This is a postprint version of the following published document:

Zeydan, E., Manges-Bafalluy, J. & Turk, Y. (2021). Intelligent Service Orchestration in Edge Cloud Networks. *IEEE Network*, 35(6), 126–132.

DOI: [10.1109/mnet.101.2100214](https://doi.org/10.1109/mnet.101.2100214)

© 2021, IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Intelligent Service Orchestration in Edge Cloud Networks

Engin Zeydan*, Josep Mangués-Bafalluy*, Yekta Turk†

*Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), Barcelona, Spain, 08860.

Email: {engin.zeydan,josep.mangués}@cttc.cat

†Aselsan Corp., Istanbul, Turkey, 34906. Email: yektaturk@aselsan.com.tr

Abstract—The surge in data traffic is challenging for the network infrastructure owners coping with stringent service requirements (e.g. high bandwidth, ultra-low latency) as well as shrinking per GB revenues. Network softwarization and edge computing are powerful candidates to mitigate these issues. In parallel, there is an increasing demand for network virtualization and container-based services. In this study, we investigate the management of Software Defined Networking (SDN)-based transport network and edge cloud service orchestration. To this end, we use a machine learning (ML)-based design to manage both transport and edge cloud resources of a mobile network effectively. To generate and use real-world data inside our ML platform, we use Graphical Network Simulator-3 (GNS3) emulator environment. Our emulation results indicate that almost all of the trained ML models can accurately select the correct Edge Clouds (ECs) (i.e. with high test accuracy) under the considered two scenarios when transport and EC network parameters are considered in comparison to models trained via only transport or cloud based parameters. At the end of the paper, we also provide an evolved architecture where the proposed ML platform can be embedded in an end-to-end mobile network architecture and H2020 5Growth project's baseline management platform.

Keywords—orchestration, mobile operator, transport, cloud, emulation, machine learning.

I. INTRODUCTION

A distributed mobile network environment with integrated edge and cloud computing capability should support many different services and support each service with its own set of dependencies. As network usage increases or decreases with end-user demands, it is expected that the capacity of the services that are deployed to the different edges of the mobile network react these changes rapidly. In this case, the provided capacity of the Edge Clouds (ECs) that are distributed in the different parts of the Mobile Network Operator (MNO) infrastructure should be re-evaluated. The distributed structure of the EC environments eases to achieve better network services for users while providing many alternative edge points in case of failures. However, the costs can be enormous over time if the allocated resources for the services are not managed properly. Moreover, the orchestration task will be more complicated to provide high availability in a scalable manner when the Life Cycle Management (LCM) of the EC services are

orchestrated and Key Performance Indicators (KPIs) from the transport networks are managed separately.

The orchestration and the LCM operations for network services that are running in Virtual Network Functions (VNFs) are performed by Network Function Virtualization Orchestrator (NFVO) in the European Telecommunications Standards Institute (ETSI) Management and Orchestration (MANO) architecture. By enabling the LCM, any network service can be modified without a direct request coming from operations support system or business support system. Therefore, the NFVO can react to events or the changes in the status of the network service and has ability to decide to change some parameters related with this network service. However, the current trend in mobile networks is to utilize containers for mobile service applications with service orchestration tools, such as Kubernetes. In these architectures that run cloud services in containers, LCM operations are executed by the software extensions, such as Kubernetes Operators, Auto-scaler, etc. [1].

A. The Requirement for Transport & Cloud Awareness

The main drawback of current LCM entities in container-based domains is their lack of monitoring the status of the KPIs in other segments of the mobile network that can affect their decisions. This limited perspective may not be a problem when managing network services in a centralized data-center environment since the physical servers are installed adjacently and all resources are assigned to the mobile network from a single point. However, the situation becomes challenging in the MNO infrastructure where the Core Network (CN) can be distributed using mobile ECs. Problems in the transport network, which connects the distributed ECs, affect the accessibility and need to be considered during orchestration decisions. In situations where a transport network problem arises, it is necessary to change the serving EC. EC selection from the perspective of only transport network parameters or based on only EC resource availability parameters may create extra cost and performance degradation. For these reasons, the KPIs collected by the orchestrator for network services should be analyzed together with transport network KPIs to maintain stability on the overall of the network that covers both the cloud and transport domains. However, in networks with a large number of transport nodes and with multiple ECs, it is

Fig. 1: Illustration of the cloud-only, transport only and both transport & cloud awareness during backbone link problems.

impossible to apply rigid rule-based policies, thus Machine Learning (ML) guidance need to be exploited.

B. Problem Statement over a Case Study

Fig. 1 illustrates a high level schematic diagram of the considered problem in which there are three regions, namely Region-1, Region-2 and Region-3. Each region is being served by several Base Stations (BSs) and a corresponding EC node for each region. We assume that the same service (having a single VNF) can be deployed in different regions via corresponding ECs. Requests arrive from the affected Transport Module (TM) of BS-1 with certain capacity and link level requirements. We assume that there is a faulty backbone link marked with red color and capacity limitations in EC-2 marked with exclamation mark in Fig. 1. The main aim is to accommodate those requests of the TM of BS-1 under both cloud and transport network constraints. To solve this problem, different strategies can yield different EC connection options, hence different paths to heal the affected traffic of the requests. In our problem, we use the following three different strategies for comparison purposes:

- 1) **Cloud-aware approach:** In this strategy, only cloud KPIs are used and Kubernetes cannot decide which core to select since it only selects the core based on cloud KPIs (central processing unit (CPU), memory, storage), unaware of the underlying transport network KPIs.
- 2) **Transport-aware approach:** In this case, the transport network is deciding which core to select, but unaware of the cloud KPIs. In case core is busy, the forwarding of traffic will congest the CN and service disruptions can occur. Note that this considers the problem as only a routing optimization issue.
- 3) **Cloud & Transport-aware approach:** This strategy combines transport network and Cloud KPIs to decide

EC selection in case the already selected EC is busy or not available due to various factors.

For example, by only observing cloud level parameters in the cloud-aware strategy, links marked with orange colors in path number 1, by only checking transport parameters in transport-aware strategy links marked with purple color in path number 2 and by observing both transport and cloud parameters in cloud & transport-aware strategy links marked with green color in path number 3 can be selected to route the traffic from affected TM of BS-1 to EC-1, EC-2 and EC-3, respectively. Note that by just observing and making decisions using only cloud or transport-aware based parameters, the requests of the affected TM of BS-1 cannot be met. For example, cloud-aware based decision does not give the best link, which may violate the link level requirements, the transport-aware based decision can give better transport link paths, but EC-2's capacity limitations may not accommodate the capacity level requests. For this reason, the selection of the most suitable EC should depend on both transport and cloud parameters observations.

II. SERVICE ORCHESTRATION

A. Management & Orchestration of Cloud and Transport Networks

Network-aware resource provisioning is a well researched field, several theoretical results and practical achievements (e.g., OpenStack or Kubernetes as a Virtual Infrastructure Manager (VIM)) have already been published. Many European Horizon 2020 projects has worked on network orchestration and management using Artificial Intelligence (AI)/ML techniques [2]. Standardization organizations such as ETSI, 3rd Generation Partnership Project (3GPP) also aim to form AI-based network orchestration [3]. In particular, ETSI has been proposing Experiential Networked Intelligence (ENI) reference architecture. On the other hand, 3GPP

presented network data analytics function [4] is a promising solution that has the aim to bring intelligence with data analytics to mobile networks. However, these studies are on the conceptual level.

The traditional orchestration ways, such as exact optimization methods, heuristics, meta-heuristics and game-based methods, etc. may need to be designed to a specific problem, require high computational complexities and lack scalability features for large scale network optimization problems. Therefore, combining network orchestration with emerging ML concepts has been the focus of many research papers in recent years using reinforcement learning strategies in [5], slice admission policy control in [6], root cause analysis in [7] and network-aware scheduling policies to consider network parameters at deployment time in [8]. In this paper, we continually monitor network parameters during the whole service operation lifetime.

B. Data-driven Service Orchestration Approaches

Service orchestration can be normally divided into two components including VNFs embedding and links embedding. Link mapping can be solved based on Multi-Commodity Flow algorithm (MaVenM) and the shortest-path algorithm (MaVen-S) whereas for VNF mapping, an Markov Decision Proces (MDP) based approach is also currently popular [9]. In the specific considered problem of this paper, the environment modelling and ML-based algorithm designing are two important processes. The data-driven and ML-based solutions for the VNF embedding approaches can be adopted to deal with the service orchestration problem [10]. However, edge cloud network scenario can be complex and definitely more dynamic compared to the traditional core cloud datacenter networks. For this reason, some basic ML-based approaches cannot adapted for real time embedding. However, high accuracy of traditional ML models for VNFs performance prediction and placement at network cloud and edge facilities with the most prominent accuracy performances being exhibited by models such as K-Nearest Neighbors Regression, Decision Tree, and Support Vector Regression have also been demonstrated [11].

In this paper, a new ML-based data-driven EC selection method for mobile operators is presented. The main goal is to decide (possibly rank) all ECs that are available (i.e., with adequate resources to accommodate requests) even during transport network failures. The method is assessed within an emulated test environment (flexible for large-scale scenarios) consisting of Software-Defined Networking (SDN) controllers and Open Virtual Switch (OVS) gathering transport network traces and Kubernetes clusters gathering cloud parameters. The main benefit comes from an improved data connection/restoration capability when failures in SDN-based transport networks occur. Finally, we provide a new design of proposed AI/ML platform embedded in an end-to-end mobile network architecture based on extension of the 5Growth¹ project's baseline platform.

Fig. 2: The interaction of the ML platform with Orchestrator and SDN controller.

III. INTEGRATION OF SDN-BASED TRANSPORT INTO SERVICE ORCHESTRATION

Fig. 2 shows the block diagram of the proposed methodology. There are three main entities in this block diagram: (a) Service Orchestrator (SO) for measuring the status of cloud parameters, (b) SDN controller for collecting network related data, and (c) ML entity for executing the decision making process. The SDN controller and the orchestrator interact with the ML Platform via RESTful APIs. The output decision of the ML entity is an ranked list of candidate ECs.

A. Orchestration of Edge Cloud Clusters & The Role of SDN

A cluster can be described as a source for all containerized mobile service applications and includes one or more ECs. The service orchestration capability is distributed to all ECs in the cluster with the pre-defined master and slave ability. A master EC that is positioned on top of a cluster is responsible for guiding the cluster and the other ECs in the cluster are the slave ECs that process assigned tasks. In cases where a mobile service application is deployed on a cluster, the master schedules the deployment and distributes the work to the slaves. The slave ECs run the mobile service applications inside the containers and execute start, stop, scale up/down instructions according to the requests coming from the cluster master. Moreover, connections via Northbound Interface (NBI) of SDN controller make SDN-based transport simpler and ideal for interaction with ML platform positioned in an upper hierarchical layer.

B. Machine Learning Guidance for Orchestration

ML-based analysis has been recognized as an efficient way to provide sophisticated decision making when the

¹www.5growth.eu/

considered number of factors and the required sources of information are at large-scale. Fig. 2 shows a high level interaction of the ML platform that can collect data from the service orchestrator (cloud related) and the SDN controller (transport related), and performs intelligent decisions. The decision from ML platform is forwarded to the SO for scaling of the selected EC and SDN controller for path selection to the selected EC through Representational State Transfer (REST) Application Programming Interface (API) using Hypertext Transfer Protocol-Secure (HTTPS).

In large-scale deployments, deciding the best EC connection is key for increasing mobile network reliability. However, most current approaches on data collection take simple network indicators due to the difficulty of modeling and collecting realistic data from both network planning and cloud monitoring tools. In the absence of such large-scale modeling approaches, emulators such as Graphical Network Simulator-3 (GNS3) emulation environment embedded with SDN controllers and Kubernetes on the cloud side would allow data collection from realistic network measurements. This can build precise and enhanced ML-based models to make enhanced decisions and actions.

IV. EXPERIMENTAL EVALUATION

A. Test Setup

For comparisons and validations, we use three different strategies as given Section I-B. In an ideal condition, ML models should be validated over real networks. However, since this is extremely difficult in large-scale deployment scenarios, we resort to GNS3 shown in Fig. 3 to demonstrate the results of our ML-assisted approach. For this purpose, we built a Kubernetes cluster that includes one master node (labelled as M1) and two slave nodes (labelled as EC1 and EC2), a request node (labelled as TM of BS-1). Then, we integrated our Kubernetes cluster within the GNS3 as depicted in Fig. 3. The same network service is installed in all containers (in each of the nodes of the cluster) that are located separately and a single-domain orchestration is assumed. The GNS3 is used to provide a SDN environment consisting of four OVSs. L1 link faults between OVS-1 and OVS-3 are created inside GNS3 according to the possible and most usually faced faults, as described in [12].

B. Feature Selection & Dataset Generation

In Cloud-aware approach, there are 3 ECs and 2 features per EC namely *free_CPU* and *free_memory* values. In Transport-aware approach, there are 4 links and 12 features per link namely “Receive_Bw_bytes”, “Receive_Bw_packet_number”, “Transmit_Bw_bytes”, “Transmit_Bw_packet_number”, “collision_count”, “crc_errors”, “frame_errors”, “receive_errors”, “receive_overrun”, “transmit_drops”, “receive_drops”, “transmit_errors”. To obtain meaningful features in the transport-aware decisions, two routing paths for each EC are used and the corresponding features are obtained by summing all the values of each 12 features. For example, for the M1

Fig. 3: Experimental setup created by using the GNS3 emulation environment for validation purposes.

EC of Fig. 3, two routing paths are available to reach BS1: the first one is using link L1 and the second one is using path L2-L3-L4. In this case, the number of newly formed features becomes 72 (2 links per EC $\times$ 12 features per link $\times$ 3 ECs).

In Cloud & Transport-aware approach, the total number of features as input to ML algorithms becomes 78 (72 (from transport) + 6 (from cloud)) including both cloud and transport features. In addition to the above features, the requests of the nodes (CPU, memory, link level parameters) are also added as additional features into the ML algorithm for training, validation as well as testing. The cloud-aware and transport-aware approaches are used as benchmark to check the benefit of using the combined dataset for appropriate EC selection.

To create a *training dataset*, we first generate traffic from TM of BS-1 into each of the EC nodes M1, EC1 and EC2 and label them with the corresponding EC names. For example, traffic between TM of BS-1 and M1 in Fig. 3 is labelled as M1 EC. The same training dataset generations and corresponding labelling are also done with EC1 and EC2. After training datasets are obtained, different ML models are compared via splitting the training dataset into training and validation datasets (with 80%-20% split). The generated dataset that is used in our analysis (as described in Section IV-A) is uploaded to IEEE Dataport’s web portal in [13].

C. Test Scenarios

The following two test scenarios are created to obtain a *test dataset* (i.e. features and their values from the generated traffic samples):

- **In scenario I**, L1 link in Fig. 3 is disrupted with no change in ECs's capacities. In this controlled scenario considering transport and cloud level parameters, the requests of the TM of BS-1 can be best fulfilled by EC1 over L4 link, which is selected as the ground-truth for algorithm comparisons.
- **In scenario II**, L1 link in Fig. 3 is disrupted as well as the memory capacity of EC1 is reduced substantially, hence M1 and EC1 cannot fulfill the requests from the TM of BS-1. In this controlled scenario, the requests by TM of BS-1 can only be fulfilled by EC2 (over L4-L3 links) which is selected as the ground-truth for algorithm comparisons.

Note that depending on the requirements of the requests of the TM of BS-1 in Fig. 3, the corresponding links and hence the selection of ECs by the ML platform may also vary. During our training process, we assumed that there is enough capacity (i.e. CPU and memory availability) and link availability in all considered ECs for all requests of the TM of BS-1.

D. AI/ML Platform and Algorithm Selection

The ML platform in Fig. 2 collects the transport network links' KPIs from the OpenDayLight Controller and container related KPIs from Kubernetes cluster where the network service is located. The requests to each container are established from the TM of BS-1 of Fig. 3 with various CPU, memory and link requirements.

The ML platform performs classifications on each row sample of the test dataset (created from scenario I and scenario II) using various pre-trained ML models. To select the most eligible ML algorithms in our data engineering pipeline and obtain the inference time in a realistic distributed computing environment, we used algorithms for multi-class classification implemented in Spark MLlib library² installed via Cloudera Manager over two node cluster. The utilized ML algorithms are selected to be Logistic Regression (LR), Random Forest Classifier (RFC), Naive Bayes Classifier (NBC) and Decision Tree Classifier (DTC) which demonstrated almost the same high accuracy model performances (above 98%) after hyper-parameter optimizations on the training dataset.

E. Evaluations of Strategies

The accurate EC selections for scenario I and scenario II are highlighted with green colors in Table I. Note that the inference time as well as the corresponding 95% Confidence Interval (CI) values of Table I for each model are extracted using both monitoring and AI/ML platform described in Section VI and 1000 independent spark jobs for each model's inference times. The highest percentage of test dataset (generated in both scenario I and II) that are labeled by ML model as either M1, EC1 or EC2 are shown in parenthesis in Table I. First of all, the analysis results indicate that no ML algorithms can find the appropriate

EC in cloud-aware only and transport-aware only strategies in both scenarios. On the other hand, in cloud & transport-aware strategy, RFC and DTC algorithms were able to select EC1 and EC2 correctly in both scenario I and scenario II, respectively. In this strategy, LR and NBC algorithms were not able to yield accurate EC mappings.

More specifically, **in cloud-aware strategy**, 100% of test dataset samples are labelled correctly as EC1 in scenario-I for all the algorithms, whereas incorrect labelling of M1 or EC1 is done in scenario-II for all algorithms. **In transport-aware strategy**, none of the algorithms can identify the correct ECs in both scenarios. **In cloud & transport-aware strategy**, DTC and RFC algorithms can label test dataset with the correct ECs based on majority ruling over classification predictions (i.e. 56% and 94% of test dataset are labelled with EC1 in scenario-I and 100% of test dataset are labelled with EC2 in scenario-II for DTC and RFC algorithms respectively). Note that more data during the training phase as well as frequent model updates can help to better generalize the selection of ECs in the system. These results clearly identify the importance of utilizing both cloud and transport level features for the most appropriate EC selection in both disruption scenarios (failures in the transport link connections) as well as unavailable resource scenarios (e.g. via capacity limitations of the ECs). On the other hand, although RFC has given very powerful results in comparison to other algorithms, it can also take a little higher inference time, i.e. 90.8 msec on average. At the same time, utilizing cloud & transport-aware strategy yields higher inference times in comparison to cloud-aware and transport-aware strategies whose inference times are below 70 and 90 msec respectively due to lower number of features. Note that, in Kubernetes clusters, the best practice for autoscaling takes around 30 seconds [1]. Therefore, even if the transport network makes decisions lower than this value, the overall joint decision process (Cloud & Transport) will depend on this value. This shows that the inference time calculated by the proposed platform has a limited impact in delay.

As a consequence of the above analysis, the performance gain obtained by *in cloud & transport-aware strategy* is expected to be much higher for users that are using low volume or bursty traffic generating services (such as voice over IP (VoIP), Machine Type Communication (MTC), etc.) than for users using data-intensive services (such as video on demand streaming, etc.) when failures in transport network emerge. This is due to the fact that low volume generating services will be migrated more smoothly to new ECs than data-intensive services, which also depend on users' behaviour and their locations.

V. CHALLENGES & POTENTIAL SOLUTIONS

(i) *Deployment*: 5G services provided by ECs and clustering options may vary. From the perspective of cloud and SDN, there are two possible paths for ML integration in SO:

- *Evolutionary (brownfield) deployment*: This approach is used when a huge number of installed base of

²<https://spark.apache.org/mllib/>, Accessed July-2021

TABLE I

CLASSIFICATION ACCURACY AND MODEL INFERENCE TIMES FOR SCENARIO-I AND SCENARIO-II IN (A) CLOUD-AWARE, (B) TRANSPORT-AWARE AND (C) CLOUD & TRANSPORT-AWARE STRATEGIES.

Scenario I (True Label EC1)	Strategies/Algorithms	LR	RFC	NBC	DTC
	Cloud-aware	EC1 (100%)	EC1 (100%)	EC1 (100%)	EC1 (100%)
	Transport-aware	EC2 (96%)	EC1 (83%)	EC2 (96%)	EC1 (83%)
	Cloud & Transport-aware	EC2 (71%)	EC1 (94%)	EC2 (75%)	EC1 (56%)
Scenario II (True Label EC2)	Cloud-aware	EC1 (100%)	M1(100%)	M1(100%)	M1(100%)
	Transport-aware	EC2 (95%)	EC1 (100%)	EC2 (76%)	EC1 (100%)
	Cloud & Transport-aware	M1 (100%)	EC2 (100%)	EC2 (60%)	EC2 (100%)
Inference Time (ms) (mean & CI (95%))					
	Cloud-aware	65.2 & [63.9-66.6]	65.5 & [63.9-67.1]	61.8 & [60.5-63.1]	59.9 & [58.8-60.9]
	Transport-aware	86.2 & [84.9-87.5]	88.1 & [86.8-89.3]	78.8 & [77.5-80.1]	86.5 & [84.9-88.1]
	Cloud & Transport-aware	86.6 & [85.2-88.1]	90.8 & [89.2-92.3]	82.7 & [81.1-84.4]	89.9 & [88.7-91.2]

mobile networks exist and new deployments should evolve on top of the existing infrastructure. Brown-field deployments are envisaged to evolve rather than completely replacing the used architecture.

- *Clean-slate approach (greenfield) deployment*: This approach has more degrees of freedom and is not subject to constraints of the already deployed existing systems. On the other hand, it can be more costly and substantial issues, such as security, scalability, and performance, make this approach more challenging.

(ii) *Reliability*: The most suitable solution to train ML in a real mobile operator environment will be to create potential faulty cases during a maintenance window period (e.g. early morning hours). However, if there is an undetected problem of flow messages of OpenFlow or in cluster control messages flowing to the ML platform during the training phase, then the training phase may be problematic. In this case, ML analysis will decide the wrong EC selection. Moreover, the interface connecting the ML platform to the SO and to the SDN controller should be robust against interface errors, load imbalance and inter-controller/inter-cluster (in case of more than one controller/cluster) latency surges. Finally, the interface design and standardization efforts are crucial for inter-operability. This item should be evaluated and developed under the concept of zero touch network service management of ETSI [14].

(iii) *Scalability & ML Algorithms*: EC network scenarios can be complex and dynamic, compared to the traditional core cloud data center networks. In different network topologies (e.g. when ring or star connections in a transport network vary) or cases when ECs are distributed, same ML algorithms cannot be re-utilized. Additionally, some basic ML-based approaches can be time consuming and cannot be adapted for real time embedding. To deal with the online and dynamic service orchestration in these large and complex network scenarios, the ML algorithms should be updated depending on the topology changes. The relation between the selected ML algorithm and the scalability issues may emerge in two dimensions:

- 1) EC scalability: The dispersion rate of ECs within the

transport network may make evident the limitations of the utilized ML algorithm. The use of more than one cluster and classifying the transport network based on these clusters may overcome this limitation.

- 2) Computational scalability: The analysis covering all the transport nodes controlled may have scalability issues due to (de)installed nodes. The ML computation, data processing and decision logic may become over-complex due to link and node failures in the network. For instance, the total number of features in transport network with K number of features per link and can easily expand to $(L \times K \times M)$ where L is the number of links per EC and M is the number of ECs.

A possible solution to this challenge can be utilizing federated learning based approaches. In highly dynamic network environments with frequent local topological updates, federated learning agents can generate local models, later to be aggregated in a centralized global model.

(iv) *Service-focused Decisions*: The 5G CN is centered around service-based architecture with network slicing to offer flexible development and management services. Moreover, micro service-based architecture allows the operator to run customized network slices by selecting the required applications. This flexibility and the characteristics of slices constitute a heterogeneous network environment. In this case, EC selection may change depending on the provided service and its requirements. It is an issue that can be solved by observing ML behaviour for a long time after the deployment and optimizing it on service basis.

VI. PROPOSED END-TO-END DESIGN

For cloud segment management, Kubernetes has been used as the de-facto standard in industry acting as the engine for orchestration of containers. Although Kubernetes offers a myriad of options and is a mature container orchestration engine, the capabilities are limited by only observing the cloud-related parameters and not easily extended into network transport domain observations. Therefore, there is no standard way of incorporating network data with cloud related data sources, such as CPU and memory.

Fig. 4: Proposed AI/ML platform integrated with an end-to-end mobile network management & orchestration architecture.

To circumvent the above limitations, we discuss a new reference architecture that is suitable for intelligent automated network management and orchestration named as 5Growth architecture. Fig. 4 shows the proposed high-level architecture where an AI/ML platform is embedded in an end-to-end mobile network architecture and 5Growth baseline management platform. The 5Growth baseline architecture consists of three layers, as illustrated in Fig. 4. Vertical Slicer serves as the vertical portal to requests, manages vertical services, and creates custom network slice(s) tailored to the service needs. Service Orchestrator is in charge of end-to-end service and resource orchestration of the Network Function Virtualization (NFV)-NS (Network Service) and management of their service life-cycles and deciding the optimum placement of the NFVs and allocating virtual networking, computing and storage resources across a single or multiple domains, based on service requirements and availability of the resources offered by the local domain and each of the peer administrative domains. Finally, Resource Layer is responsible for managing the physical/virtual in-

frastructure resources to deploy the network services and the required transport connections to interconnect them, as requested by the SO.

Fig. 4 also shows the closed loop workflow for automatic network control and intelligent decision making. In a first step, the monitoring platform (e.g. via Apache Kafka) collects data from the 5Growth infrastructure and transfers it to AI/ML platform. In the second step, the collected data is analyzed via the algorithms present in the AI/ML platform (e.g. via Apache Spark). In the third step, the automatic actions and network control are taken over the 5Growth infrastructure with the aid of AI/ML platform algorithm's decisions. Note that a more detailed version of the 5Growth architecture components and their interactions with the AI/ML platform can be found in [15].

VII. CONCLUSIONS AND REMARKS FOR FUTURE

In this paper, we focused on management of SDN-based transport network and EC service orchestration. We

introduced our proposal of network and cloud resource-aware system which can make decisions relying on ML models. We discussed its system design in detail, explained the potential realization of an integrated mobile EC and backhaul architecture and demonstrated it within a realistic emulated environment integrating cloud orchestrator and SDN controller. The effects of topological changes on the performance of the applied ML algorithm and analyzing the scaling properties of the learning and decision processes when we add extra features to be considered during the operation are other research directions.

VIII. ACKNOWLEDGMENT

This work has been partially funded by the EU H2020 5Growth Project (grant no. 856709), by MINECO grant TEC2017-88373-R (5G-REFINE) and Generalitat de Catalunya grant 2017 SGR 1195.

REFERENCES

- [1] Kubernetes, "Autoscaling components for Kubernetes," <https://bit.ly/2W9OkI8>, [Online; accessed July-2021].
- [2] 5G-PPP White Paper, "Cloud Native and 5G Verticals' services," <https://5g-ppp.eu/wp-content/uploads/2020/02/5G-PPP-SN-WG-5G-and-Cloud-Native.pdf>, 2020, [Online; accessed March-2021].
- [3] Y. Wang *et al.*, "Network management and orchestration using artificial intelligence: Overview of ETSI ENI," *IEEE Communications Standards Magazine*, vol. 2, no. 4, pp. 58–65, 2018.
- [4] 3GPP Technical Specification, "5G System; Network Data Analytics Services Stage 3 (Release 15)," *Group Core Network and Terminals*, 2020.
- [5] C. Natalino *et al.*, "Machine learning aided orchestration in multi-tenant networks," in *2018 IEEE Photonics Society Summer Topical Meeting Series (SUM)*. IEEE, 2018, pp. 125–126.
- [6] M. R. Raza, A. Rostami, L. Wosinska, and P. Monti, "A slice admission policy based on big data analytics for multi-tenant 5g networks," *Journal of Lightwave Technology*, vol. 37, no. 7, pp. 1690–1697, 2019.
- [7] Y. Turk, E. Zeydan, and Z. Bilgin, "A machine learning based management system for network services," in *2019 Int. Conf. on Wireless and Mobile Computing, Networking and Communications*. IEEE, 2019, pp. 1–8.
- [8] J. Santos *et al.*, "Towards network-aware resource provisioning in kubernetes for fog computing applications," in *2019 IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2019, pp. 351–359.
- [9] S. Haeri and L. Trajković, "Virtual network embedding via monte carlo tree search," *IEEE transactions on cybernetics*, vol. 48, no. 2, pp. 510–521, 2017.
- [10] A. Dalgkitis, P.-V. Mekikis, A. Antonopoulos, and C. Verikoukis, "Data driven service orchestration for vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, 2020.
- [11] M. Bunyakitanon *et al.*, "Auto-3p: An autonomous vnf performance prediction & placement framework based on machine learning," *Computer Networks*, vol. 181, p. 107433, 2020.
- [12] Y. Yu *et al.*, "Fault management in software-defined networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 349–392, 2019.
- [13] E. Zeydan, J. Mangues and Y. Turk, "Intelligent Service Orchestration in Mobile Cloud Edge Networks: KPIs from Network Elements, IEEE Dataport," [dx.doi.org/10.21227/m76b-hx19](https://doi.org/10.21227/m76b-hx19).
- [14] ETSI GS ZSM 007 V1.1.1, "Zero-touch network and Service Management (ZSM); Reference Architecture," <https://www.etsi.org/technologies/zero-touch-network-service-management>, 2019, [Online; accessed March-2021].
- [15] X. t. Li, "5growth: An end-to-end service platform for automated deployment and management of vertical services over 5g networks," *IEEE Communications Magazine*, vol. 59, no. 3, pp. 84–90, 2021.

Engin Zeydan received his Ph.D. degree in Electrical Engineering from Stevens Institute of Technology, Hoboken, NJ, USA, in 2011. He is currently a Senior Researcher at Centre Tecnològic de Telecomunicacions de Catalunya (CTTC). His research interests are in the areas of telecommunications and big data.

Josep Mangues-Bafalluy received the PhD degree in Telecommunications in 2003 from the Technical University of Catalonia (UPC). He is Senior Researcher and Head of the Communication Networks Division of the Centre Tecnològic de Telecomunicacions de Catalunya (CTTC).

Yekta Turk received his Ph.D. degree in Computer Engineering from Maltepe University, Istanbul, Turkey, in 2018. He is with Aselsan Corp. His research interests are in the areas of mobile radio telecommunications and computer networks.