are averaged over four simulation runs using different seeds.

B. Simulation results

1) LTC-only scenario: Fig. 8 shows the results for the LTC-only scenario. Fig. 8a shows the measured end-to-end latency for each service throughout the simulation, while Fig. 8b shows the total cost of allocated resources in the same time interval. Recall that the LTC agent determines the optimal solution to the CNF optimization problem at each LT-interval (25 s) in order to accommodate changes in service 1 demand, which increases its source rate three times during the entire simulation (at 10 s, 20 s and at 30 s; see Fig. 7).

At the beginning of the simulation, the LTC agent determined the CNF routes and minimal resource requirement (which amounted to 830 cost units) to support initial service rates. When service 1 increased its source rate by 20% at 10 s, the network successfully managed the new load, owing to the fact that the initially allocated resource blocks are still sufficient to support the heightened load.

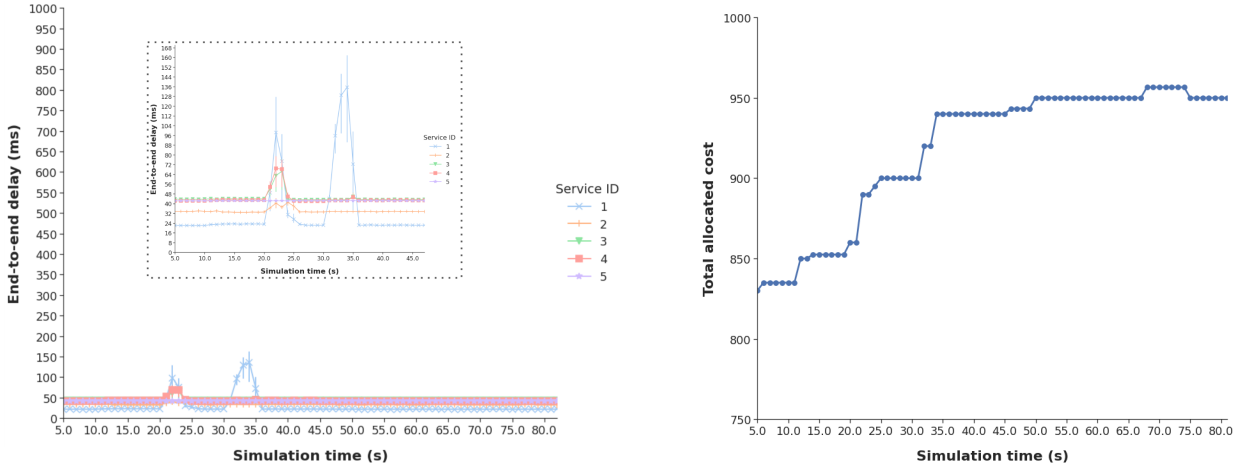
However, at the 20 s mark, service 1 further increased its source rate, resulting in the load of link $(1, 11)$ surpassing its allocated capacity. Consequently, the end-to-end service delays start going up, especially that of service 1, which reaches a delay of 345 ms. The end-to-end delays continued to escalate until the simulation reached 25 s. At this point, the LTC agent is prompted to reassess and implement a new solution. Importantly, this solution involved a reallocation of service 2 (functions 21 and 23) from edge node 1 to edge node 2 to relieve the congestion link $(1, 11)$. In terms of resource allocation, while the total allocation increased to 900 cost units, the allocation at link $(1, 11)$ is adjusted down, rectifying the previous over-allocation which had exceeded the requirements for processing service 1 alone. At 30 s, service 1 further augmented its traffic demand, once again exceeding the allocated resources, which caused another increase in latency. This uptrend persisted until a peak value of 940 ms at the 50 s mark. At 50 s, the LTC



(a) End-to-end service latency (ms).

(b) Total resource allocation cost.

Fig. 8: End-to-end service latencies and total allocated cost during the simulation time for the LTC-only scenario.



(a) End-to-end service latency (ms).

(b) Total resource allocation cost.

Fig. 9: End-to-end service latencies and total allocated cost during the simulation time for the joint LTC-STC scenario.

agent revised the solution, increasing the total allocated resources to a total cost of 950. This adjustment effectively reduced the end-to-end delays to their minimal levels. Finally, when the LTC was engaged at 75 s , the network had already achieved a steady state. Consequently, no notable changes were observed, neither in the latency metrics nor in the resource allocation cost.

2) Joint LTC-STC scenario: Fig. 9 shows the results for the joint LTC-STC scenario. Fig. 9a shows the per-service end-to-end latency during the simulation time, while Fig. 9b shows the total allocated cost during the same time period. We remark that, in this scenario, the centralized LTC agent continues to operate at every LT-interval (25 s) while the resource allocation can be further adjusted at every ST-slot (2 s) by the distributed STC agents. Note how end-to-end latencies appear bounded below 145 ms for the whole simulation period, while the total allocated cost smoothly follows changes in the service traffic demands. We analyzed simulation logs to derive the LTC agents' behavior. More specifically, at the beginning of the simulation, STC agents started slightly increasing the allocation of resource blocks to better accommodate initial service traffic demands. At 12 s, the STC agent at node 1 further increased the allocated resource blocks of link $(1, 4)$ to face the increased service 1 source rate effectively, keeping latencies under control. At time 20 s, the STC agent at node 4 increased the resource allocation of link $(4, 1)$, quickly reacting to the second increase in service 1 source rate. At time 22 s, the STC agent at node 1 effectively increased the allocated resource blocks in links $(1, 12)$ and $(1, 13)$, avoiding waiting until time 25 s, as in the LTC-only scenario, and significantly reducing

the total end-to-end delays. At 25 s, the LTC redetermined the optimal solution (as in the previous experiment). Then, starting from 30 s the STC agent at node 1 increased the allocated resource blocks of links $(1, 11)$ and $(1, 12)$ to keep service 1 delay under control. A new solution from the LTC was prompted at time 50 s confirming the goodness of the actions decided by the STC agents (indeed, it increased the allocated resources by only a few additional resource blocks). Finally, at time 66 s the STC agents decided to slightly increase the allocated resource blocks (in response to slight fast queue length variations), with the LTC agent redetermining the minimum required resource blocks at the time 75 s.

VIII. Conclusions and Future Work

In this work, we proposed a dual timescale orchestration system to efficiently address the end-to-end service distribution problem in NextG cloud-integrated networks. We designed a centralized LTC agent that computes the optimal solution to the underlying CNF optimization problem at each long-term interval, and distributed STC agents that use a local control policy to adjust resource allocation decisions at each short-term timeslot based on real-time queue state information. In order to properly evaluate and simulate the proposed CNF OS, we introduced key new features within ns-3 to support the modeling and simulation of SDN/NFV cloud-integrated networks equipped with flexible orchestration systems. We then implemented the proposed CNF OS along with its LTC and STC systems in ns-3, and conducted a preliminary performance evaluation. Simulation results show that the proposed CNF OS can effectively reroute service traffic and elastically allocate resources in response to changes in service demands. In particular, the centralized LTC agent is able to compute and configure end-to-end routes and overall resource allocation to support service rates with minimal cost, while distributed STC agents effectively complement the LTC agent by providing fast resource allocation adjustments based on real-time queue state information to keep end-to-end service latencies under control. Ongoing and future research directions include more advanced STC strategies , and performance evaluations in large-scale cloud-integrated networks.

Acknowledgments

This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART") and by the US National Science Foundation under grant CNS-2148315.

References

[1] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5G wireless networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1617–1655, 2016.

[2] A. A. Barakabitze and R. Walshe, "SDN and NFV for QoE-driven multimedia services delivery: The road towards 6G and beyond networks," *Computer Networks*, vol. 214, p. 109133, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128622002523

[3] M. Barcelo, A. Correa, J. Llorca, A. M. Tulino, J. L. Vicario, and A. Morell, "IoT-cloud service optimization in next generation smart environments," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 4077–4090, 2016.

[4] H. Feng, J. Llorca, A. M. Tulino, D. Raz, and A. F. Molisch, "Approximation algorithms for the NFV service distribution problem," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[5] K. Poularakis, J. Llorca, A. M. Tulino, and L. Tassiulas, "Approximation algorithms for data-intensive service chain embedding," in *Proceedings of the Twenty-First International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, ser. Mobihoc'20. New York, NY, USA: Association for Computing Machinery, 2020, p. 131–140. [Online]. Available: https://doi.org/10.1145/3397166.3409149

[6] L. J. Chaves, I. C. Garcia, and E. R. M. Madeira, "OFSwitch13: Enhancing ns-3 with OpenFlow 1.3 support," in *Proceedings of the 2016 Workshop on Ns-3*, ser. WNS3'16. New York, NY, USA: Association for Computing Machinery, 2016, p. 33–40. [Online]. Available: https://doi.org/10.1145/2915371.2915381

[7] P. Imputato and S. Avallone, "Traffic differentiation and multiqueue networking in ns-3," in *Proceedings of the 2017 Workshop on Ns-3*, ser. WNS3 '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 79–86. [Online]. Available: https://doi.org/10.1145/3067665.3067677

[8] M. Bagaa, T. Taleb, J. Bernabe, and A. Skarmeta, "QoS and resource-aware security orchestration and life cycle management," *IEEE Transactions on Mobile Computing*, vol. 21, no. 08, pp. 2978–2993, aug 2022.

[9] M. Gramaglia, I. Digon, V. Friderikos, D. von Hugo, C. Mannweiler, M. A. Puente, K. Samdanis, and B. Sayadi, "Flexible connectivity and QoE/QoS management for 5G networks: The 5G NORMA view," in *2016 IEEE International Conference on Communications Workshops (ICC)*, 2016, pp. 373–379.

[10] P. Krishnan, K. Jain, P. G. Jose, K. Achuthan, and R. Buyya, "SDN enabled QoE and security framework for multimedia applications in 5G networks," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 17, no. 2, apr 2021. [Online]. Available: https://doi.org/10.1145/3377390

[11] P. Demestichas, A. Georgakopoulos, K. Tsagkaris, and S. Kotrotsos, "Intelligent 5G networks: Managing 5G wirelessmobile broadband," *IEEE Vehicular Technology Magazine*, vol. 10, no. 3, pp. 41–50, 2015.

[12] H. Harkous, B. A. Hosn, M. He, M. Jarschel, R. Pries, and W. Kellerer, "Performance-aware orchestration of P4-based heterogeneous cloud environments," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2023.

[13] C. Quadri, A. Ceselli, and G. P. Rossi, "Multi-user edge service orchestration based on deep reinforcement learning," *Computer Communications*, vol. 203, pp. 30–47, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366423000737

[14] H. Feng, J. Llorca, A. M. Tulino, and A. F. Molisch, "Optimal dynamic cloud network control," *IEEE/ACM Transactions on Networking*, vol. 26, no. 5, pp. 2118–2131, 2018.

[15] J. Zhang, A. Sinha, J. Llorca, A. M. Tulino, and E. Modiano, "Optimal control of distributed computing networks with mixed-cast traffic flows," *IEEE/ACM Transactions on Networking*, vol. 29, no. 4, pp. 1760–1773, 2021.

[16] Y. Cai, J. Llorca, A. M. Tulino, and A. F. Molisch, "Ultra-reliable distributed cloud network control with end-to-end latency constraints," *IEEE/ACM Transactions on Networking*, vol. 30, no. 6, pp. 2505–2520, 2022.

[17] H. Yin, P. Liu, K. Liu, L. Cao, L. Zhang, Y. Gao, and X. Hei, "Ns3-ai: Fostering artificial intelligence algorithms for networking research," in *Proceedings of the 2020 Workshop on Ns-3*, ser. WNS3 2020. New York, NY, USA: Association

for Computing Machinery, 2020, p. 57–64. [Online]. Available: https://doi.org/10.1145/3389400.3389404