On SDN-driven Network Optimization and QoS aware Routing using Multiple Paths

Miloud Bagaa[1], Diego Leonel Cadette Dutra[2], Tarik Taleb[1,3,4] and Konstantinos Samdanis[5]
[1] Dep. of Communications and Networking School of Electrical Engineering, Aalto University, Espoo, Finland
[2] Federal University of Rio de Janeiro, Rio de Janeiro, Brazil
[3] Centre for Wireless Communications (CWC), University of Oulu
[4] Department of Computer and Information Security, Sejong University, South Korea
[5] Nokia Bell Labs, Munich, Germany
Emails:{firstname.lastname}@aalto.fi; diegodutra@lcp.coppe.ufrj.br; konstantinos.samdanis@nokia-bell-labs.com

***Abstract*—Software Defined Networking (SDN) is a driving technology for enabling the 5th Generation of mobile communication (5G) systems offering enhanced network management features and softwarization. This paper concentrates on reducing the operating expenditure (OPEX) costs while i) increasing the quality of service (QoS) by leveraging the benefits of queuing and multi-path forwarding in OpenFlow, ii) allowing an operator with an SDN-enabled network to efficiently allocate the network resources considering mobility, and iii) reducing or even eliminating the need for over-provisioning. For achieving these objectives, a QoS aware network configuration and multipath forwarding approach is introduced that efficiently manages the operation of SDN enabled open virtual switches (OVSs). This paper proposes and evaluates three solutions that exploit the strength of QoS aware routing using multiple paths. While the two first solutions provide optimal and approximate optimal configurations, respectively, using linear integer programming optimization, the third one is a heuristic that uses Dijkstra short-path algorithm. The obtained results demonstrate the performance of the proposed solutions in terms of OPEX and execution time.**

***Index Terms*—SDN, Cloud networks, QoS, Optimization, Multi-path routing**

I. Introduction

5G is expected to be a revolution to mobile communications enabling new services with stringent requirements, while opening the network to multiple tenants driving new business opportunities. The 5G system mainly consists of three-parts, which are: i) The Radio Access Network (RAN) that introduces new radio technologies, such as millimeter-wave and massive multiple-input and multiple-output (MIMO), to increase the bandwidth and reduce the delay; ii) the core network that relies on softwarization and a cloud-native architecture [2]–[6], which decouples the control from the user plane, introducing new network functions; iii) SDN-enabled transport network that interconnects the RAN and the core network, facilitating value added services and applications hosted in different servers referred to as Data Networks (DNs) [7].

An abridged version of this paper has been published in the proceedings of the 2017 edition of the IEEE GLOBECOM [1].

To ensure the stringent requirements of 5G services, the core network ought to provide an end-to-end path optimization service that targets the applications' desired requirements, e.g., delay and bandwidth, allowing a flexible allocation of resources. To meet these Key Performance Indicators (KPIs), mobile network operators typically over-provision network resources to assure the desired QoS considering the peak demands. This approach simplifies the network design as the mobile network operator statically maps the resources on the underlying transport network links and nodes. However, over-provisioning becomes costly with the evolution of a plethora of new applications and services with rigorous performance demands. It also proves to be inefficient since the mobile network is not flexible to acquire or modify the allocated transport network resources, while path assignment takes no consideration of the user mobility.

In the 5G era, coordination among the mobile and transport networks is also required to facilitate network slicing and enable multi-tenancy by allocating end-to-end resources on-demand. To effectively address this, a new Application Programming Interface (API) is introduced in 3GPP [8] and IETF [9] as shown in Fig.1. This type of mobile-transport API relates the 5G network orchestration and management system, which handles 3rd party requests and the life-cycle management (LCM) of the mobile network, with the underlying transport network SDN controller that configures paths with the desired performance capabilities to connect the RAN and core network. The mobile-transport API allows: i) transport network resource capabilities exposure towards the 5G orchestration and management system adopting resource abstraction models [10], [11], and ii) carries out the mapping of mobile network service requirements towards the transport network resources and paths, including life-cycle management.

SDN [12], [13] is introduced as a 5G enabler in the transport network layer allowing programmability and efficient traffic steering [14]–[16]. SDN leverages white box switches for ensuring network connectivity by considering different media including both wired and wireless communications, i.e., via an Access Point in case of wireless medium or directly plugged to SDN-enabled switches in case of a wired network. This paper

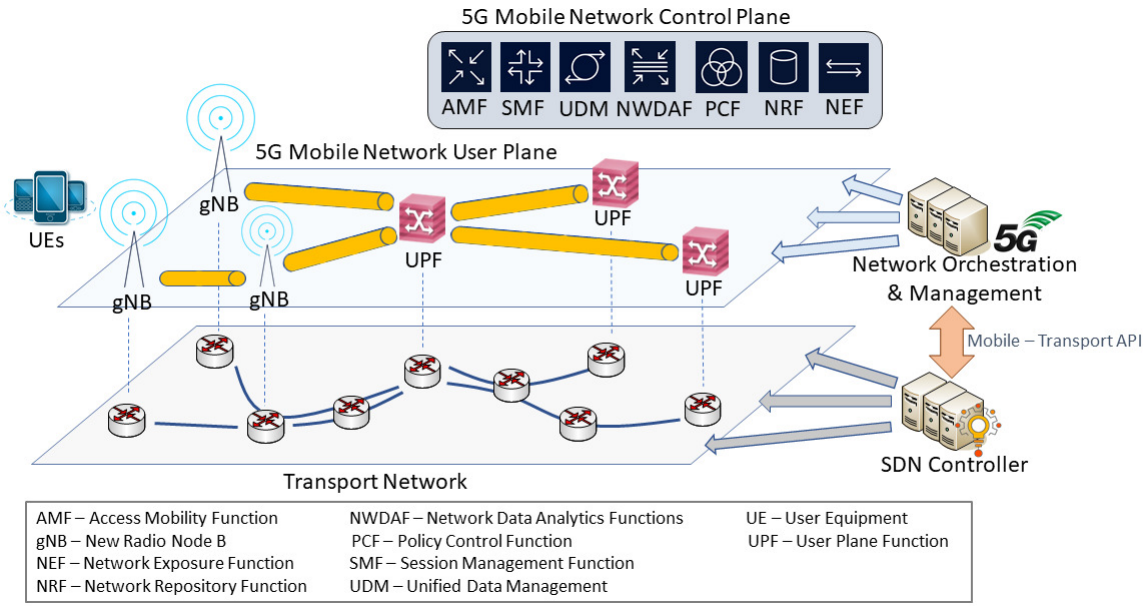


Figure 1: Mobile and transport network system architecture.

explores the QoS queuing feature in OpenFlow that allows a mobile operator with an SDN-enabled network to efficiently allocate resources taking advantage of the available multiple paths [17] of the underlying transport network infrastructure. This approach enables fine-tuning in resource allocation, which improves network utilization, while also assuring QoS provisioning. Nevertheless, its efficiency relies on a frequent collection of performance statistics and re-computation of the allocated resources, ideally considering the patterns of arriving and handover users.

Our interest concentrates on how to compute and update the set of transport network paths considering an SDN controller (e.g., ONOS [18], [19]) responsible for managing the data plane by pushing different network configurations. The data plane may adopt the queue control available on version 1.3 of the OpenFlow protocol to facilitate multiple paths towards an end-host or end-point, e.g., base station, using multiple paths to forward different flows via distinct routes simultaneously. A mobile user that streams, for instance, a video on demand via the proposed SDN paradigm, requires the mobile operator to allocate and maintain the desired bandwidth once it admits the streaming request and initiates the streaming upon receiving such a confirmation. This highlights the importance of admission control in establishing QoS, which needs to exploit user context and mobility, alongside the Service Level Agreements (SLAs).

Besides users' context and SLAs, assuring network utilization efficiency can significantly influence the maintenance of the desired QoS. However, the time needed to determine a new network configuration that reflects such utilization efficiency plays a significant role in practice. If a new network configuration needs a long amount of time to be determined, i.e., far beyond the time it takes a user to connect or change the attached evolved NodeB (eNodeB), then it may no longer have a practical value for optimizing the network utilization. This paper proposes and evaluates three SDN-based resource allocation techniques leveraging the benefits of multipath forwarding in network utilization efficiency. The two first solutions configure path and connectivity resource allocations in the entire network upon the reception of a new user service request or upon a user movement. While one of these solutions, named Full Paths Re-computation (FPR), provides an optimal configuration by exploring linear integer programming, the second solution, named Heuristic Paths Re-computation (HPR), provides efficient network configuration by exploiting Dijkstra shortest-path Algorithm. The third solution, referred to as Partial Paths Re-computation (PPR), provides an approximate optimization, re-computing and configuring only the network resources related with newly attached or handover users, by maintaining a resource availability topology pruning links and/or the associated link capacity that is already allocated to static, i.e., non-moving, users.

The contributions of this paper include: i) an SDN-based network resource utilization strategy exploiting multipath forwarding in a mobile network environment, ii) the formulation and analysis of three algorithms for multipath forwarding considering user mobility context and QoS demands, iii) an evaluation study based on simulation, demonstrating the efficiency of each proposed solution in terms of network utilization and OPEX reduction costs, while assuring a reasonable computational time. Our focus concentrates on the algorithms' insights assuming that the related scalability issues can be handled considering: i) domain-specific deployments with a moderate amount of routers and switches, and ii) algorithm execution once a significant change occurs in the allocated network resources.

The remaining of this paper is organized as follows. Section II presents the related work. Section III describes the network model, while Section IV-A describes the two first solutions that provide full paths re-computation. Section V

elaborates on the solution that only partially re-computes the forwarding graph. Section VI analyzes the performance evaluation. Finally, Section VII concludes the paper.

II. Related work

The use of SDN [20] introduces the capability for resource programmability allowing QoS provisioning across heterogeneous equipment and networks. Sonkoly et al. [21] have described a preliminary set of QoS capabilities for the European OpenFlow testbed Ofelia. Their major contribution focuses on QoS provision, introducing performance measurements and resource management mechanisms, which facilitate queue configuration and flow mapping to particular queues.

Egilmez et al. [22] present a solution that uses OpenFlow to guarantee end-to-end QoS for video streaming, by performing traffic classification and route allocation with bandwidth guarantees. The proposed solution demonstrates significant improvements for video performance, but it is not scalable since it assumes that a single SDN controller manages the entire network. To resolve scalability issues, Egilmez and Tekalp extended their initial work in [23] considering multiple distributed controllers. Each controller is responsible for allocating QoS routes within an Autonomous System (AS) or domain, exploiting an aggregate network view to perform inter-AS QoS routing.

Similarly, Sharma et al. [24] proposed an SDN QoS framework relying on the Floodlight controller of the Ofelia testbed to prioritize selected traffic over best-effort. The proposed framework concentrates on inter-domain aspects considering a single controller per AS, which communicates via a northbound API with a bandwidth broker responsible for providing the respective policies for assuring SLAs with end customers or neighboring brokers. Authors, in [25], [26], have addressed the dynamic control assignment problem. Whilst [25] aims to minimize the average response time of the control plane based on the stable matching problem with transfer, [26] applies the randomized fixed horizon control framework translating the problem into a series of stable matching ones with transfers.

The aforementioned solutions perform programmable traffic prioritization and load balancing to minimize congestion considering stationary clients. Tomovic et al. [27] have introduced the notion of fairness in SDN-based QoS provisioning, which aims to minimize the degradation of best-effort traffic while guaranteeing the desired QoS for priority flows. This framework is based on resource reservation, optimum path selection, and admission control using the POX controller. So far, none of the SDN-based QoS solutions considered leverages the benefits of multiple paths. A simple technique, orthogonal to SDN that distributes incoming flows uniformly across a set of pre-determined paths is known as Equal Cost Multipath (ECMP) [28]. Although ECMP is scalable, it is not considering the load of each path and hence it can introduce traffic imbalance and potential congestion. In the context of SDN, Celenlioglu and Mantar [29] proposed a routing and resource management model leveraging the benefits of load balancing utilizing multiple pre-established paths with resource reservation for intra-domain environments. Such a scheme improves routing scalability and decreases the admission time assuring QoS guarantees for stationary nodes.

Jinyao et al. [30] proposed HiQoS, a sophisticated SDN-based multipath solution that uses OpenFlow queuing mechanisms to implement multipath forwarding and bandwidth guarantees. The proposal relies on a modified version of Dijkstra's algorithm that considers QoS constraints to compute such multipaths, which are stored in a hash-map. HiQoS allocates QoS paths using this map in combination with a real-time network state, allowing rapid recovery from failures. The proposed path selection uses a price and distance criteria, and the authors' main goal is to distribute traffic over the network through the allocation of paths with the minimum load. Sahhaf et al. [31] have proposed a similar approach considering an adaptive multipath provisioning scheme that selects paths with maximum bandwidth and availability. Hussain et al. [32] have evaluated an SDN based multipath solution for data center networks using Floodlight. Flows are scheduled using a hash function over a set of pre-computed paths, prioritizing the least congested paths with the capability of reactively altering the forwarding rules for flows with longer times.

Cross-layer coordination among ISPs utilizing multi-paths to achieve optimal resource allocation and increased reliability is considered by Basit et al. [33]. Huang et al. experimentally evaluated an SDN multipath solution for GridFTP [34], [35] to address traffic engineering considering a multipath modification of Dijkstra's algorithm to increase data transfer rates. A study on SDN enabled disjoint multipath routing is performed by Fu and Wu [36] demonstrating the benefits of load balancing against the conventional shortest path routing, considering different network graph models. Guillen et al. [37] proposed a hybrid, i.e. server and path, load balancing that allows higher throughput by exploiting SDN-based multipath capabilities for a distributed storage system. An overlay multipath framework focusing on matching underlay paths to reduce QoS degradation due to uncertainty is explored in Guan et al. [38] considering the service type. Our approach adopts a similar multi-path strategy but focuses on minimizing the equipment usage as long as the QoS constraints are satisfied instead of distributing the load equally, considering an environment with frequent user mobility.

Dwarakanathan et al. [39] have introduced a high availability QoS-aware module to ensure the desired bandwidth with respect to service types. The proposed technique allows an aggregated resource allocation on the corresponding switches ensuring scalability while providing regular network "health" checks. Yoon and Kamal [40] have proposed a mixed integer linear programming model and a local optimization heuristic based on simulated annealing for minimizing the energy consumption in SDN networks while guarantying the desired QoS. Our approach would also rely on a regular resource check policy to assure that the allocated network resources are optimal with respect to user mobility patterns, but with the additional objective to minimize OPEX using as fewer switches as possible. Tariq and Bassiouni [41] have proposed an SDN enabled QoS-aware Multipath-TCP (MPTCP) solution based on Dijkstra's algorithm that selects P paths between

two end nodes, while Wang et al. [42] explore multipath forwarding for scheduling MPCTP flows in a virtualized environment. However, MPTCP splits a flow into sub-flows that are forwarded into multiple paths, which may prove to be impractical for none-TCP flows and complex to handle for mobile users.

III. Network Model and Problem Formulation

Let $G(V, E, \mathcal{W})$ denote a weighted graph, where V represents a set of nodes and E the set of edges in the network. $V = \mathcal{C} \cup \mathcal{O} \cup \mathcal{S}$, where $\mathcal{C}$, $\mathcal{O}$, and S denote the set of clients, the set of Open Virtual Switches (OVSs), and the set of servers in the network, respectively. Each edge is associated with a weight $\mathcal{W}$, where $\mathcal{W}_{u,v}$ of an edge $(u, v) \in E$ denotes the bandwidth capacity between nodes u and v.

Table I: List of Notations

Notation	Description
$\mathcal{C}$	The set of clients in the network.
$\mathcal{O}$	The set of open virtual switches in the network.
$\ddot{\mathcal{O}}$	The set of none-activated open virtual switches in the network.
$\mathcal{S}$	The set of servers in the network.
$G(V, E, \mathcal{W})$	The graph that shows the network topology, where $V = \mathcal{C} \cup \mathcal{O} \cup \mathcal{S}$, E denotes the set of edges, $\mathcal{W}$ denotes the bandwidth communication between different nodes in V and $\omega_{i,j} \in \mathcal{W}$ denotes the bandwidth capacity between $i, j \in V$.
$\ddot{G}(V, E, \mathcal{W})$	A copy of G that removes the stationary clients and the resources used by them from the original graph G.
$X_{i,j}$	A decision boolean variable that shows if a node i selects j as parent. $\mathcal{X}_{u,v} = \begin{cases} 1 & \text{If } u \text{ selects } v \text{ as parent} \\ 0 & \text{Otherwise} \end{cases}$
$\mathcal{C}_s$	Denote the set of the clients of the server $s \in \mathcal{S}$.
$\lvert\mathcal{C}_s\rvert$	Denote the cardinality of $\mathcal{C}_s$.
$\mathcal{S}_c$	Denote the set of the servers of the client $c \in \mathcal{C}$.
$X_{i,j}^{c,s}$	A decision boolean variable that shows if a node i selects j as parent to forward the traffic from c to s. $\mathcal{X}_{u,v}^{c,s} = \begin{cases} 1 & \text{If } u \text{ would forward the traffic of client } c \\ & \text{to the server } s \text{ through the OVS } v \\ 0 & \text{Otherwise} \end{cases}$
Y_o	A decision boolean variable that shows if a switch $o \in \mathcal{O}$ is selected to forward the traffic or not. $\mathcal{Y}_o = \begin{cases} 1 & \text{If } o \text{ is selected to forward user's traffic} \\ 0 & \text{Otherwise} \end{cases}$
$\mathcal{T}_{i,j}$	A real number variable representing the amount of traffic that would be forwarded from i to j.
$T_{i,j}^{c,s}$	A real number variable representing the amount of traffic that client c sent to server s that would be forwarded from i to j.
$\eta(u)$	A function that returns the neighbors of node u in graph G.
$\mathcal{F}_{i,j}^{s}$	An integer variable that mimics packet flow generated from different clients towards server s.
$F_{i,j}^{c,s}$	An integer variable that mimics packet flow generated from client c towards the server s.

This paper considers two types of communication: i) the wireless communication between each client $c \in \mathcal{C}$ and the access node (eNodeBs) and ii) the wired-line communication between different SDN-enabled switches (OVSs). For the communication between a client and a specified server, we have the following types of communication. The first hop between the client and the access point is a wireless interface, while the remaining network links towards the server are wired. Let $\eta(u)$ represent the set of neighbors of a node $u \in V$. Assuming that each client requires a specified service at a time from a single server with a certain upper bound bandwidth, we are interested in accommodating efficiently the maximum amount of different client requests. For seeking to remove the ambiguity and without loss of generality, if a client is interested in $\mathcal{N}$ services, we simply replicate that client by $\mathcal{N}$, with each one representing a specified service. We denote by $\mathcal{S}_s$ for $s \in \mathcal{S}$, the set of clients that are interested in a service offered by a particular server, formally represented as $\bigcup_{s \in \mathcal{S}} \mathcal{S}_s = \mathcal{C}$. Each client $c \in \mathcal{C}$ requires a specified upper bound bandwidth $\mathcal{B}_c$ related with a different service. Table I summarizes the notations used in this paper.

IV. Efficient management of the full paths re-computation

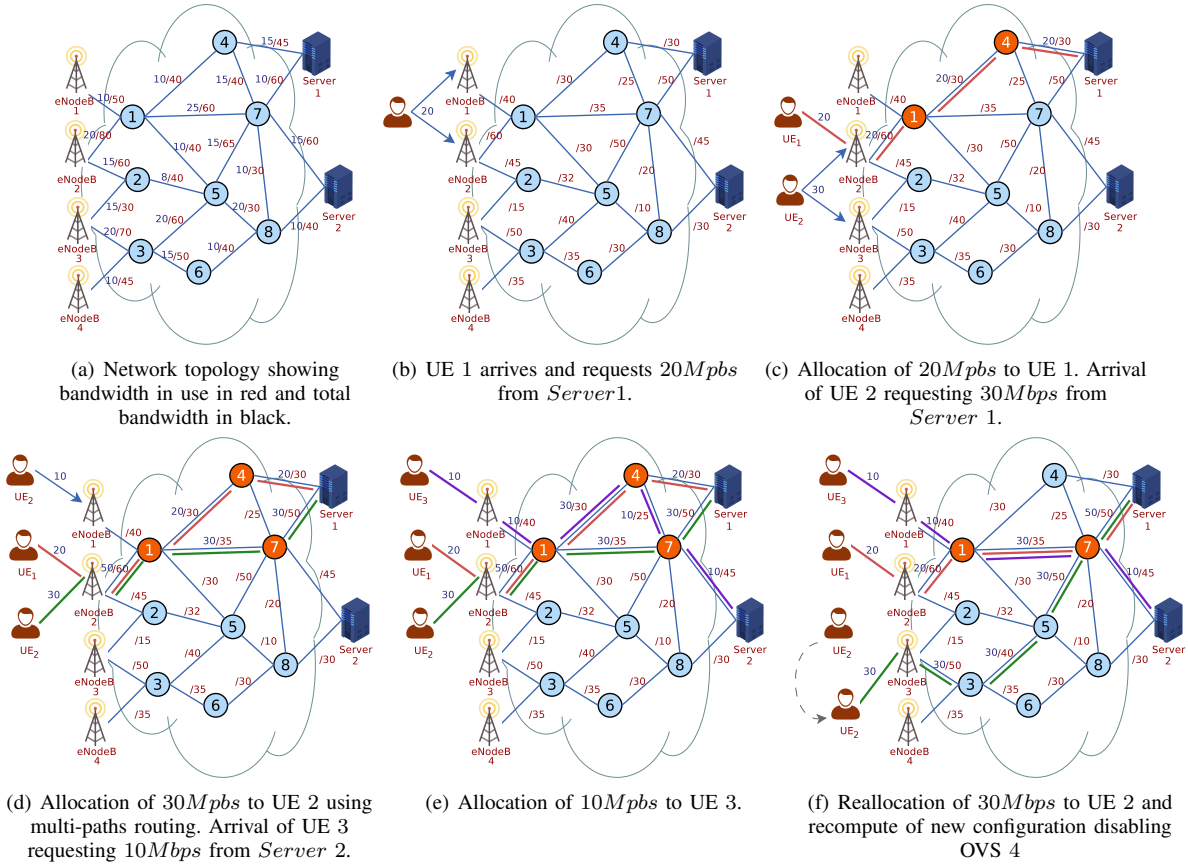
This section elaborates on the two optimization solutions. The first solution, dubbed optimal Full Paths Re-computation (FPR), aims to provide the optimal configuration considering all users. The second solution, named Heuristic full Paths Re-computation (HPR), resolves an approximated optimization problem aiming to keep the computation time of determining a configuration bounded. Both solutions reduce OPEX costs while ensuring end-to-end QoS by re-configuring the paths connecting all clients to corresponding servers. This strategy optimizes the resource allocation when new clients attach to the network and/or perform a handover. Both algorithms should be executed periodically or upon a significant network load alternation. At each execution, FPR and HPR will compute and configure new routes between different clients and their respective servers. The input of the FPR and HPR is a graph G, considering all client(s) that request a path or need a path modification towards a related server.

A. FPR: Full Paths Re-computation

In what follows, we describe the solution FPR related to the optimization problem (1a) - (1k). For all $u \in \mathcal{C} \cup \mathcal{O}$ and $v \in \mathcal{O} \cup \mathcal{S}$, we define the following variables. For each server $s \in \mathcal{S}$, we define a matrix $\mathcal{F}^s$ of integer variables that represents the traffic generated and forwarded to that server. Each element $\mathcal{F}_{i,j}^s$ represents the number of flows that shall be forwarded from i to j, whereby $i \in \mathcal{C} \cup \mathcal{O}$ and $j \in \mathcal{O} \cup \mathcal{S}$.

$$\min \sum_{\forall i \in \mathcal{O}} \mathcal{Y}_i \tag{1a}$$

s. t.



(a) Network topology showing bandwidth in use in red and total bandwidth in black.

(b) UE 1 arrives and requests $20Mpbs$ from $Server1$.

(c) Allocation of $20Mpbs$ to UE 1. Arrival of UE 2 requesting $30Mbps$ from $Server$ 1.

(d) Allocation of $30Mpbs$ to UE 2 using multi-paths routing. Arrival of UE 3 requesting $10Mbps$ from $Server$ 2.

(e) Allocation of $10Mpbs$ to UE 3.

(f) Reallocation of $30Mbps$ to UE 2 and recompute of new configuration disabling OVS 4

Figure 2: Illustrative example that shows the execution of FPR solution.

$$\forall i \in \mathcal{C} : \sum_{\forall j \in \eta(i)} \mathcal{X}_{i,j} = 1 \tag{1b}$$

$$\forall i \in \mathcal{C}, \forall j \in \eta(i) : \mathcal{T}_{i,j} = \sum_{\forall s \in \mathcal{S} \wedge i \in \mathcal{C}_s} \lambda_i^s \times \mathcal{X}_{i,j} \tag{1c}$$

$$\forall i \in \mathcal{O} : \sum_{\forall j \in \eta(i) \cap (\mathcal{C} \cup \mathcal{O})} \mathcal{T}_{j,i} = \sum_{\forall j \in \eta(i) \cap (\mathcal{O} \cup \mathcal{S})} \mathcal{T}_{i,j} \tag{1d}$$

$$\forall i \in \mathcal{C} \cup \mathcal{O}, \forall j \in \eta(i) \cap (\mathcal{O} \cup \mathcal{S}) : \mathcal{T}_{i,j} \leq \mathcal{W}_{i,j} \times \mathcal{X}_{i,j} \tag{1e}$$

$$\forall i \in \mathcal{O}, \forall j \in \eta(i) \cap (\mathcal{O} \cup \mathcal{S}) : \mathcal{X}_{i,j} \leq \mathcal{Y}_i \tag{1f}$$

$$\forall i \in \mathcal{O}, \forall j \in \eta(i) \cap (\mathcal{O} \cup \mathcal{C}) : \mathcal{X}_{j,i} \leq \mathcal{Y}_i \tag{1g}$$

$$\forall i \in \mathcal{S} : \sum_{\forall j \in \eta(i)} \mathcal{F}_{j,i}^i = |\mathcal{C}_i| \tag{1h}$$

$$\forall s \in \mathcal{S}, \forall i \in \mathcal{C}_s : \sum_{\forall j \in \eta(i)} \mathcal{F}_{i,j}^s = 1 \tag{1i}$$

$$\forall i \in \mathcal{O}, \forall s \in \mathcal{S} : \sum_{\forall j \in \eta(i) \cap (\mathcal{C} \cup \mathcal{O})} \mathcal{F}_{j,i}^s = \sum_{\forall j \in \eta(i) \cap (\mathcal{O} \cup \mathcal{S})} \mathcal{F}_{i,j}^s \tag{1j}$$

$$\forall s \in \mathcal{S}, \forall i \in \mathcal{C} \cup \mathcal{O}, \forall j \in \mathcal{O} \cup \mathcal{S} : 0 \leq \mathcal{F}_{i,j}^s \leq |\mathcal{C}_s| \times \mathcal{X}_{i,j} \tag{1k}$$

In the objective function (1a), we aim to minimize the number of OVSs used in configuring all the paths between clients and servers, fulfilling the following constraints. Constraint (1b) ensures that each user should be connected to only one eNodeB or gNB, which is associated with a corresponding OVS, whereby it receives and transmits data traffic. Constraint (1c) represents the traffic aggregated from a client towards all related servers that transverse the attached gNB. Constraint (1d) ensures that all traffic received by an OVS from its neighbors $\eta(i)$ or clients must be equal to the output traffic by that said OVS. Constraint (1e) ensures that the traffic forwarded from a node (i.e., client or OVS) to another node (i.e., OVS or server) should not exceed the capacity of the link that interconnects these two nodes. For $i \in \mathcal{C}$, the traffic generated from the client i should not exceed the link capacity between that client and the serving gNB. Likewise, for $i \in \mathcal{O}$, the aggregated traffic forwarded from an OVS i to its successor j should not exceed the capacity of that link. Constraints (1f) and (1g) ensure that only activated OVSs should participate in forwarding the traffic. None activated OVSs are not meant to be instantiated, and hence cannot participate in forwarding data traffic.

Constraints (1h), (1i), (1j) and (1k) ensure that the connectivity between the clients and their respective servers is established without any routing loops. Constraint (1h) ensures that the number of flows arriving at a given server equals

to the number of clients' requests. Constraint (1i) ensures that only one flow should be created between a client and server, supporting the OpenFlow protocol requirements related to the routing data traffic between clients and servers. In fact, OpenFlow can identify a flow by the source and destination address, while an SDN controller can make decisions only on a per flow basis. In the proposed solution, when an OVS receives two packets with different source addresses and the same destination, it can make distinct decisions in forwarding those packets to two different OVSs. Constraint (1j) ensures that the number of incoming flows to an OVS equals exactly to the number of outcoming flows. This constraint helps for ensuring the connectivity and preventing the creation of routing loops. Constraint (1k) forces the generated flow of a client to be routed only within the configured paths while avoiding loops.

Figure 2 elaborates a detailed example that illustrates the operation of the FPR solution considering a simple mobile network that consists of four eNodeBs/gNBs, a set of OVSs numbered from 1 to 8 and two servers. For clarity, we suppressed the SDN controller from Figure 2, but only show the effects of its operation. Figure 2(a) illustrates the network in its initial configuration, showing the bandwidth resources partially in use by residing tenants as highlighted in red numbers. Based on this topology, we derive the reference graph G by removing all used resources as depicted in Figure 2(b). Our FPR algorithm is executed periodically or upon a significant load alternation in order to ensure the desired user QoS.

A mobile user that needs to access a server should attach to an eNodeB/gNB in its vicinity as shown in Figure 2(b), where the arriving User Equipment (UE) 1 requires a 20 Mbps streaming flow from server 1. UE 1 is attached to eNodeB 2 instead of eNodeB 1, as eNodeB 2 can comply with the corresponding demand allowing load balancing. Figure 2(c) shows UE 2, which can connect either to eNodeB 1 or eNodeB 2 requesting a 30 Mbps streaming flow from server 1. The allocation of 30 Mbps to UE 2 through eNodeB 2 is depicted in Figure 2(d), demonstrating that OVS 1 uses multiple paths for ensuring QoS since the available capacity of a single path cannot support the requested UEs 1 and UEs 2 towards server 1. The same figure depicts the arrival of UE 3 that requests 10 Mbps from server 2, while figure 2(e) shows the corresponding resource allocation through eNodeB 1. UE 2 handover to eNodeB 3 is illustrated in Figure 2(f) highlighting the relocation of 30 Mbps on top of the updated reference graph that allows OVS 4 to be disabled since the traffic of UE 3 is redirected via a new route elaborating the efficiency of the FPR solution.

This example demonstrates the success of the FPR solution for connecting all the UEs with the related servers using the smallest number of OVSs, i.e. reducing the OPEX cost, without affecting the desired QoS. Every UE is connected to its corresponding server using a dedicated path without any loops. This is ensured thanks to the constraints (1b), (1c), (1h), (1i), (1j) and (1k). It is also observed that the received traffic equals the forwarding one at each OVS, which is aligned with the constraint (1e), while the assigned links between OVSs are not overloaded conforming constraint (1e). Finally, it is assured that only the activated OVS (i.e., colored green) participate in forwarding traffic between clients and servers, which respects the constraints (1f) and (1g).

Theorem 1. *The complexity of FPR is more than* $\mathcal{O}(2^{|\mathcal{O}|+\Delta\times(|V|+|\mathcal{S}|\times(|\mathcal{C}|+|\mathcal{O}|-1))})$*, where* Δ *denotes the graph degree of* G*.*

Proof. We have solved the optimization of FPR using Gurobi optimizer that uses the branch-and-bound method. Jeroslow [43] proved that the complexity of branch-and-bound for a binary linear program is $\mathcal{O}(2^{\mathcal{N}})$, where N is the number of binary variables in the optimization. In the FPR optimization problem, we have three kinds of binary and integer variables that require the branch-and-bound method. The first set of binary variables is $X_{i,j}$, for $i, j \in V$, that denotes if a node i selects j from its neighbors as successor. If we denote by Δ the graph degree of G, and based on the observation that servers do not have successors, we will have a maximum number of X variables $(|V|-|S|)\times\Delta$. The second set of binary variables is Y_o for $o \in \mathcal{O}$, which means that we have $|\mathcal{O}|$ of Y variables in the system. Finally, we have the set of integer variables $\mathcal{F}^s_{i,j}$, where $s \in \mathcal{S}$, $i \in \mathcal{C} \cup \mathcal{O}$ and $j \in \mathcal{S} \cup \mathcal{O}$. Based on the observation that node i selects its successor only from its neighbors, the number of variables of $\mathcal{F}$ is $|\mathcal{S}| \times |\mathcal{C} \cup \mathcal{O}| \times \Delta$. Also, based on the observation that the sets $\mathcal{C}$, $\mathcal{O}$ and $\mathcal{S}$ are independent, we have the number of the integer variables of $\mathcal{F}$ is $|\mathcal{S}| \times (|\mathcal{C}| + |\mathcal{O}|) \times \Delta$. Then, the number $\mathcal{N}$ of variables that we need to use branch-and-bound to solve the optimization of FPR equals to: $(|V| - |S|) \times \Delta + |\mathcal{O}| + |\mathcal{S}| \times (|\mathcal{C}| + |\mathcal{O}|) \times \Delta$. Thus, the run-time complexity of FPR is more than $\mathcal{O}(2^{|\mathcal{O}|+\Delta\times(|V|+|\mathcal{S}|\times(|\mathcal{C}|+|\mathcal{O}|-1))})$. □

B. HPR: Heuristic Paths Re-computation

As mentioned in Section I, the 5G network management and orchestrator communicates the resource demands among the indicated end points, e.g. gNBs and UPFs (as shown in Fig. 1), considering the user mobility via the mobile-transport API towards the transport network SDN controller. The SDN controller in turn, executes the proposed algorithms to optimize the resource allocation and network utilization. The proposed algorithms ensure an efficient life cycle management for network slicing in the mobile network allowing the transport network to assure the targeted KPIs at a low cost. The algorithms introduced in this paper are executed periodically or upon a significant network load alternation. The suggested algorithms should be executed as background processes to feed the forwarding mechanisms that operate using the routes already determined. Hence, they should not affect the QoS at the data plane. After the convergence of an algorithm, routing alternations would be enforced using the SDN transport controller(s). To follow dynamic traffic changes in the network a periodic or a threshold based approach should be employed to trigger the corresponding algorithm to re-compute fast potential routing changes.

The FPR optimization solution, described in the previous subsection, can provide an optimal configuration by using

linear integer programming that leverages the branch and bound method. Unfortunately, the FPR solution could take a long time before providing an optimal configuration due to the use of branch and bound method for solving the mixed linear integer programming optimization. The proposed optimization model of FPR cannot be time efficient for providing network configurations for a big network. However, it could serve as a baseline approach for evaluating different heuristics as suggested later.

In this section, we introduce the HPR heuristic, which provides an efficient configuration at a reasonable time. HPR explores the Dijkstra shortest-path algorithm for allocating different paths between UEs and servers. In contrast to FPR solution, the HPR one can run at the order of 10 MHz or at least every 100 ms considering a periodic approach. Algorithm 1 illustrates the different steps of the HPR Algorithm. The HPR Algorithm uses $G(V, E, \mathcal{W})$, $\mathcal{C}$ and $\mathcal{S}$ as inputs, while its output consists of the activated OVSs Φ and the generated paths $\mathcal{P}$. It starts by initializing the selected OVSs Φ with an empty set (Algorithm 1: Line 1). Then, the HPR algorithm enters a loop to calculate the routing paths between each client and server (Algorithm 1: Lines $2-50$). An access point Ψ_c of a client $c \in \mathcal{C}$ should be able to handle the data traffic between that client c and its servers $\mathcal{S}_c$. A forwarding node $o \in \mathcal{O}$ can be selected as an access point, unless the bandwidth between the client c and o is less than the expected $\mathcal{S}_c$ (i.e., $\mathcal{W}_{c,o} < \sum_{\forall s \in \mathcal{S}_c} \lambda_s^c$). Hence, the algorithm initially removes the forwarding nodes that cannot be assigned as access points Ψ_c for a client $c \in \mathcal{C}$ (Algorithm 1: Lines $3-8$). For each client $c \in \mathcal{C}$, a set of nodes $\mathcal{V}$ is initialized considering the set V as input (Algorithm 1: Line 3). Then, each forwarding node ($o \in \eta(c) \cap \mathcal{O}$) that is a neighbor of $c \in \mathcal{C}$ should be removed if it cannot be used as an access point (Algorithm 1: Lines $4-8$).

As the next step, paths are calculated, between the client c and its corresponding servers, one by one (i.e., $\forall s \in \mathcal{S}_c$). For this reason, each server $s \in \mathcal{S}_c$ associated with a client c, relates to an inner loop in the HPR algorithm (Algorithm 1: Lines $9-49$). In the inner loop, two temporary variables $\hat{\mathcal{V}}$ and $\hat{\mathcal{E}}$ are initialized as $\mathcal{V}$ and E, respectively, (Algorithm 1: Lines $10-11$), in order to avoid affecting the values of $\mathcal{V}$ and $\mathcal{E}$ when a new path is calculated. Then, in the inner loop, all links that cannot handle the expected traffic between the client c and its server s are removed (Algorithm 1: Lines $12-16$). So none other clients and servers are involved in forwarding traffic, except c and s (Algorithm 1: Line 17). In order to reduce the complexity of the algorithm, any other node that cannot participate in forwarding traffic should also be removed. Formally, a node $o \in \mathcal{O} \cap \hat{\mathcal{V}}$ cannot participate in traffic forwarding if it is a leaf node ($|\eta(o)| = 1$). For this reason, in the HPR algorithm, an infinite loop is defined to remove all the leaf nodes in a cascading way (Algorithm 1: Lines $18-24$).

The HPR algorithm aims to re-use the activated OVSs, indicated by Φ, as much as possible in order to reduce OPEX, while ensuring QoS. In the algorithm, a new graph $\mathcal{G}(\hat{\mathcal{V}}, \hat{\mathcal{E}}, \hat{\omega})$

Algorithm 1 HPR Algorithm.

Require:
$G(V, E, \mathcal{W})$: Network graph.
$\mathcal{C}$: List of UEs.
$\mathcal{S}$: List of servers in the network.

1: $\Phi = \emptyset$
2: **for** $c \in \mathcal{C}$ **do**
3: $\mathcal{V} = V$
4: **for** $o \in \eta(c) \cap \mathcal{O}$ **do**
5: **if** $\mathcal{W}_{c,o} < \sum_{\forall s \in \mathcal{S}_c} \lambda_s^c$ **then**
6: $\mathcal{V} = \mathcal{V}/o$
7: **end if**
8: **end for**
9: **for** $s \in \mathcal{S}_c$ **do**
10: $\hat{\mathcal{V}} = \mathcal{V}$
11: $\hat{\mathcal{E}} = E$
12: **for** $(u, v) \in \hat{\mathcal{E}}$ **do**
13: **if** $\mathcal{W}_{u,v} < \lambda_s^c$ **then**
14: $\hat{\mathcal{E}} = \mathcal{E}/(u, v)$
15: **end if**
16: **end for**
17: $\hat{\mathcal{V}} = \hat{\mathcal{V}}/((\mathcal{S} \cup \mathcal{C})/\{c, s\})$
18: **while** True **do**
19: **if** $\exists o \in \mathcal{O} \cap \hat{\mathcal{V}} - \{\Psi_c\} \wedge |\eta(o)| \leq 1$ **then**
20: $\hat{\mathcal{V}} = \hat{\mathcal{V}}/o$
21: **else**
22: Break
23: **end if**
24: **end while**
25: **for** $u \in \hat{\mathcal{V}}$ **do**
26: **if** $u \in \Phi$ **then**
27: $u.\omega = 1$
28: **else**
29: $u.\omega = |\hat{\mathcal{V}}|$
30: **end if**
31: **end for**
32: $\hat{\omega} = \emptyset$
33: **for** $(u, v) \in \hat{\mathcal{E}}$ **do**
34: $\hat{\omega}_{u,v} = \frac{u.\omega + v.\omega}{2}$
35: **end for**
36: **if** $\Psi_c == \emptyset$ **then**
37: $\hat{c} = c$
38: **else**
39: $\hat{c} = \Psi_c$
40: **end if**
41: $\mathcal{P}_{c,s} = short\_path(\hat{c}, s, \hat{\mathcal{V}}, \hat{\mathcal{E}}, \hat{\omega})$
42: **if** $\Psi_c == \emptyset$ **then**
43: $\Psi_c = \mathcal{P}_{c,s}[1]$
44: **end if**
45: $\Phi = \Phi \cup (\mathcal{P}_{c,s} \cap \mathcal{O})$
46: **for** $0 \leq i < |\mathcal{P}_{c,s}|$ **do**
47: $\mathcal{W}_{i,i+1} = \mathcal{W}_{i,i+1} - \lambda_{c,s}$
48: **end for**
49: **end for**
50: **end for**

is generated from $G(V, E, \mathcal{W})$ by selecting the OVSs from Φ to interconnect c with s (Algorithm 1: Lines $25-35$). To accomplish this a weight $\hat{\omega}_{u,v}$ of an edge $(u, v) \in \hat{\mathcal{E}}$ is defined according to the nature of u and v as follows:

- If $u, v \in \Phi$, then $\hat{\omega}_{u,v} = 1$;
- If $u \in \Phi \wedge v \notin \Phi$, then $\hat{\omega}_{u,v} = \frac{|\hat{\mathcal{V}}|+1}{2}$;
- If $u, v \notin \Phi$, then $\hat{\omega}_{u,v} = |\hat{\mathcal{V}}|$;

A new variable $\hat{c}$ is defined to assign the client c if the access point Ψ_c is not yet selected. Otherwise, this variable should assign the access point Ψ_c (Algorithm 1: Lines $36-40$). Formally, $\hat{c}$ is used to compute the Dijkstra shortest-path algorithm between the access point Ψ_c or the client c, and the server s (Algorithm 1: Line 41). Using Dijkstra shortest-path algorithm with $\hat{\omega}$, it is ensured that the selection of the lowest number of OVSs does not belong to Φ.

Finally, the algorithm defines the access point of c if it is not yet defined (Algorithm 1: Lines $42-44$). Formally, the access point c is defined as the first hop and should be used to interconnect c with s (i.e., $\Psi_c = \mathcal{P}_{c,s}[1]$). Then, the selected OVSs, Φ, is updated by considering the additional OVSs used to interconnect c with s (Algorithm 1: Line 45). Lastly, $\mathcal{W}$ is updated by removing the expected traffic between the client c and the server s (Algorithm 1: Lines $46-48$). $\mathcal{W}$ is updated in order to ensure that the required QoS between all the clients and servers is not affected.

Theorem 2. *The run time complexity of HPR is $\mathcal{O}(|\mathcal{C}|(\Delta + |\mathcal{S}|(2|E| + 3|V|)))$, where Δ denotes the graph degree of G.*

Proof. HPR has a long loop that starts at line 2 and ends at line 50 (Algorithm 1: HPR Algorithm). In this loop, we iterate on the number of clients $\mathcal{C}$, thus its complexity is $\mathcal{O}(|\mathcal{C}|)$. This loop has two inner loops. The first loop that starts at line 4 and ends at line 8. In this loop, we iterate on the OVSs that are neighbors of the client c, and hence the complexity of this loop is $\mathcal{O}(\Delta)$, such that Δ is the graph degree. Meanwhile, the second loop starts at line 9 and ends at line 49. In this loop iterates on the servers, and hence its complexity is $\mathcal{O}(|\mathcal{S}|)$. This loop consists of 5 parts. The first part starts at line 12 and ends at line 16, and its complexity is $\mathcal{O}(|E|)$. The second part starts at line 18 and ends at line 24, and its complexity $\mathcal{O}(|V|)$. The third part starts at line 25 and ends 31, and its complexity is $\mathcal{O}(|V|)$. The fourth part starts at line 33 and ends at line 35, and its complexity is $\mathcal{O}(|E|)$. Finally, the fifth step is the last one that starts at line 46 and ends at the line 48, which has run-time complexity of $\mathcal{O}(|V|)$. Considering all the aforementioned steps, the complexity of HPR is $\mathcal{O}(|\mathcal{C}|(\Delta + |\mathcal{S}|(2|E| + 3|V|)))$. □

V. PPR: Partial Paths Re-computation

The FPR and HPR solutions aim to get an optimal configuration by recomputing the paths for all the UEs, both stationary and mobile. For instance, when the execution period is elapsed, even when only one UE joins the network or handovers to another location, both FPR and HPR solutions may recompute the paths for all the UEs in order to optimally configure the network. In real network deployments, UEs could be in the order of hundreds of thousands even in smaller regions, which may affect the computation time for converging in a new path configuration. In order to mitigate such a problem, herein, we suggest a new solution, named Partial Paths Re-computation (PPR) that aims to lightweight the re-computation of the users' paths. The basic idea of the PPR solution is to avoid the re-configuration of paths related to stationary UEs that remain constant from the previous iteration (i.e., same UE locations and QoS requests). In fact, the PPR solution considers only newly attached or handovered UEs. Other UEs that keep a constant state, including the associated resources, are removed from the network graph before starting the recomputation. Conceptually, the PPR solution introduces a filter phase that removes the stationary UEs and the associated bandwidth utilization resources from the network graph before executing the optimization algorithm.

As mentioned before, we denote by $G(V, E, \mathcal{W})$ a weighted graph that reflects the current state of the network, where $V = \mathcal{C} \cup \mathcal{O} \cup \mathcal{S}$ represents a set of clients $\mathcal{C}$, OVSs $\mathcal{O}$, and servers S, respectively. Let $\mathcal{C} = \dot{\mathcal{C}} \cup \ddot{\mathcal{C}}$, where $\dot{\mathcal{C}}$ denotes the set of stationary UEs since the last execution of PPR and $\ddot{\mathcal{C}}$ stands for the set of newly attached and/or handovered UEs. Note that each UE $i \in \dot{\mathcal{C}}$ has already been allocated predefined paths towards the corresponding server. We have two sets of OVSs: i) Activated OVSs $\dot{\mathcal{O}}$ that participate in forwarding packets towards and from stationary UEs $\dot{\mathcal{C}}$ and ii) none-activated OVSs $\ddot{\mathcal{O}} = \mathcal{O} - \dot{\mathcal{O}}$ that are excluded from the routing process. We denote by $\ddot{G}(\mathcal{V}, \mathcal{E}, \omega)$ an updated version of graph G that excludes stationary UEs and the associated bandwidth resource utilization. $\ddot{G}(\mathcal{V}, \mathcal{E}, \omega)$ is derived from G as follows. First, $\mathcal{V}$ is generated from V by removing the stationary UEs $\dot{\mathcal{C}}$, hence $\mathcal{V} = \ddot{\mathcal{C}} \cup \mathcal{O} \cup \mathcal{S}$. Second, $\mathcal{E}$ is generated from E by removing any edge $(a, b) \in E$ from $\mathcal{E}$, such that $a \in \dot{\mathcal{C}}$ or $b \in \dot{\mathcal{C}}$. Finally, ω is generated from $\mathcal{W}$ by removing the resources used by $\dot{\mathcal{C}}$. When generating ω we have taken into account the resources of the path between a stationary client and its corresponding server. At each iteration, the aim of PPR is to use the minimum number of none-activated OVSs $\ddot{\mathcal{O}}$ for connecting new and handovered UEs.

The Algorithm 2 describes the main functionality of the PPR solution. Initially, PPR gets the graph G and the list of servers $\mathcal{S}$ as input and then it has an infinite loop, whereby it checks if there are any updates in the network state, i.e. by the arrival of new and/or handover UEs. If so, $\ddot{G}(\mathcal{V}, \mathcal{E}, \omega)$ would be derived from G by removing the stationary UEs $\dot{\mathcal{C}}$. Then, the set of none-activated OVS $\ddot{\mathcal{O}}$ is specified based on $\mathcal{O}$ by removing OVSs already in use by UEs $\dot{\mathcal{C}}$. Last but not least, the optimization problem defined by the function $Optimization_{PPR}(G, \ddot{G}, \dot{\mathcal{C}}, \mathcal{S}, \ddot{\mathcal{O}})$ is executed for re-configuring network paths. In what follows, we define the different variables used in the optimization problem $Optimization_{PPR}(G, \ddot{G}, \dot{\mathcal{C}}, \mathcal{S}, \ddot{\mathcal{O}})$. After executing $Optimization_{PPR}(G, \ddot{G}, \dot{\mathcal{C}}, \mathcal{S}, \ddot{\mathcal{O}})$, the network reconfiguration decisions enable the SDN controller(s) to perform the appropriate network state modifications. Finally, the algorithm waits for a significant change in the network state before executing the optimization of PPR again.

Algorithm 2 PPR Algorithm.

Input:
G: The input original graph.
$\mathcal{S}$: List of servers in the network.

while true **do**
  if $\mathcal{C}_2 \neq \emptyset$ **then**
    $\ddot{G} = G.copy()$
    $\dot{\mathcal{C}} = \mathcal{C} - \ddot{\mathcal{C}}$
    $\ddot{\mathcal{O}} = \ddot{G}.remove(\dot{\mathcal{C}})$
    $Optimization_{PPR}(G, \ddot{G}, \dot{\mathcal{C}}, \mathcal{S}, \ddot{\mathcal{O}})$
  end if
  $wait()$
end while

For each server $s \in \mathcal{S}$ and UE $c \in \mathcal{C}$, we define an integer variable $F^{c,s}$ that mimics packet flow generated from client c towards the server s. Each element $F^{c,s}_{i,j}$ represents the number of flow packets from UE c to server s, which should be forwarded from $i \in \mathcal{C} \cup \mathcal{O}$ to $j \in \mathcal{O} \cup \mathcal{S}$. In what follows, we define the optimization problem $Optimization_{PPR}(G, \ddot{G}, \dot{\mathcal{C}}, \mathcal{S}, \ddot{\mathcal{O}})$.

$$\textbf{min} \sum_{\forall i \in \ddot{\mathcal{O}}} \mathcal{Y}_i \tag{2a}$$

s. t.

$$\forall s \in \mathcal{S}, \forall c \in \mathcal{C}_s \cap \ddot{\mathcal{C}} : \sum_{\forall j \in \eta(c)} \mathcal{X}^{c,s}_{c,j} = 1 \tag{2b}$$

$$\forall s_1 \in \mathcal{S}, \forall s_2 \in \mathcal{S}, \forall c \in \mathcal{C}_{s_1} \cap \mathcal{C}_{s_2} \cap \ddot{\mathcal{C}}, \forall j \in \eta(c) : \mathcal{X}^{c,s_1}_{c,j} = \mathcal{X}^{c,s_2}_{c,j} \tag{2c}$$

$$\forall s \in \mathcal{S}, \forall c \in \mathcal{C}_s \cap \ddot{\mathcal{C}}, \forall j \in \eta(c) : \mathcal{T}^{c,s}_{c,j} = \lambda^s_c \times \mathcal{X}^{c,s}_{c,j} \tag{2d}$$

$$\forall s \in \mathcal{S}, \forall c \in \mathcal{C}_s \cap \ddot{\mathcal{C}}, \forall i \in \mathcal{O} : \sum_{\forall j \in \eta(i) \cap (\{c\} \cup \mathcal{O})} \mathcal{T}^{c,s}_{j,i} = \sum_{\forall j \in \eta(i) \cap (\mathcal{O} \cup \mathcal{S})} \mathcal{T}^{c,s}_{i,j} \tag{2e}$$

$$\forall s \in \mathcal{S}, \forall c \in \mathcal{C}_s \cap \ddot{\mathcal{C}}, \forall i \in \{c\} \cup \mathcal{O}, \forall j \in \eta(i) \cap (\mathcal{O} \cup \mathcal{S}) : \mathcal{T}^{c,s}_{i,j} \leq \mathcal{W}_{i,j} \times \mathcal{X}^{c,s}_{i,j} \tag{2f}$$

$$\forall s \in \mathcal{S}, \forall c \in \mathcal{C}_s \cap \ddot{\mathcal{C}} : \sum_{\forall i \in \eta(c) \cap (\mathcal{O} \cup \{s\})} \mathcal{T}^{c,s}_{c,i} = \sum_{\forall i \in \eta(s) \cap (\mathcal{O} \cup \{c\})} \mathcal{T}^{c,s}_{i,s} \tag{2g}$$

$$\forall c \in \ddot{\mathcal{C}}, \forall j \in \eta(c) \cap (\mathcal{O} \cup \mathcal{S}) : \sum_{\forall s \in \mathcal{S}} \mathcal{T}^{c,s}_{c,j} \leq \mathcal{W}_{c,j} \tag{2h}$$

$$\forall i \in \mathcal{O}, \forall j \in \eta(i) \cap \mathcal{O} : \sum_{\forall s \in \mathcal{S}} \sum_{\forall c \in \mathcal{C}_s \cap \ddot{\mathcal{C}}} \mathcal{T}^{c,s}_{i,j} \leq \mathcal{W}_{i,j} \tag{2i}$$

$$\forall s \in \mathcal{S}, \forall i \in \eta(s) : \sum_{\forall c \in \mathcal{C}_s \cap \ddot{\mathcal{C}}} \mathcal{T}^{c,s}_{i,s} \leq \mathcal{W}_{i,s} \tag{2j}$$

$$\forall c \in \ddot{\mathcal{C}}, \forall s \in \mathcal{S}, \forall i \in \ddot{\mathcal{O}}, \forall j \in \eta(i) \cap (\mathcal{O} \cup \{c\}) : \mathcal{X}^{c,s}_{j,i} \leq \mathcal{Y}_i \tag{2k}$$

$$\forall c \in \ddot{\mathcal{C}}, \forall s \in \mathcal{S}, \forall i \in \ddot{\mathcal{O}}, \forall j \in \eta(i) \cap (\mathcal{O} \cup \mathcal{S}) : \mathcal{X}^{c,s}_{i,j} \leq \mathcal{Y}_i \tag{2l}$$

$$\forall s \in \mathcal{S}, \forall c \in \ddot{\mathcal{C}} \cap \mathcal{C}_s, \forall i \in \mathcal{O} : \sum_{\forall j \in \eta(i) \cap (\mathcal{O} \cup \{s\})} \mathcal{X}^{c,s}_{i,j} \leq 1 \tag{2m}$$

$$\forall s \in \mathcal{S}, \forall c \in \ddot{\mathcal{C}} \cap \mathcal{C}_s, \forall i \in \mathcal{O} : \sum_{\forall j \in \eta(i) \cap (\mathcal{O} \cup \{c\})} \mathcal{X}^{c,s}_{j,i} \leq 1 \tag{2n}$$

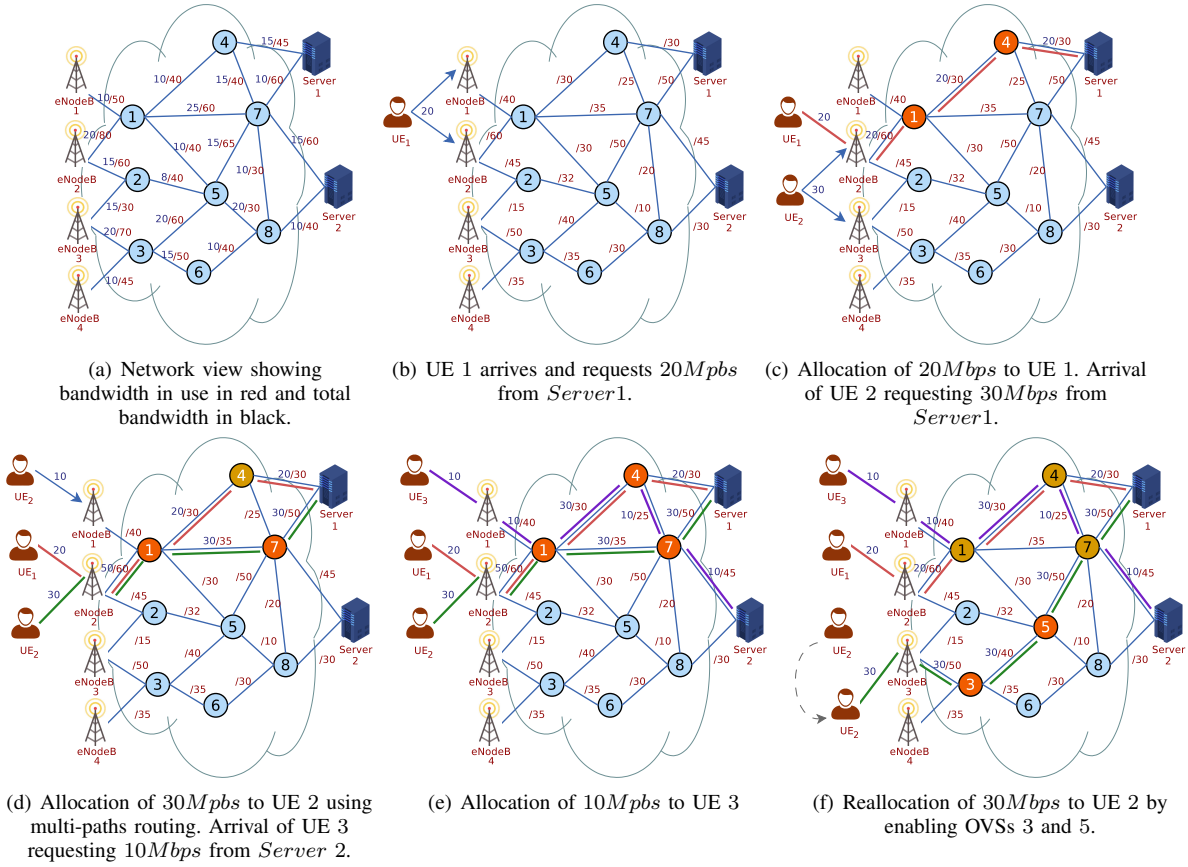
$$\forall s \in \mathcal{S}, \forall c \in \mathcal{C}_s \cap \ddot{\mathcal{C}} : \sum_{\forall j \in \eta(c)} F^{c,s}_{c,j} = 1 \tag{2o}$$

$$\forall s \in \mathcal{S} : \sum_{\forall c \in \mathcal{C}_s \cap \ddot{\mathcal{C}}} \sum_{\forall j \in \eta(s)} F^{c,s}_{j,s} = |\mathcal{C}_s| \tag{2p}$$

$$\forall c \in \ddot{\mathcal{C}}, \forall s \in \mathcal{S}, \forall i \in \mathcal{O} : \sum_{\forall j \in \eta(i) \cap (\{c\} \cup \mathcal{O})} F^{c,s}_{j,i} = \sum_{\forall j \in \eta(i) \cap (\mathcal{O} \cup \{s\})} F^{c,s}_{i,j} \tag{2q}$$

$$\forall s \in \mathcal{S}, \forall c \in \mathcal{C}_s \cap \ddot{\mathcal{C}}, \forall i \in \{c\} \cup \mathcal{O}, \forall j \in \mathcal{O} \cup \mathcal{S} : 0 \leq F^{c,s}_{i,j} \leq |\mathcal{C}_s| \times \mathcal{X}^{c,s}_{i,j} \tag{2r}$$

In the objective function (2a), PPR aims to minimize the utilization of none-activated OVSs $\ddot{\mathcal{O}}$ by using the OVSs $\dot{\mathcal{O}}$ already in use. This helps in reducing the overall OPEX cost. In the optimization problem, the network graph is updated by removing the stationary UEs and updating the capacity of the associated paths by removing the allocated resources in use. Constraints (2b) and (2c) ensure that each UE is attached to only one eNodeB, with the traffic generated towards the related servers transversing through that same eNodeB. Constraint (2d) computes the amount of traffic $\mathcal{T}^{c,s}_{c,j}$ generated from a UE $c \in \mathcal{C}_s$ to the corresponding server $s \in \mathcal{S}$ through a neighbor j. $\mathcal{T}^{c,s}_{c,j} = \lambda^s_c$ only if j is selected as the attach point for c, otherwise $\mathcal{T}^{c,s}_{c,j} = 0$. Constraint (2e) ensures that the received traffic at each OVS from a specified UE $c \in \mathcal{C}_s \cap \ddot{\mathcal{C}}$ equals the forwarded traffic from that OVS towards the corresponding server $s \in \mathcal{S}$. This constraint helps to preserve the traffic in the network, assuring paths without loops.



(a) Network view showing bandwidth in use in red and total bandwidth in black.

(b) UE 1 arrives and requests $20Mpbs$ from $Server1$.

(c) Allocation of $20Mbps$ to UE 1. Arrival of UE 2 requesting $30Mbps$ from $Server1$.

(d) Allocation of $30Mpbs$ to UE 2 using multi-paths routing. Arrival of UE 3 requesting $10Mbps$ from $Server$ 2.

(e) Allocation of $10Mpbs$ to UE 3

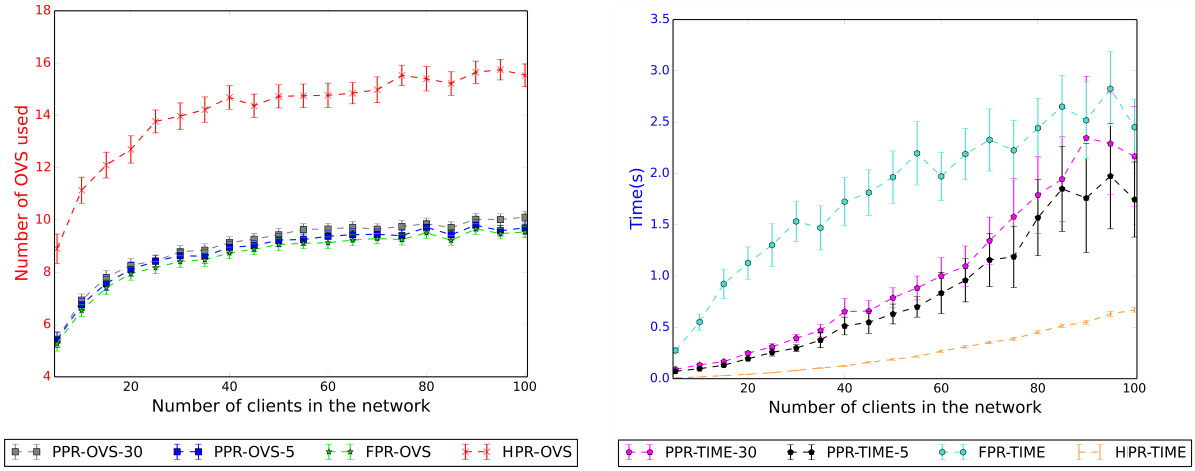
(f) Reallocation of $30Mbps$ to UE 2 by enabling OVSs 3 and 5.

Figure 3: Illustrative example that shows the execution of PPR solution.

Constraint (2f) ensures that each link in the network is not handling traffic load beyond its capacity. Constraint (2g) ensures that the dedicated bandwidth from a client to a specified server remains the same through the entire path. Constraints (2h), (2i) and (2j) ensure that the aggregated traffic through a dedicated link should not exceed its capacity. Constraints (2k) and (2l) ensure that the OVSs in $\ddot{\mathcal{O}}$ should participate in routing packets if and only if they are activated. Constraints (2m), (2n), (2o), (2p), (2q) and (2r) ensure the connectivity between the clients and their respective servers, while preventing routing loops. Constraint (2m) ensures that an OVS $i \in \mathcal{O}$ can have at most one successor for a specified path between a client and a server, while constraint (2m) ensures that the OVS i has only at most one predecessor. These two constraints help for preventing loops in the paths. Constraint (2o) ensures that only one flow is established from a UE towards its respective server satisfying the OpenFlow protocol requirements for routing data traffic between clients and servers. Constraint (2p) ensures the number of flows arriving to a specified server equals exactly the number of its clients. Constraint (2q) ensures that the number of incoming flows to an OVS exactly equals to the number of outcoming flows from that OVS. This constraint helps ensuring connectivity, preventing routing loops. Constraint (2r) forces a client flow to be routed only within the allocated path avoiding loops.

Figure 3 illustrates an example for operating the PPR solution. We have adopted the same network used for the description of FPR to show the main differences. Figure 3(a) illustrates the network in its initial configuration, showing the bandwidth resources partially in use as highlighted in red numbers. Figure 3(b) depicts the arrival of UE 1 requesting a 20 Mbps streaming flow from server 1. PPR initially computes both $\ddot{G}$ and $\ddot{\mathcal{O}}$ from G and as UE 1 is the first UE attached to the network, G and $\ddot{G}$ as well as $\ddot{\mathcal{O}}$ and $\mathcal{O}$ are identical. Using the $Optimization_{PPR}$ procedure, UE 1 is attached to eNodeB 2, which can comply with the desired QoS demand, while OVS 1 and OVS 4 are selected for routing the traffic between UE 1 and server 1.

The arrival of UE 2 that resides in the vicinity eNodeB 1 or eNodeB 2 is shown in Figure 3(c). UE 2 requests a 30 Mbps flow from server 1. Before starting the $Optimization_{PPR}$ procedure, PPR updates $\ddot{G}$ by removing UE 1 and the corresponding allocated bandwidth from G, while $\ddot{\mathcal{O}}$ is updated by removing OVSs 1 and OVSs 4. Then the PPR executes the $Optimization_{PPR}$ procedure as shown in Figure 3(d) allocating 30 Mbps to UE 2 through eNodeB 2, and using OVS 1 and OVS 7. This figure demonstrates that OVS 1 uses diverse paths to forward flows ensuring the desired QoS between UE 1 and UE 2 from one side, and server 1 from another. The arrival of UE 3 that requests a 10 Mbps flow from server 2 is also shown in Figure 3(d). The PPR solution



(a) The cost of proposed solutions

(b) The run-time of proposed solutions

Figure 4: Impact of number of clients on different solutions with variant mobility patterns

performs the following preparations: i) $\ddot{G}$ by removing UE 2 and the corresponding allocated bandwidth and ii) removing OVS 7 from $\ddot{\mathcal{O}}$, before re-computing the network resources, allocating 10 Mbps to UE 3 as illustrated in Figure 3(e). Figure 3(f) depicts the handover of UE 2 towards eNodeB 3 and demonstrates the relocation of a 30 Mbps flow on top of the updated reference graph by activating OVSs 3 and OVSs 5. In contrast to the FPR solution that disabled OVS 4, PPR does not disable this OVS as it is still in use by UE 3, and only partial path re-computation is adopted instead.

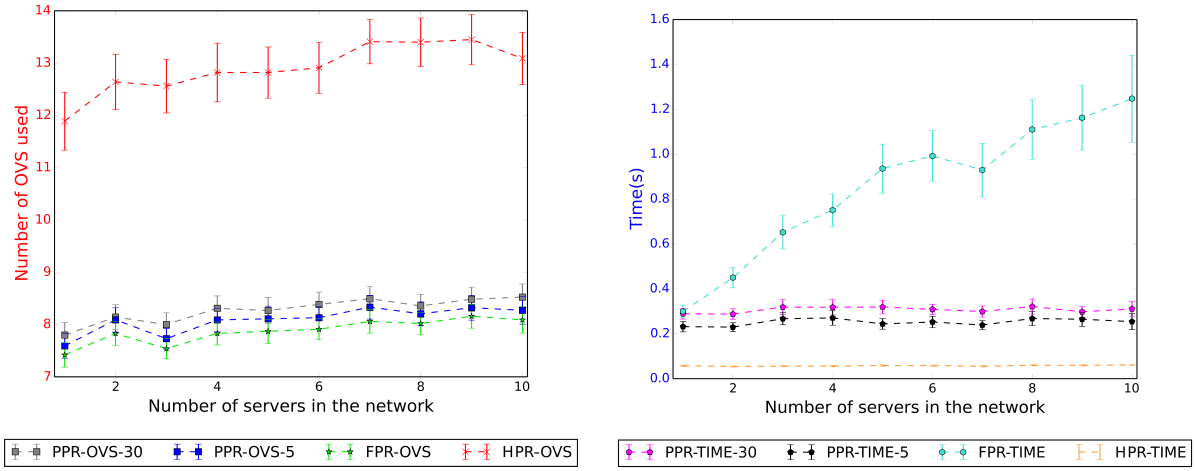
From this example, the PPR solution succeeded in connecting all UEs with their corresponding servers using a near optimal solution with dedicated paths avoiding loops, while reducing the execution time. This is ensured thanks to the constraints (2b), (2c), (2o), (2p), (2q) and (2r). It is observed that the incoming traffic equals the outgoing one at each OVS, which respects the constraint (2e) and (2g). The assigned links between OVSs are not overloaded according to constraints (2h), (2i) and (2j). Finally, it is noted that only the activated OVS (i.e., in green color) participated in forwarding traffic between clients and servers based on the constraints (2k) and (2l).

Theorem 3. *The complexity of PPR is more than* $\mathcal{O}(2^{|\mathcal{O}|+2\Delta\times|\mathcal{C}|\times|\mathcal{S}|\times(|\mathcal{C}|+|\mathcal{O}|)})$*, where* Δ *denotes the graph degree of* G*.*

Proof. A Gurobi optimizer is also used for solving the linear integer programming model of PPR solution. We have also used the branch-and-bound method for getting the final configuration. According to Jeroslow [43] the complexity of branch-and-bound for a binary linear program is $\mathcal{O}(2^{\mathcal{N}})$, where N is the number of binary variables. In the PPR optimization problem, we have three kinds of binary and integer variables that require the branch-and-bound method. The first set of binary variables is $X_{i,j}^{c,s}$, for $i, j \in V$, $c \in \mathcal{C}$ and $s \in \mathcal{S}$. This variable denotes if a node i selects j from its neighbors as successor for handling the traffic between the client c and the server s. If we denote by Δ the graph degree of G, and based on the observation that servers do not have successors, the maximum number of X variables is $|\mathcal{C}| \times |\mathcal{S}| \times (|\mathcal{C} \cup \mathcal{O}|) \times \Delta$. As the sets $\mathcal{C}$ and $\mathcal{O}$ are independent, the maximum number of X variables is $|\mathcal{C}| \times |\mathcal{S}| \times (|\mathcal{C}| + |\mathcal{O}|) \times \Delta$. The second set of binary variables is Y_o for $o \in \mathcal{O}$, which means that we have $|\mathcal{O}|$ of Y variables in the system. Finally, we have the set of integer variables $\mathcal{F}_{i,j}^{c,s}$, where $c \in \mathcal{C}$, $s \in \mathcal{S}$, $i \in \mathcal{C} \cup \mathcal{O}$ and $j \in \mathcal{S} \cup \mathcal{O}$. Based on the observation that node i selects its successor only from its neighbors, the number of variables of $\mathcal{F}$ is $||\mathcal{C}| \times \mathcal{S}| \times |\mathcal{C} \cup \mathcal{O}| \times \Delta$. Also, based on the observation that the sets $\mathcal{C}$, $\mathcal{O}$ and $\mathcal{S}$ are independent, we have the number of the integer variables of $\mathcal{F}$ is $|\mathcal{C}| \times |\mathcal{S}| \times (|\mathcal{C}| + |\mathcal{O}|) \times \Delta$. Then, the number $\mathcal{N}$ of variables that need to use branch-and-bound to solve the optimization of PPR equals to: $|\mathcal{O}| + 2 \times |\mathcal{C}| \times |\mathcal{S}| \times (|\mathcal{C}| + |\mathcal{O}|) \times \Delta$. Thus, the run-time complexity of PPR is more than $\mathcal{O}(2^{|\mathcal{O}|+2\Delta\times|\mathcal{C}|\times|\mathcal{S}|\times(|\mathcal{C}|+|\mathcal{O}|)})$. □

VI. Simulation Set-up and Result Analysis

This section introduces the simulation set-up and provides an analysis of the obtained results for each proposed solution. We have implemented and evaluated the FPR, PPR, and HPR solutions using Python, an extended package for graph theory called Networkx and Gurobi Optimizer software. All the execution time measurements are based on an Intel Core $i5$ 3570 CPU system clocked at 3.4 GHz, with 16 GB of RAM, and running Ubuntu 16.04. Our proposed solutions were evaluated by varying the number of clients, servers, and OVSs on the network while using random graph topologies to change their respective connectivity. We ran 100 repetitions, changing the clients' positions and computed the number of OVSs used and the execution time. Hereafter, we present the mean and 95% confidence interval of the number of OVSs used and the execution time in seconds. In each evaluation repetition, the positions of clients and servers were uniformly distributed,



(a) The cost of proposed solution

(b) The run-time of proposed solutions

Figure 5: Impact of the number of servers on solutions with variant mobility patterns

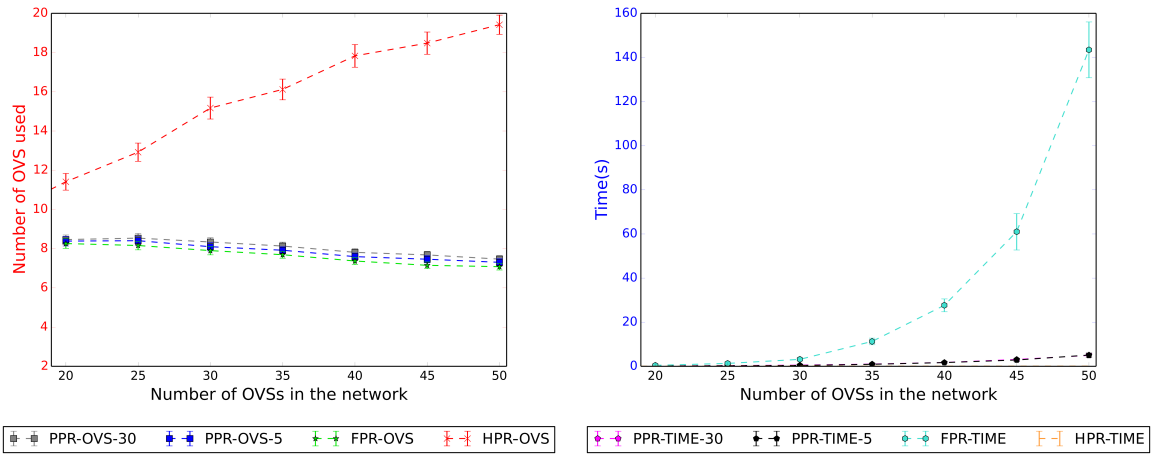
with a client connected to a single OVS as described in Section III. In all of the results presented, the following simulation parameters were introduced as can be seen in the legends of Figures 4 to 6: i) the solution name, i.e., FPR, PPR, and HPR, ii) the type of results including OVS for the number of SDN-enabled switches or TIME is for the execution time, and iii) the percentage of user mobility, i.e., 5%, 30%. For the FPR and HPR labels, we do not include the percentage of mobility since this parameter is orthogonal to the solutions.

Figure 4 shows how the variability of the number of clients from 5 to 100 impacts the performance of our FPR, PPR, and HPR solutions. We have considered 5% and 30% of UEs mobility, while maintaining the number of OVSs and servers to 25 and 10, respectively. In Fig. 4(a) the Y-axis shows the number of OVSs used, while in Fig. 4(b) the Y-axis shows the execution time in seconds for computing the aforementioned solutions. The results obtained in Fig. 4(a) show that for FPR and PPR, the number of activated OVSs increases from 5 to 10 when we vary the number of clients from 5 to 100, mainly stabilizing after 40 clients with FPR outperforming PPR, as expected, while HPR at steady state using 64.2% more OVSs then FPR. The mean number of activated OVSs for FPR is 8.618 with a standard deviation of 1.083. Concerning the PPR solution, the mean number of used OVSs is 8.794 with a standard deviation of 1.084 when having 5% user mobility and 9.024 with a standard deviation of 1.149 for 30% user mobility. Looking only at the steady state part of our simulation, the mean number of OVSs activated for HPR is 14.151 with a standard deviation of 1.689.

Furthermore, the results for the computational time summarized in Fig. 4(b), show that as the number of clients increases, the computational cost increases linearly for our solutions. The linear regression parameters for the mean computational time of the FPR solution were 0.01 and 1.48, as α and β, respectively. For PPR, when evaluating a scenario with 5% of mobility α and β equal to 0.022 and -0.484, respectively, while α and β increase to be 0.842 and 0.63 for mobility of 30%. Moreover, our results also show that HPR is less impacted by the number of clients than FPR and PPR, with an α and β equal to 0.010 and -0.30, respectively. We can also observe that the number of clients has no impact on HPR in terms of execution time when the number of clients is lower than 40.The results, presented in Figure 4, allow to conclude that FPR provides slightly better performance in terms of the number of activated OVSs compared to the PPR and HPR solutions, irrespective of the number of OVSs and the mobility of clients. Also, we have noticed that the PPR solution perform better than HPR one considering the activated OVSs. However, HPR outperforms both FPR and PPR solutions in terms of execution time.

Figure 5 shows the results of our evaluation for FPR, PPR, and HPR when we vary the number of servers from 1 to 10 while fixing the number of clients and OVSs to 25. Fig. 5(a) shows that as we increase the number of servers in our network the number of activated OVSs increases linearly, albeit at a very low rate, with the number of servers, which confirms the previously presented results. It also shows us that as in the previous results the number of activated OVSs for HPR is significantly higher than FPR and PPR. Fig. 5(b) shows that while the mean computational time to find a solution increases linearly with the number of servers, with the linear regression of these samples returned 0.132 and 0.362, at α and β, respectively. The mean computational time for both PPR and HPR remained almost constant, with PPR showing its sensibility to user mobility and HPR been almost immune to it.

Figure 6 presents the performance evaluations of FPR, PPR, and HPR solutions when varying the number of OVSs while the number of servers and clients remain fixed at 10 and 25, respectively. Fig. 6(a) shows that increasing the number of available OVSs in the network has a positive impact on both solutions FPR and PPR and negative impact on HPR one. This effect can be explained as follows: increasing the number



(a) The cost of proposed solution

(b) The run-time of proposed solutions

Figure 6: Impact of number of OVSs on different solutions with variant mobility patterns

of OVSs leads to an increase of the possibility for finding more optimal paths for both FPR and PPR algorithms. This has a negative impact on the HPR solution as it increases the number of hops between users and their corresponding servers and likewise the probability for getting in a local minimum. Moreover, it is shown that FPR has slightly better performance than PPR for 5% and 30% of users' mobility. Fig. 6(b) indicates that PPR and HPR were able to reduce the rate of exponential cost to find a viable configuration for our network as we increase its density. The results for PPR-30% and PPR-5% show that the mobility patterns have a lower impact on the computational time than the density of OVSs in the network. This behavior is caused by the reuse of existing paths instead of re-computing (with a mean time of $1.686s$ for PPR-30% and $1.622s$ for PPR-5%). Meanwhile, the mean computational cost for HPR was $0.064s$.

Table II shows the mean computational cost and its standard deviation to update the mobile network graph based on our filter procedure and the user mobility patterns for our PPR solution. For the three scenarios, the computational cost behavior was close to a constant for both percentiles of mobility evaluated. This shows that the cost has a strong correlation with the number of clients and OVSs than the number of servers. This behavior is the result of the number of flows that is directly proportional only to the number of clients in the network.

Table II: Filter Algorithm Times

Figure	Mobility (%)	Mean (s)	St. Deviation (ms)
Figure 4(a)	5% of client mobility	0.843	0.629
	30% of client mobility	1.018	0.744
Figure 5(a)	5% of client mbility	0.252	0.015
	30% of client mbility	0.307	0.012
Figure 6(a)	5% of client mbility	1.622	1.675
	30% of client mbility	1.686	1.629

Furthermore, the solutions presented in this paper and theretofore evaluated in this section were designed to be periodically executed or upon a significant network load alternation. We execute the proposed solutions in the background without affecting routing protocols or impacting the data plane's QoS, which remains consistent. The experimental evaluation shows that under these assumptions, our proposed solutions, compute valid routing configurations where the computational cost increases with the number of OVSs in the network. Analyzing the computational cost results from figures 4(b) to 6(b), we can observe that both FPR and PPR have a higher sensitivity to the number of elements in the network, i.e. OVS for FPR and number of end-hosts for both PPR and FPR, while HPR is only sensitive to end-hosts mobility. Meanwhile, as all three solutions presented in this paper adhere to the optimization model in Section III, it is ensured that the bandwidth requested by each flow or user is fulfilled, and hence the desired QoS. Moreover, the results presented in Figures 4(a) to 6(a) show that both PPR and HPR have a higher network resource consumption, i.e. mean number of OVSs, than FPR. Also, from this figure, we observe that the HPR solution has the worst performance in terms of network resource consumption. In our model, the number of OVSs for a given network has a direct relationship with the amount of bandwidth and path availability. Hence, any technique that provides network configurations, which use more OVS than the minimum necessary (i.e., HPR), supports a smaller number of end hosts traffic than the one that minimizes the number of OVSs (i.e., FPR and PPR).

VII. Conclusion

This paper extends our previous study on multiple path forwarding introducing two new solutions for an SDN-enabled network: i) PPR that determines the routing paths only for newly arrived or handover users while maintaining constant the remaining routes related to stationary users and ii) HPR a heuristic that explores Dijkstra shortest-path algorithm for

allocating different paths between UEs and servers. Furthermore, we compare these solutions with our previously published FPR [1] solution that re-establishes the routing paths for all users towards the desired servers from scratch obtaining an optimal network configuration. We evaluated the proposed solutions based on OPEX costs, which is measured in this paper as the number of activated OVSs in viable network configuration, and computation time considering QoS guarantees without over-committing the network resources to end-users. Our analysis and results showed that the PPR solution can determine a network configuration that offers QoS guarantees while keeping the number of activated OVSs close to the optimum result obtained with FPR. However, even with a lower computational cost, albeit still too high for some intransigent use cases. To address these limitations, we have also developed and evaluated HPR, a solution that is able to keep the computational cost almost constant albeit at a cost of a higher number of activated OVSs. Finally, our evaluations allow us to conclude that FPR, PPR, and HPR are sensitive to the OVSs density. This increases the computational time to find a viable configuration for FPR and PPR, while HPR can find a viable configuration faster than both of them, albeit using a higher number of activated OVSs. These results highlight the trade-off between computational cost and the number of OVSs used to find a feasible configuration in the proposed model.

ACKNOWLEDGMENT

This work was partially supported by the European Union's Horizon 2020 Research and Innovation Program through the MonB5G Project under Grant No. 871780, by the Academy of Finland 6Genesis project under Grant No. 318927, and by the Academy of Finland CSN project under Grant No. 311654.

REFERENCES

[1] D. L. C. Dutra, M. Bagaa, T. Taleb, and K. Samdanis, "Ensuring end-to-end qos based on multi-paths routing using SDN technology," in *IEEE Global Communications Conference (GLOBECOM)*, Singapore, Singapore, Dec 2017.

[2] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini and H. Flinck, "Network Slicing & Softwarization: A Survey on Principles, Enabling Technologies & Solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429 – 2453, 2018.

[3] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Architecture & Orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, May 2017.

[4] T. Taleb, M. Corici, C. Parada, A. Jamakovic, S. Ruffino, G. Karagiannis, and T. Magedanz, "EASE: EPC as a service to ease mobile core network deployment over cloud," *IEEE Network*, vol. 29, no. 2, pp. 78–88, March 2015.

[5] T. Taleb, "Toward carrier cloud: Potential, challenges, and solutions," *IEEE Wireless Communications*, vol. 21, no. 3, pp. 80–91, June 2014.

[6] T. Taleb, B. Mada, M. I. Corici, A. Nakao, and H. Flinck, "PERMIT: Network Slicing for Personalized 5G Mobile Telecommunications," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 88–93, May 2017.

[7] NGMN Alliance, "5G White Paper," Tech. Rep., February 2015. [Online]. Available: {https://www.ngmn.org/uploads/media/NGMN_5G_White_Paper_V1_0.pdf}

[8] 3GPP TS 28.530, "Management and Orchestration; Concepts, Use cases and Requirements," vol. v16.0.0, Sep. 2019.

[9] R. Rokui, et.al, "5G Transport Slice Connectivity Interface," *IETF Internet-Draft*, Jul. 2019.

[10] Q. Wu, S. Litkowski, L. Tomotaki, K. Ogaki, "YANG Data Model for L3VPN Service Delivery," *IETF RFC 8299*, Jan. 2018.

[11] B. Wen , G. Fioccola, C. Xie, L. Jalil,, "A YANG Data Model for L2VPN Service Delivery," *IETF RCF 8466*, Oct. 2018.

[12] ONF, "https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR-521$_SDN_Architecture_issue_1.1.pdf SDN Architect

[13] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[14] A. Ksentini, M. Bagaa, T. Taleb, and I. Balasingham, "On using bargaining game for Optimal Placement of SDN controllers," in *2016 IEEE International Conference on Communications (ICC)*, Kuala Lumpur, Malaysia, May 2016.

[15] A. Ksentini, M. Bagaa, and T. Taleb, "On Using SDN in 5G: The Controller Placement Problem," in *IEEE Global Communications Conference (GLOBECOM)*, Washington, DC, USA, Dec 2016.

[16] R. A. Addad, D. L. C. Dutra, T. Taleb, M. Bagaa, and H. Flinck, "mira!: An sdn-based framework for cross-domain fast migration of ultra-low latency 5g services," in *2018 IEEE Global Communications Conference (GLOBECOM)*.

[17] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[18] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "ONOS: Towards an Open, Distributed SDN OS," in *ACM HotSDN*, August 22 2014.

[19] R. A. Addad, D. L. C. Dutra, M. Bagaa, T. Taleb, H. Flinck, and M. Namane, "Benchmarking the ONOS Intent Interfaces to Ease 5G Service Management," in *2018 IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, United Arab Emirates, Dec 2018, pp. 1–6.

[20] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," in *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, 2014, pp. 1617 – 1634.

[21] B. Sonkoly, A. Gulyás, F. Németh, J. Czentye, K. Kurucz, B. Novák, and G. Vaszkun, "On QoS Support to Ofelia and OpenFlow," in *European Workshop on Software Defined Networking*, Darmstadt, Germany, Oct 2012.

[22] H. E. Egilmez, S. T. Dane, K. T. Bagci, and A. M. Tekalp, "OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks," in *Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference*, Hollywood, CA, USA, Dec 2012.

[23] H. E. Egilmez and A. M. Tekalp, "Distributed QoS Architectures for Multimedia Streaming Over Software Defined Networks," *IEEE Transactions on Multimedia*, vol. 16, no. 6, pp. 1597–1609, Oct 2014.

[24] S. Sharma, D. Staessens, D. Colle, D. Palma, J. Gonçalves, R. Figueiredo, D. Morris, M. Pickavet, and P. Demeester, "Implementing Quality of Service for the Software Defined Networking Enabled Future Internet," in *3rd European Workshop on Software Defined Networks*, London, UK, Sept 2014.

[25] T. Wang, F. Liu, J. Guo, and H. Xu, "Dynamic SDN controller assignment in data center networks: Stable matching with transfers," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, San Francisco, CA, USA, April 2016.

[26] T. Wang, F. Liu, and H. Xu, "An Efficient Online Algorithm for Dynamic SDN Controller Assignment in Data Center Networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2788–2801, Oct 2017.

[27] S. Tomovic, N. Prasad, and I. Radusinovic, "SDN control framework for QoS provisioning," in *22nd Telecommunications Forum Telfor (TELFOR)*, Belgrade, Serbia, Nov 2014.

[28] C. Hopps, "(analysis of an equal-cost multi-path algorithm," in *IETF RFC 2992*.

[29] M. R. Celenlioglu and H. A. Mantar, "An SDN Based Intra-Domain Routing and Resource Management Model," in *IEEE International Conference on Cloud Engineering*, March 2015, pp. 347–352.

[30] J. Yan, H. Zhang, Q. Shuai, B. Liu, and X. Guo, "HiQoS: An SDN-based multipath QoS solution," *China Communications*, vol. 12, no. 5, pp. 123–133, May 2015.

[31] S. Sahhaf, W. Tavernier, D. Colle, and M. Pickavet, "Adaptive and reliable multipath provisioning for media transfer in SDN-based overlay networks," in *Computer Communications*, vol. 106, July 2017, pp. 107–116.

[32] S. A. Hussain, S. Akbar, and I. Raza, "A dynamic multipath scheduling protocol (DMSP) for full performance isolation of links in software

defined networking (SDN)," in *2nd Workshop on Recent Trends in Telecommunications Research (RTTR)*, Palmerston North, New Zealand, Feb 2017.
[33] A. Basit, S. B. Qaisar, H. R. Syed, and M. Ali, "SDN Orchestration for Next Generation Inter-Networking: A Multipath Forwarding Approach," *IEEE Access*, March 2017.
[34] C. Huang, C. Nakasan, K. Ichikawa, and H. Iida, "A Multipath Controller for Accelerating GridFTP Transfer over SDN," in *IEEE 11th International Conference on e-Science*, Munich, Germany, Aug 2015.
[35] ——, "An SDN-Based Multipath GridFTP for High-Speed Data Transfer," in *IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, Nara, Japan, June 2016.
[36] M-J. Fu and F. Wu, "Investigation of Multipath Routing Algorithms in Software Defined Networking," in *IEEE International Conference on Green Informatics (ICGI)*, August 15-17 2017.
[37] T. A. T. S. L. Guillen, S. Izumi and H. Muraoka, "SDN-based Hybrid Server and Link Load Balancing in Multipath Distributed Storage Systems," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, April 23-27 2018.
[38] Y. Guan, W. Lei, W. Zhang, S. Liu, and H. Li, "Scalable orchestration of software defined service overlay network for multipath transmission," in *Computer Networks*, vol. 137, June 2018, pp. 132–146.
[39] S. Dwarakanathan, L. Bass, and L. Zhu, "Cloud Application HA Using SDN to Ensure QoS," in *IEEE 8th International Conference on Cloud Computing*, New York, NY, USA, June 2015.
[40] M. S. Yoon and A. E. Kamal, "Power Minimization in Fat-Tree SDN Datacenter Operation," in *IEEE Global Communications Conference (GLOBECOM)*, San Diego, CA, USA, Dec 2015.
[41] S. Tariq and M. Bassiouni, "QAMO-SDN: QoS aware Multipath TCP for software defined optical networks," in *12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, Las Vegas, NV, USA, Jan 2015.
[42] Q. Wang, G. Shoi, Y. Liu, Y. Hu, Z. Guo and W. Chang, "Implementation of Multipath Network Virtualization With SDN and NFV," *IEEE Access*, vol. 6, pp. 32 460 – 32 470, May 2018.
[43] Jeroslow, R.G, "Trivial integer programs unsolvable by branch-and-bound," *Springer-Verlag Mathematical Programming*, vol. 6, no. 1, pp. 105 – 109, 1974.

Dr. Miloud Bagaa received the Engineer's, master's,and Ph.D. degrees from the University of Science and Technology Houari Boumediene, Algiers,Algeria, in 2005, 2008, and 2014, respectively.From 2009 to 2015, he was a Researcher with the Research Center on Scientific and Technical Information, Algiers. From 2015 to 2016, he was with the Norwegian University of Science and Technology, Trondheim, Norway. He is currently a Senior Researcher with Aalto University. His research interests include wireless sensor networks, Internet of Things, 5G wireless communication, security, and networking modeling. From2015 to 2016, he received the Post-Doctoral Fellowship from the European Research Consortium for Informatics and Mathematics.

Dr. Diego L. C. Dutra is a professor at Federal University of Rio de Janeiro (UFRJ), Brazil, where he is also a member of the COMPASS Laboratory. He received a B.Sc. in Computer Science from UFF/Brazil, his M.Sc. and D.Sc. degrees in Systems Engineering and Computer Science Program from Federal University of Rio de Janeiro, Brazil, in 2007 and 2015, respectively. He has worked as a postdoctoral researcher in the COMPASS/UFRJ and MOSA!C Lab/Aalto, from 2015 to 2016 and 2016 to 2017, respectively. His research interests include computer architecture, high-performance computing, virtualization, cloud computing, wireless networking, mobile system, and Software-Defined Systems.

Prof. Tarik Taleb received the B.E. degree (Hons.) in information engineering and the M.Sc. and Ph.D. degrees in information sciences from GSIS, Tohoku University, Sendai, Japan, in 2001, 2003, and 2005, respectively. He is currently a Professor with the School of Electrical Engineering, Aalto University, Espoo, Finland. He is a member of the IEEE Communications Society Standardization Program Development Board. In an attempt to bridgethe gap between academia and industry, he founded the IEEE-Workshop on Telecommunications Standards: From Research to Standards, a successful event that was recognized with the Best Workshop Award by the IEEE Communication Society (Com-SoC). Based on the success of this workshop, he has also founded and has been the Steering Committee Chair of the IEEE Conference on Standards for Communications and Networking. He is the General Chair of the 2019 edition of the IEEE Wireless Communications and Networking Conference to beheld in Marrakech, Morocco. He is/was on the Editorial Board of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the IEEE Wireless Communications Magazine, the IEEE JOURNAL ONINTERNET OFTHINGS,the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE COMMUNICATIONS SURVEYS & TUTORIALS, and a number of Wiley Journals. He is an IEEE Communications Society (ComSoc) Distinguished Lecturer.

Dr. Konstantinos Samdanis (konstantinos.samdanis@nokia-bell-labs.com) received the M.Sc. and Ph.D. degrees from King's College London in 2003 and 2009 respectively. He worked for NEC Europe, Heidelberg, between 2009 to 2016 as a Senior Researcher and a Broadband Standardization Specialist, involved in numerous EU projects, including 5G-NORMA, iJOIN, BeFemto, and standardization activities in BBF, focusing on Mobile Backhaul and 3GPP SA5 in the area of Self-Organized Networks. From 2016 to 2018 he moved to Huawei Technologies, Munich taking the role of Principal Researcher for 5G carrier networks, where he was involved in strategy and research for 5G architectures and transport networks. His main actives involved the specification of the Mobile-Transport API for network slicing in BBF and 3GPP SA5, while he was also involved as a delegate at IETF in the Network and Routing Area WG focusing on SR and VPN+. Since 2019 he is a Research Project Manager at Nokia Bell Labs, Munich involved in standardization activities on 5G core and network management concentrating on network analytics and AI/ML, while also acting as a delegate in 3GPP SA5 and SA6. Konstantinos served as an Editor on the Network Slicing feature topic at the IEEE Communications Magazine in 2017 and as Guest Editor for the IEEE JSAC Series on Network Softwarization and Enablers. He has arranged and authored a book in Green Communications with Wiley and is the author of over 80 academic publications and 30 patent applications.