Evaluating the Impact of Delay Constraints in Network Services for Intelligent Network Slicing based on SKM Model

Ahmed El-mekkawi, Xavier Hesselbach, and Jose Ramon Piney

Abstract: **Solving the problem of network resource allocation with delay constraint is a significant challenge for realizing future Internet and 5G networks services such as advanced mobile broadband services and Internet of things (IoT), especially under the network slicing scenario. The impact of delay constraints may lead to rejection of demands, resulting in low resource utilization of network resources. This is especially severe when dynamic traffic is considered. Therefore, intelligent resource allocation algorithms are required to use the network resources in delay constrained scenario efficiently. Moreover, these algorithms should guarantee quality of service (QoS) between different priority slices during congestion case. Therefore, in this paper, we analyze the impact of delay constraint on the performance of an online resource allocation algorithm based on an intelligent efficient squatting and kicking model (SKM), proved in other works to be the most effective up to the present time yet. SKM incorporates kicking and squatting of resources as innovative techniques enabling it to achieve 100% resource utilization and acceptance ratio for higher priority slices in scenarios where the other state of art algorithms not able to reach by far in some scenarios. Simulation results showed that incorporating delay constraints has a significant impact on the performance, resulting in up to 10% and 4% reduction in terms of average resource utilization and acceptance ratios respectively.**

Index Terms: **5G, delay, network slicing, resource allocation, SKM, quality of service.**

I. INTRODUCTION

5G networks and beyond are expected to significantly offer new capabilities in order to satisfy the stringent requirements of the services that they are envisaged to serve. Network slicing has emerged as a promising paradigm through which the divergent requirements of future services will be met. In this regard, the 3rd generation partnership project (3GPP) has identified enhanced mobile broadband (eMBB), massive Internet of things (MIoT), ultra-reliable and low-latency communication (URLLC) and vehicle-to-everything (V2X) as the four critical usage scenarios in 5G communication systems [1], [2].

However, the above scenario presents two major challenges in a number of dimensions: First, the network resources are limited and exhaustible which posses challenges regarding how to efficiently allocate these resources to the different slices while meeting the divergent service requirements such as delay and throughput; secondly, the different service slices are characterized by different priorities and criticality, which causes complexities regarding real-time end-to-end (E2E) quality of service (QoS) routing of the different services while managing the prioritization levels across the different slices. This is more critical under network resource scarcity especially under disaster events and network congestion. In this context, the network is not able yet to take intelligent decisions in order to optimize the behaviour [3].

Consequently, this scenario necessitate development of intelligent resource allocation algorithms with joint capability to: i) Maximize the utilization of network resources while achieving possible maximum productivity and facilitation of sharing resources among slices while allowing a specific slice to meet the the demanded service level agreement (SLA); ii) guarantee prioritization of critical services especially under congested scenarios; and iii) satisfy all the constraints related to the request especially end-to-end delay [4]–[6].

As a contribution to the above challenges, in [7]–[9], an efficient algorithm based on squatting and kicking techniques has been introduced. The squatting technique provides for sharing of unused resources between higher and lower priority service slices while kicking technique ensures proper QoS for higher traffic priority slices by expelling lower priority slices from resources directly assigned to them. The results from the simulations revealed that the introduced algorithm was optimal in terms of resource allocation and QoS for high priority users and admission control while improving the total resource utilization. However, the work proposed in [7]–[9] was analysed based on scenarios that are not representative of a realistic 5G network environment by either considering single link network topology [7], [8] or assuming requests arrive offline [9]. However, in a realistic 5G scenario, the network topology is complex, the transmission is real-time, the requests arrive in online mode, and the services are delay constrained. The online arrival of requests makes it imperative to keep the status of the substrate network resources always up-to-date, in order to directly assess the probabilities of allocating other requests as they arrive. With this motivation, this paper extends the above work by developing an intelligent algorithm that uses the intelligence of SKM strategy for efficient deployment and allocation of network resources in a multi-slice scenario. We formally define the proposed algorithm to solve the problem of real-time resource

Manuscript received December 2, 2021; revised July 18, 2021; approved for publication by Hyoil Kim, Division III Editor, July 19, 2021.

This work has been partially supported by the Ministerio de Economy Competitividad of the Spanish Government under project PID2019-108713RB-C51.

The authors are with the Universitat Politècnica de Catalunya, Department of Network Engineering, Jordi Girona 1-3, C3 Campus Nord, Barcelona, 08034, Spain. email: ahmed.mohamed.abdelaty.elmekaw, Xavier.Hesselbach @upc.edu, jpiney @entel.upc.edu

A. El-mekkawi is the corresponding author. Digital

Object Identifier: 10.23919/JCN.2021.000024

allocation for QoS end-to-end routing considering realistic network behaviour by incorporating delay constraints and considering full network topology under online request arrival. In addition, this work extensively analyses the impact of delay on the performance of the proposed algorithm. Moreover, although network slicing is envisaged to be implemented in an end-to-end fashion across links and nodes, the resources of the virtualized node functions can be scaled up with more ease compared to the link resources. Therefore, the link resources form the performance bottleneck of the network especially under bandwidth intensive applications. Cognizant of this fact, this paper focuses on how intelligence can be deployed in network function virtualization (NFV) in order to provide efficient utilization of link bandwidth resources in a multi-slice scenario considering strong constraints as required in 5G networks. NFV technology accelerates the process of user-oriented services execution appearing in both cost and time saving by allowing execution and deployment of middleboxes as virtual network functions (VNFs) running on virtual machines (VMs). Furthermore, NFV facilitates service deployment by employing the concept of service chaining (SC) [10]. Specifically, in practical scenarios, the computational complexity to find the best paths from a source node to a destination node is huge, so the proposed algorithm is suggested as a suitable candidate to be applied inside NFV under shape of SC that provides the huge computational capacity required by the network to make intelligent decisions regarding admission control, routing path computation and resource allocation. In line with bandwidth resource management in a multi-slice scenario, bandwidth allocation models (BAMs) provide enhanced metrics compared to best-effort models. SKM outperforms the others by far especially, during congested scenarios, as shown in [7], and so this is the algorithm considered in this work.

In summary, our contributions are the following:

1. An intelligent service deployment algorithm that uses SKM strategy to jointly maximize resource utilization, acceptance rate and ensure QoS for higher priority slices while meeting various service constraints in a multi-slice scenario.
2. In-depth analysis of the impact of delay constraints on the performance of the proposed online algorithm, which represents the direct applicability of network slices on future 5G networks and beyond.
3. A formal definition of the proposed algorithm is introduced considering a real-time application for full network topology with delay constraints request.
4. Performance evaluations and analyses of the proposed algorithm are presented against routing algorithms incorporating BAM strategies in terms of several metrics reflecting the ability to manage multi-slice requests in a resource-limited 5G network reflecting the ability to accommodate various input traffic loads as well as the lifetime of requests.

A practical evaluation scenario:

In real 5G networks, the services, applications, and users need to interact with the infrastructure network directly and in real-time [11]. Hence, in the context of network slicing, the requests must be analyzed and assigned on the physical substrate network, online, using the shared resources effectively, adhering to the necessary service qualities as required. Simultaneously, it is imperative to keep the status of the substrate network resources always up-to-date, in order to directly assess the probabilities of allocating other requests as they arrive. To this end, the proposed deployment algorithm of this paper is planned for an online scenario, and its main aim is to successfully allocate the requests among different priority slices, on real-time, while maximizing the total resource consumption in the entire substrate network considering E2E delay as the primary allocating constraint. The algorithm manages the network demands sequentially and continues observing and updating the substrate network, to allow more resources to be utilized in the future by new demands.

The remainder of the paper is organized as follows: Section II presents the related work. Section III will introduce the system models and online problem formulation. Section IV presents a description of the proposed online algorithm, and analysis for its evaluation results will be introduced in Section V. Finally, Section V Concludes the paper and proposes suggestions for future work.

II. RELATED WORK

QoS management across the large-scale multi-slice network is a broad topic with existing works focusing on: (i) Development of a new bandwidth and QoS algorithm that relies on optimizing resource allocation and (ii) defining a new method for multi-path routing that takes into account QoS constraints.

To develop new bandwidth allocation algorithms for realizing E2E network slices, BAMs such as maximum allocation model (MAM) [12], the Russian doll model (RDM) [13] and the AllocTC-Sharing (AllocTC) model [14] are of high importance in the context of efficient and customized use of network resources among different traffic classes (slices). The primary issue of the models dependent on RDM is that the resources reservation is done from the bottom to top, which implies that lower priority classes share their resources with higher priority classes and not the other way round. Likewise, the fundamental issues of the models dependent on MAM are that any class can not utilize the available resources from another given class. To defeat MAM and RDM implementation problems, various works have been done proposing new dynamic bandwidth sharing algorithms dependent on changed MAM or RDM methodologies, for example, [15]–[20]. Although, these models can not guarantee high acceptance for higher priority classes and give 100% network utilization simultaneously.

Optimum utilization can be accomplished by reserving resources either from above or from below. In this respects, the authors in [14] presented a model called AllocTC which allows low priority requests to share the unused resources from high priority requests and the other way around. The authors in [21] present another technique with a blend of (MAM, RDM, G-RDM, and AllocTC) models dependent on a controller by utilizing various measurements to change from one model to another to improve implementation in terms of link use, blocking probability, and packet number. In [22], the authors presented a novel software-defined network (SDN)-based design following a new smart and dynamic model (Smart Alloc) for assignment and dealing with the QoS and routing with QoS requirements for a multi-protocol label switching traffic engineering diffserv aware

(MPLS DS-TE) network. This model is performed dependent on RDM and AllocTC approaches and points, initially, to order flows dependent on their threshold severity (high, medium, and low). Regardless of the priority of the high flow threshold, the last can profit by the borrows of the other categories. Also, to gather bandwidth from other categories and to determine the fairness index to allocate resources efficiently to all flows considering their priorities. This model was executed on a controller to handle QoS and to route for only the MPLS DS-TE networks. Notwithstanding, every one of these models can not guarantee high admission for higher priority slices.

To ensure paths with enough bandwidth considering large scale network, the authors in [23] presented a path enhancement approach that incorporates two principal parts: request sorting and path assignment. The authors compute the priority of all requests as its bandwidth demand divided by its hop. First, the algorithm updates the weight of each link based on bandwidth usage. Then, requests are arranged in descending order of priorities. For each request, the least weighted path is chosen.

Regarding bandwidth and delay constraints routing, the authors of [24] proposed a minimum delay model which eliminates all links with deficient available bandwidth along the required paths and afterwards decides the ideal path which has a minimal delay in the network. In [25], the authors proposed the delay weighted maximum capacity routing algorithm which determine the delay-related shortest paths for all source and destination node pairs. By implementing the Dijkstra [26] algorithm, the optimal path is one that meets the delay and bandwidth limitations with the least weight. The authors in [27] proposed a Yen's algorithm for each source and destination node to determine the candidate path set, including k-loop-less paths in increasing order of delay. Besides, the weight function indicated in the links is adjusted accordingly.

The path selection strategy in our work differs from those adopted in the above works by considering the residual bandwidth resources in a multi-slice scenario while considering E2E delay constraints.

III. SYSTEM MODEL AND PROBLEM FORMULATION OF THE ONLINE PROBLEM

Since the resource allocation problem deals with making decisions about an efficient utilization under limited physical substrate network resources, the resource allocation problem was traditionally modelled as an optimization problem of an objective function, and constrained by controlling conditions, matching the availability of the resources against the requirements, while using the limited physical resources.

In this work, the problem is formulated with the link and slice constraints, subject to the priority service demands, network capacity and link resources (e.g., bandwidth). The inputs to the resource allocation phase are slice traffic, network capacity and substrate link resources. The output is the optimal deployment path for simultaneous slices demands that maximizes the network resources utilization while ensuring high admission for higher priority slices based on SKM. In this regard, the optimization of usage has two considerations. Network capacity is the primary consideration for describing the maximum number of resources that can be provided for forwarding traffic, which also plays a critical role in network load-balancing. Besides, the second consideration, we also take into account the deployment propagation distance of forwarded traffic in terms of the substrate link resources.

Note that in this study, to facilitate the assessment of the proposed deployment algorithm, we applied only three types of slices discussed below, and also assume that service demand is acceptable when link resources are available along the required routing path from the source node to the destination node. The slices considered in this work are: 1) eMBB slice: This kind of slice is not strict with specific QoS requirements. Therefore, we assumed that this slice meets the lower priority service demands; 2) MIoT slice: This type of slice is characterized by a very large number of connected devices typically transmitting a relatively low volume of non-delay sensitive data. We assumed that this slice satisfies the intermediate priority service demands and 3) uRLLC slice: This type of slice is strict with the delay requirements. We assumed that this slice satisfies the highest priority service demands. Please note that in this work we only focused on the problem of dynamic and efficient allocation of bandwidth resources considering E2E delay to the services with different priorities and thereby improving the service quality in congested and extreme scenarios such as emergency and disasters. Moreover, we would like to mention that other parameters such as reliability can be incorporated into our proposed algorithm which is considered future work.

Table 1 summarizes all description of parameters, decision variables and main metrics used in this article.

A. Infrastructure Network Model

We model the provided physical substrate network as a directed graph $\mathcal{G}(\mathcal{X}, \mathcal{L})$ where $\mathcal{X}$ and $\mathcal{L}$ indicate the set of all substrate nodes and substrate links respectively. Whenever such a connection exists, we indicate by $l \in \mathcal{L}$ as a single substrate link between substrate node $i \in \mathcal{X}$ and substrate node $j \in \mathcal{X}$. Each substrate link l is described by i) maximum link resources capacity $\mathcal{R}(l)$; ii) available link resources at a given time denoted by $\mathcal{R}_a^t(l)$; iii) consumed link resources $\mathcal{R}_z^t(l)$ at time t; iv) a set of traffic slices assigned along the link are denoted by $(\mathcal{CT}s)(l)$, where $\mathcal{CT}_{\mathcal{N}}(l)$ is the highest priority slice and $\mathcal{CT}_{\infty}(l)$ is the lowest priority slice; v) actually allocated resources to slice c $\mathcal{S}_c(l)$, where $c \in [1, \mathcal{N}]$; vi) slice resource constraints $\mathcal{RC}_c(l)$; vii) propagation delay $\delta(l)$. Whenever such a path exists, we denote by $\mathcal{P}_{s,r}^n(l)$ as a possible physical path between substrate node (source node) $s \in \mathcal{X}$ and substrate node (destination node) $r \in \mathcal{X}$, the nth path between substrate node s and substrate node r for all $n \in [1, \mathcal{P}_{\text{set}}^{s,r}(l)]$.

B. Slice Demand Model

Each demand belonging to any type of slice to be served in the network is defined by i) A source node $s \in \mathcal{X}$; ii) a destination node $r \in \mathcal{X}$; iii) the amount of resources required belonging to slice c $d_w(\mathcal{CT}_c)$, where demand $w \in [1, \mathcal{D}]$; iv) priority $c_{d_w} \in [1, \mathcal{N}]$; VI) maximum acceptable E2E delay $\delta_{s,r}(l)$ and v) lifetime interval t_{d_w}. Besides, we consider in this work that the size of the demand can be translated into a demanded number of link resources, so candidate paths from source to desti-

Table 1. Notation and variables.

Notation	Description
$\mathcal{G}$	Directed graph of the physical network.
$\mathcal{X}$	Substrate network nodes.
$\mathcal{L}$	Substrate network links.
l	Single hop edge from substrate node i to j.
$\mathcal{RC}_c(l)$	Resource constraints for slice c also equal to maximum reservable resources for slice c in l.
$(\mathcal{CT}_c)(l)$	Priority slice c in l.
$\mathcal{R}(l)$	maximum reservable resources for all slices together and is equal to link capacity.
$d_w(\mathcal{CT}_c)$	The amount of resources (size) of demand w belonging to slice c.
t	time variable.
$\mathcal{R}_a^t(l)$	Available resource capacity on l at time t.
$\mathcal{R}_z^t(l)$	The consumed resources capacity on l at time t.
$\mathcal{X}_w^{t,l}$	is the binary variable equal to 1 if the demand $w \in \mathcal{W}$ is assigned resources at link $l \in \mathcal{L}$, zero otherwise.
$\mathcal{P}_{s,r}^n(l)$	Single routing path from s to r for the demand.
$\mathcal{P}_{\text{set}}^{s,r}(l)$	All possible paths from source s to destination r in the network.
$\delta(l)$	Propagation delay
$\delta_{s,r}(l)$	maximum acceptable E2E delay.
t_{d_w}	The duration of demand w.
$\mathcal{T}$	Duration of the simulation window in time units.
$\mathcal{D}_c$	Total number of demands by slice c.
$\mathcal{D}$	Total number of demands by all slices.
$\mathcal{S}_c(l)$	The actually allocated resources to slice c on l.
c_{d_w}	Priority of demand w.
$\mathcal{BD}$	Number of blocked demands by all slices.
$\mathcal{BD}_c$	Number of blocked demands by slice c.
$\mathcal{AD}$	Number of accepted demands by all slices.
$\mathcal{AD}_\rfloor$	Number of accepted demands by slice c.
μ	The mean value of the utilization of all links across the network.
$\mathcal{P}_{\mathcal{LTH}}$	The number of preemption of higher priority traffic by lower priority traffic
$\mathcal{P}_{\mathcal{HTL}}$	The number of preemption of lower priority traffic by higher priority traffic
$\mathcal{P}_{re}$	Number of preempted demands in the whole network.
$\mathcal{SH}_q(l)$	Squatted resources from higher priority $slice_q$ on l. Where $q \in [1, \mathcal{N}]$
$\mathcal{SL}_q(l)$	Squatted resources from lower priority $slice_q$ on l.
$\mathcal{K}_q(l)$	Kicked resources from lower priority $slice_q$ on l.
$\mathcal{Z}(d_w)$	1 if demand w is successfully mapped.

nation are calculated when the demand has reached. Moreover, in this work the priorities are enforced at the slice class level. In other words, different slice classes have different priorities. Therefore, in the provided formulation, the priorities are indirectly incorporated in terms of a user of a given class's right of access to the network resources by way of its ability to kick users of other classes, and the maximum amount of the total network resources the demands of a given class can consume under network congestion.

C. Formulation of the Online Objective Function

The principal purpose of SKM is to successfully accommodate all arriving demands on the online scenario, while maximizing overall resource utilization in the whole substrate network, by effectively allocating available resources for service demand in physical network paths. Demands in the online mode arrive with duration time t_{d_w}, therefore, the objective function must consider allocating demands during the time intervals specified by each related demand. Thus, the main objective can be expressed as:

$$\max \left\{ \mathcal{U}(\mathcal{T}) = \frac{1}{\mathcal{T}} \sum_{t \in \mathcal{T}} \frac{1}{\mathcal{L}} \sum_{\forall l \in \mathcal{L}} \sum_{w \in \mathcal{W}} \frac{\mathcal{X}_w^{t,l} * d_w(\mathcal{CT}_c)}{\mathcal{R}(l)} \right\} \quad (1)$$

As shown in (1), the resource allocation problem is aimed at maximizing the utilization ($\mathcal{U}(\mathcal{T})$). $\mathcal{U}(\mathcal{T})$ is the utilization of the links across the network in each time window $\mathcal{T}$ [28]. The link resource utilization relates to the ratio of the used link resources to the link capacity averaged over all substrate links. $\mathcal{X}_w^{t,l} \in [1, 0]$ is binary variable equal to 1 if the demand $w \in \mathcal{W}$ is assigned resources at link $l \in \mathcal{L}$, zero otherwise. $d_w(\mathcal{CT}_c)$ denotes the demanded bandwidth resources by demand w. $\mathcal{T}$ denotes the duration of the simulation window in time units.

The total consumed resources at edge $l \in \mathcal{L}$ at any unit time t given by:

$$\mathcal{R}_z^t(l) = \sum_{w \in \mathcal{W}} \mathcal{X}_w^{t,l} * d_w(\mathcal{CT}_c) \quad (2)$$

This objective is subject to:

1. link constraints:

$$\sum_{\forall l \in \mathcal{P}_{s,r}^n} \mathcal{RC}_c(l) \leq \mathcal{R}(l), \forall t, i, j \in \mathcal{X}, i \neq j \quad (3)$$

$$\sum_{\forall i,j \in \mathcal{X}} \delta(l) \leq \delta_{s,r}(l), \forall i, j \in \mathcal{X}, \forall l_{s,r} \in \mathcal{X}, i \neq j \quad (4)$$

$$\mathcal{Z}(d_w) = \mathcal{R}_a^t(l) \geq \mathcal{P}_{s,r}^n(l) d_w(\mathcal{CT}_c), \quad \forall t, \forall l \in \mathcal{P}_{s,r}^n, w \in [1, \mathcal{D}] \quad (5)$$

Equation (3) guarantees that the maximum reservable bandwidth for a link l at any time is less than or equal to the link capacity of that link. E2E delay in $\mathcal{P}_{s,r}^n(l)$ is controlled through (4) to be less than or equal to the maximum demanded delay by demand w at a given time. (5)

characterizes if demand w was successfully assigned at a specified time on all links along the path that have more available resources than required.

2. Slice constraints:
To allocate the demand into a set of traffic slices for each link along the requested path, we use SKM model proposed in [7]. SKM's model contains two techniques; squatting and kicking techniques. Squatting technique helps in sharing of unused resources between higher and lower priority service slices while kicking technique ensures proper QoS for higher traffic priority slices by expelling lower priority slices from resources directly assigned to them. SKM performs four steps to allocate each demand, which are as follows:

Step 1 (MAM): Upon arrival of a demand $d_w(\mathcal{CT}_c)$ belonging to slice c, the following constraints are checked:

$$\mathcal{S}_c(l) \leq \mathcal{RC}_c(l) \tag{6}$$

$$\sum\nolimits_{c=1}^{\mathcal{N}} \mathcal{RC}_c(l) = \mathcal{R}(l) \tag{7}$$

Equation (6) ensures that the actually allocated resources of the new demand plus the old demand do not exceed slice resources constraint while (7), ensures that the total amount of slices resources constraints should be equal to link capacity $\mathcal{R}(l)$. If constraints are satisfied, $d_w(\mathcal{CT}_c)$ is accepted. Otherwise, try step 2.

Step 2 (Squatting-High or RDM): Try to share (squat) unused resources starting from the higher adjacent priority slice upwards until there are enough resources to satisfy $d_w(\mathcal{CT}_c)$. If the resources are enough, then accept $d_w(\mathcal{CT}_c)$, otherwise, try step 3. Note that the total allocatable resources in $\mathcal{CT}_c(l)$ cannot exceed the slice resource constraint $\mathcal{RC}_c(l)$ plus all squatted resources from higher priority slices as in (8). (9) indicates that $\mathcal{SH}_q(l)$ is less or equal to the difference between the slice resource constraint and the minimum between the allocated and the reserved resources for the same slice. Note that the highest priority slice cannot use Squatting-High strategy.

$$\mathcal{S}_c(l) \leq \mathcal{RC}_c(l) + \sum\nolimits_{q=c+1}^{\mathcal{N}} \mathcal{SH}_q(l) \tag{8}$$

$$\mathcal{SH}_q(l) \leq \mathcal{RC}_q(l) - \min(\mathcal{SH}_q(l), \mathcal{RC}_q(l)) \tag{9}$$

Step 3 (Squatting-Low): Try to squat unused resources starting from the lower adjacent priority slice downwards until there are enough resources to satisfy $d_w(\mathcal{CT}_c)$. If the squatted higher resources plus the squatted lower resources satisfy $d_w(\mathcal{CT}_c)$, then accept $d_w(\mathcal{CT}_c)$, otherwise, try step 4. Equation (10) indicates that the total allocatable resources in $\mathcal{CT}_c(l)$ cannot exceed the slice resource constraint plus all squatted resources in both squatting high and low. Moreover, $\mathcal{SL}_q(l)$ works like $\mathcal{SH}_q(l)$, but from lower slices, as shown in (11). Note that the lowest priority slice cannot use Squatting-Low strategy.

$$\begin{aligned}\mathcal{S}_c(l) \leq & \mathcal{RC}_c(l) + \sum\nolimits_{q=c+1}^{\mathcal{N}} \mathcal{SH}_q(l) \\ & + \sum\nolimits_{q=1}^{c-1} \mathcal{SL}_q(l)\end{aligned} \tag{10}$$

$$\mathcal{SL}_q(l) \leq \mathcal{RC}_q(l) - \min(\mathcal{SL}_q(l), \mathcal{RC}_q(l)) \tag{11}$$

Step 4 (Kicking): Try to kick (preempt) the assigned resources partially or totally starting from the lowest priority slice upwards through the lower adjacent slice until there are enough resources to satisfy $d_w(\mathcal{CT}_c)$. If the sum of squatted higher resources plus the squatted lower resources plus the kicked lower resources satisfy $d_w(\mathcal{CT}_c)$, then accept $d_w(\mathcal{CT}_c)$ and count the kicked demands as blocked demand for the same slice else, $d_w(\mathcal{CT}_c)$ will be rejected. Equation (12) ensures that the total allocatable resources cannot exceed the total of slice resource constraint plus all squatted resources in both squatting high and low plus all kicked resources from the lower priority slices. Moreover, the total kicked resources from lower slice q, $\mathcal{K}_q(l)$ cannot exceed the slice resource constraints $\mathcal{RC}_q(l)$ as (13). Note that the lowest priority slice cannot use kicking strategy.

$$\begin{aligned}\mathcal{S}_c(l) \leq & \mathcal{RC}_c(l) + \sum\nolimits_{q=c+1}^{\mathcal{N}} \mathcal{SH}_q(l) \\ & + \sum\nolimits_{q=1}^{c-1} \mathcal{SL}_q(l) + \sum\nolimits_{q=1}^{c-1} \mathcal{K}_q(l)\end{aligned} \tag{12}$$

$$\mathcal{K}_q(l) \leq \mathcal{RC}_q(l) \tag{13}$$

Obtaining an optimal solution for the above-formulated problem would involve computing all possible paths between source and destination, then enumerating all service deployment combinations in order to identify the optimal solution from all the feasible solutions. Evidently, this is a typical NP-hard problem. As such, exact solutions, as well as approaches based on conventional solvers such as CPLEX and Gurobi to solve the above problem, are not feasible in terms of execution time for delay-sensitive 5G applications which is the target of this paper, especially for large scale networks. Therefore, this motivates the adoption of our heuristic approach as it is able to realize the near-optimal solution with feasible execution time.

IV. DEPLOYMENT POLICY OF NETWORK SLICING BASED ON SKM

In this section, we present the proposed deployment policy for allocation of priority demands in a multi-slice network. Mainly, we provide a comprehensive discussion of the different steps included in the execution of the algorithm.

The optimal solution of resource allocation, admission control and QoS management for multiple slice network requires smart algorithms in order to dynamically support, discover, and reserve limited network resources that are often different in type, implementation and priorities. One of the main reasons for the complexity of the network resource allocation problem is the random arrival of user requests and the limited substrate network resources. Nonetheless, the resource allocation problem is an NP-hard due to link allocation problem, since it is difficult to ensure that the routing paths meet the QoS constraints under limited network resources. Besides, the difficulty in selecting the optimal path for various priority requirements and subsequent allocation of resources from source to destination in the physical substrate network. Consequently, most resource allocation algorithms need to resolve resource allocation optimization

problems on time. Thus, to resolve the objective function, this paper introduces the online deployment algorithm, as a priority aware resource allocation algorithm, which can optimize the use of resources by effectively allocating different priority service requirements in terms of link resources based on SKM strategy across the entire network considering the following constrains, namely: Link, slice and E2E delay. If the physical path links contain sufficient resources to provide the resources required for the request, a successful allocation occurs.

A. Description of the Proposed Deployment Policy

The methodology of the proposed deployment policy is described in the flowchart shown in Fig. 1. Also, The algorithm pseudo-code is specified in the Algorithm 1. At each time t, when a request arrives, the deployment process will perform four significant steps to allocate the demand:

1. **Step 1 (Routing algorithm)**: Find all possible paths from the source to destination to allocate the demand in the network.
2. **Step 2 (Resource update)**: In each time unit, before executing the resource allocation process, the algorithm continues to determine whether any request is expired or not, to eliminate its requests from the hosting resources, and updates the entire substrate network accordingly.
3. **Step 3 (Allocating decisions)**: Check all potential paths according to the available resources metric defined using a specific allocation strategy.
   - On each path (check the node for each node), check the required delay and resources for the request. If the path delay and the resources available along this path meet the requirements of the request, add the path to the list of potential paths. Otherwise, ignore the path.
   - Define a specific allocation strategy (SKM) to optimize individual node allocation based on efficient use. SKM allocates resources and control process to check whether network resources are sufficient to serve user demand resources and QoS requirements.
4. **Step 4 (Path selection strategy)**: Sort the potential paths according to the maximum resources available on the links and choose the best path to allocate the demand. If the available resources are the same in two or more paths, the potential paths will be sorted according to the path that consumes the least resources first.

More details on the steps of the proposed algorithm are discussed in the subsections.

B. Routing Algorithm

The proposed algorithm starts by checking for all possible paths to determine the transmission path for the request from source to destination. In order to find all possible paths to assign service demand, many methods and strategies such as Brute force and others can be used for this task. Due to the complexity of the path calculation, we used the k-shortest path algorithm, which is used to determine the k-shortest path where k is an integer number of shortest physical paths appropriate to satisfy the bandwidth resource requirements for various priority slices. Please note that our work assumes that the substrate network topology is considered to be constant. Thus, the main factors that express the substrate networks, for example, the quantity and connectivity of nodes and links in the substrate network, are also constant and do not change, but only the capacities of their resources differ due to their use after each time period t.

Algorithm 1: Pseudo-code of the proposed deployment algorithm

```
INPUT: G[X, L] of X routers and L links, and set of service
 demands D to be allocated;
OUTPUT: Allocation status, Z(d_w): Succeed, R: Reject
while t ≠ 0 do
    D_selected ← D_{(i−1)n+1:in} Fetch n demands consecutively from
      D;
    D_checked ← Φ(D_selected) Check Expiry of the Demands;
    D_sorted ← SortDemands(D_checked) Sort Demands;
    for Each Demand d_w = d_w(CT_c) ∈ D_sorted arriving the substrate
      network randomly at time t do
        Initialize A as empty set;
        Start SKM assignment process Loop D :Demands;
        for Each l ∈ P^n_{s,r}(l) ∈ k-shortest path list (Step 1) do
            Ensure that the link delay meet Demand d_w delay using (4);
            Calculate available resources of a link l using (14);
            if Demand d_w assignment process was successful for
              P^n_{s,r}(l) then
                Add P^n_{s,r}(l) into A as potential path;
            end
        end
        if Count A > 0 then
            for Each P^n_{s,r}(l) ∈ A do
                Determine path available resources ;
                R^t_a(l) ⇐ min(R^t_a(l));
                for (P^n_{s,r}(l) and R^t_a(l) > 0) do
                    Select the optimal path based on highest available
                      resources as (16);
                    if two paths or more have same amount of
                      available resources along the path then
                        compute the least consumed resources path
                          as (17);
                        Z(d_w): Succeed d_w for P^n_{s,r}(l);
                        Evaluate Metrics using (18)–(23)
                    end
                end
            end
        end
        else
            R: Reject d_w for P^n_{s,r}(l)
        end
    end
end
```

C. Resource Update

In each unit time, the algorithm verifies the expiration of requests and substrate network resources before allocating resources to the subsequent request. In other words, the algorithm fetches a set of multiple requests sequentially from the request generation file ($\mathcal{D}$) list and checks for the expiration of the allocated requests. Later checking the expiration stage, requests will be classified according to size and priority level from highest to lowest. Once the arrangement stage occurs, the process assignment of step 2 will be used to allocate requests along the network topology paths.

D. Embedding Decisions

This is the essential step that ensures that every request in the network is allocated, which solves the problem of allocating resources along the required physical path. To ensure allocation,

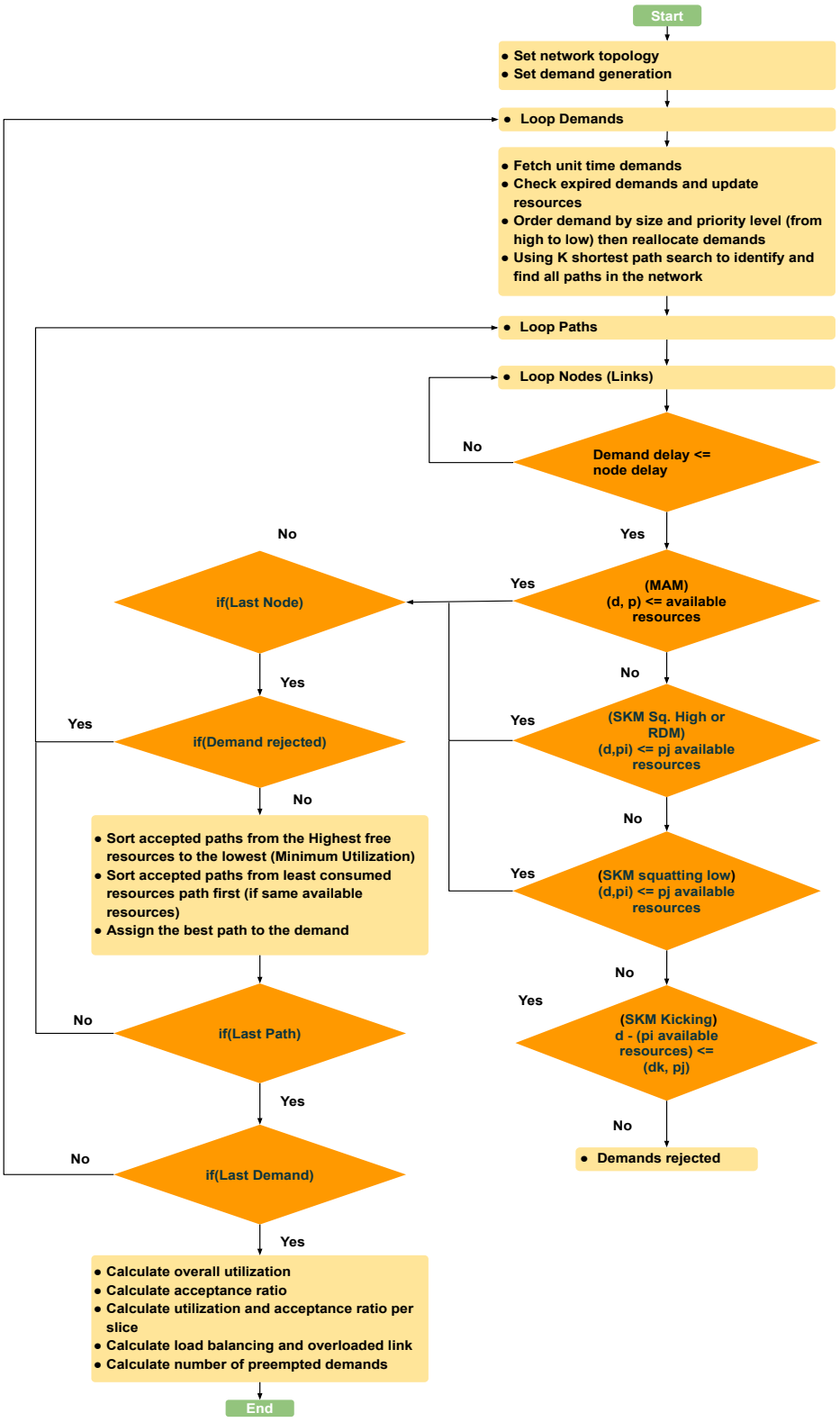

Fig. 1. Flowchart 1 illustrates the methodology structure used by the proposed deployment policy. It begins with the routing step, then it is followed by the allocating and resource update steps, and it ends with the assessment step.

we first verify that the path delay meets the demand delay. Then, we use SKM strategy in the nodes along the path to optimizing requests in terms of bandwidth by exploiting the partition and reservation of resources according to different priority slices and the flexibility to use the full amount of resources when no slice needs them. For instance, Fig. 2 describes the process of allocating a service request in the physical routing path based on SKM strategy. The service request w contains the substrate source

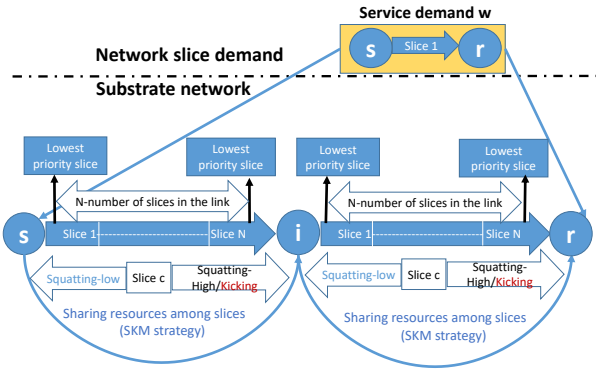

Fig. 2. Illustration of link mapping along the path using SKM strategy

node s and the substrate destination node r. The path (s, r) is the chosen physical routing path that is allocated by the service request w and contains two links si and ir for transmitting traffic from the node s to node r. Specifically, in the time period t_{d_w}, the service request w is assigned to a priority slice defined by the SKM steps in each link along the path that satisfies the bandwidth and delay required by the service request. Requests are arranged according to size and priority to reduce the number of kicking actions per unit time and also to optimize resource usage in the network because arranging requests according to size resulting in higher utilization rate in most cases. In each time unit, SKM executing a sorting process before starting a new request allocation. However, if there are no resources available for all the candidate paths to accepting the request, the request will be rejected and move to the next request from step 1. This process ends when there are no other paths to accept the requests.

E. Path Selection Procedures

In this step, the algorithm will perform three procedures to select the optimal path from the list that contains all the potential routing paths that can accept the demand, which is provided using the defined resource allocation strategy (SKM).

More details on the three procedures are described below.

Procedure 1, at any time variable t, the algorithm will determine the available link resources as follows:

$$\mathcal{R}_a^t(l) = \mathcal{R}(l) - \mathcal{R}_z^t(l), \tag{14}$$

$$\mathcal{R}_z^t(l) = \sum_{c=1}^{\mathcal{N}} \mathcal{R}(l) - (\mathcal{RC}_c(l) - \min(\mathcal{S}_c(l), \mathcal{RC}_c(l))). \tag{15}$$

Equation (14) illustrates the computation of available resources in a link. Furthermore, $\mathcal{R}_z^t(l)$ is calculated by the summation of the difference among $\mathcal{R}(l)$, and the minimum between allocated and reserved resources for each class as (15). $\mathcal{N}$ is the number of slices across the link l.

Next, the path available resources are the min value of $\mathcal{R}_a^t(l) = \min(\mathcal{R}_a^t(l))$.

Procedure 2, is to choose the best routing path considering the constraints of the resources of a link. The optimal routing path can be addressed concerning QoS constraints when its links meet resource constraints.

The links along the requested path meeting the constraints of the resources are described by (16):

$$\max\left\{\min(\mathcal{R}_a^t(l)) \geq \mathcal{P}_{s,r}^n(l) d_w(\mathcal{CT}_c)\right\}, \quad \forall l \in \mathcal{P}_{s,r}^n(l), n \in [1, \mathcal{P}_{\text{set}}^{s,r}(l)] \tag{16}$$

Procedure 3, if more than one routing path has the same amount of available resources, the best path will be the one with the lowest amount of bandwidth consumed.

To compute the less consumed path (17).

$$\min\left\{\sum_{\forall l \in \mathcal{P}_{s,r}^n(l)} \mathcal{R}_z^t(l)\right\}, n \in [1, \mathcal{P}_{\text{set}}^{s,r}(l)] \tag{17}$$

F. Time Complexity

In general, the complexity of the exact mixed-integer programming (MIP) problems is explosive in full connected networks with a relative big scenario (already with 10 nodes, or more) resulting in unaffordable execution times to solve problems with large network sizes. Actually, the problem of finding all paths in fully connected networks belongs to the family of NP-Complete problems [29]. This is even worse when considering SKM as a policy for network slicing, regarding the additional complexity, as discussed in [7]. To reduce significantly the complexity of the problem, k-shortest path is considered, which is not affecting the results, since the shortest paths will be chosen most probably because they consume fewer resources.

Our proposed algorithm has two main computation stages; stage 1 consists of finding the k-shortest possible paths in the substrate network from which a service request can be deployed. Then, the second stage involves obtaining the best path among all the feasible k-shortest path on which the request is then provisioned. The first stage is obtained using the k-shortest path algorithm, whose time complexity can be approximated as $O(k\,\mathcal{X}(\mathcal{L} + \mathcal{L}\,log\,\mathcal{L})$ where $\mathcal{X}$ and $\mathcal{L}$ the number of nodes and edges respectively, of the substrate network [30]. The second stage 2 involves checking for the feasible paths among k-shortest paths which is linear in terms of the number of paths and nodes constituting each path. Then, the feasible paths are sorted according to highest available resources and the time complexity of this cannot to exceed $O(k\,log\,k)$ where k is the maximum number of obtained feasible paths [31]. Note that the feasible paths are usually few in number, especially under congestion scenarios, hence such a step imposes a negligible execution overhead on our algorithm.

G. Evaluation Metrics

The performance of the proposed deployment algorithm will be evaluated based on acceptance ratio, total resource utilization, load balancing, overloaded link and total number of preempted demands across the network.

G.1 Acceptance ratio, $\mathcal{AR}$

It is the ratio in which the description of how the proposed algorithm works and is determined by measuring the averaging

and dividing number of requests accepted in the system in each time period by the total number of requests. [28], [32], [33].

$$\mathcal{AR} = \frac{1}{\mathcal{T}} \sum_{\forall t \in \mathcal{T}} \frac{\mathcal{AD}}{\mathcal{D}} * 100. \tag{18}$$

G.2 Average acceptance ratio per slice, $\mathcal{AR}_c$

It is the ratio in which the description of how the proposed algorithm works and is determined by measuring the averaging and dividing number of requests accepted by each slice separately in each time period by the total number of requests for the same slice. Is a ratio to represent how the proposed algorithm is performing and is calculated by averaging and dividing the number of successfully allocated demands by each slice separately at each time interval by total demands for the same slice.

$$\mathcal{AR}_c = \frac{1}{\mathcal{T}} \sum_{\forall t \in \mathcal{T}} \frac{\mathcal{AD}_c}{\mathcal{D}_c} * 100. \tag{19}$$

G.3 Average resource utilization, $\mathcal{U}$

It indicates the average utilization of substrate network links after all simulation iterations. It is determined as the ratio between the resources used and the link capacity, averaged across all substrate network links [33].

$$\mathcal{U} = \frac{1}{\mathcal{T}} \sum_{t \in \mathcal{T}} \frac{1}{\mathcal{L}} \sum_{\forall l \in \mathcal{L}} \frac{\mathcal{R}_z^t(l)}{\mathcal{R}(l)}. \tag{20}$$

G.4 Average resource utilization per slice, $\mathcal{U}_c$

It indicates the average utilization of slices is substrate network links after all simulation iterations. It is determined as the ratio between the resources used by each slice separately and the link capacity, averaged across all substrate network links

$$\mathcal{U}_c = \frac{1}{\mathcal{T}} \sum_{t \in \mathcal{T}} \frac{1}{\mathcal{L}} \sum_{\forall \mathcal{CT}_c(l) \in \mathcal{L}} \frac{\mathcal{S}_c(l)}{\mathcal{R}(l)} * 100. \tag{21}$$

G.5 Average load balancing, $\mathcal{LB}$

This parameter provides an indicator of the possibility of our proposed deployment algorithm to use the substrate resources in a uniform manner. In this study, we adopt the variance of the link resource utilization as the measure of the load balancing in the network [28].

$$\mathcal{LB} = \frac{\sum_{\forall l \in \mathcal{L}} (\mathcal{U} - \mu)^2}{|\mathcal{L}|}, \tag{22}$$

where $\mathcal{L}$ is the set of all links in the network and $|\mathcal{L}|$ is the cardinality of this set. $\mathcal{U}$ as given in (20) is the average resource utilization on the link l and μ is the mean value of this parameter across the network. The lower value of $\mathcal{LB}$ is the better load balancing performance.

G.6 Average overloaded link, $\mathcal{L}_{ov}$

This parameter provides an indicator of the possibility of our proposed algorithm to use link loads in a uniform manner. Higher relative loads or overloaded links will certainly be the goal of having long-serving queues, and so, higher delay and a higher rate of packet loss will occur. Besides, to have a sufficient QoS, we should reduce the link loads and the number of network links in the demand. Mathematically, the overloaded link performance in this study, $\mathcal{L}_{ov}$ along the network is defined in (23). The lower value of $\mathcal{L}_{ov}$ is the best-overloaded link performance.

$$\mathcal{L}_{ov} = \max(\mathcal{U} - \mu), \forall l \in \mathcal{L} \tag{23}$$

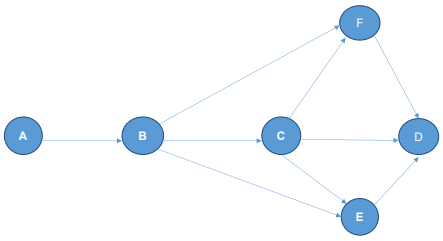

Fig. 3. An illustrative diagram showing a physical network of 6 nodes and 9 links receiving three service demands.

H. General Illustrative Example

In this subsection, we give a brief numerical example to explain the proposed deployment algorithm considering a network topology (directed graph) consisting of 6 nodes and 9 links as shown in Fig. 3. Moreover, we assume that all links have the same capacity of 30 units and the same delay of 1 ms. Every link is distributed into 3 priority slices and all slices have the same quantity of resources equal to 10 units. Furthermore, four demands (from one source to different destinations) need to be allocated based on the resources available across the network as described below. Please note that we considered the k-shortest path algorithm with k equal to 2 for this example to map the different demands with the generation rate set to one demand per each unit time as follows:

#1: From A to D, 15 units, priority 2, delay 3 ms, duration=3
#2: From A to E, 10 units, priority 3, delay 2 ms, duration=2
#3: From A to F, 20 units, priority 3, delay 5 ms, duration=4
#4: From A to F, 24 units, priority 1, delay 4 ms, duration=6

For an example scenario, Table 2 shows the SKM behaviour in the above-demonstrated example with an on-line scenario in terms of allocating and reservation of resources for the demands by considering network slices and the links capacities. In other words, the table explains the basis of our proposed deployment policy behaviour in the example shown above with an online mode in terms of resource allocation and reservation for the demand by considering traffic slices and link capacities. Moreover, the above example demonstrates SKM's ability to efficiently make decisions to determine paths according to priority and delay demands. The first column of the table shows the optimal computing paths for mapping four E2E service demands across substrate network per unit time. The second left column demonstrates the routing step of the deployment policy. Before starting the process of allocation in each unit time, the algorithm checks the expiry of allocated demands and the substrate network is updated as shown in column three (expired demands) and five (available resources in each link of path). For instance, before the arrival of demand $\#4 : 24_{1,4}(6)$, the demand $\#2 : 10_{3,2}(0)$

was expired and the available network resources was updated. Please note that the allocation of the demands is performed after the sorting process in each unit as shown in column six (Alive demands after sorting). For example, when the demand $\#4 : 24_1(6)$ arrives at the network, firstly, we must do rearranging including the new demand to the existing alive demands according to size and priority. Next, the demands $\#3 : 20_{2,5}(3)$ and $\#4 : 24_{1,4}(6)$ are allocated respectively. Also, in this example, we show how SKM uses kicking operation to favour the higher priority slices as in unit time 3. In unit time 3, there is no enough resources in the network to accept the new higher priority demand $\#3 : 20_{3,5}(3)$ so, the algorithm checks all alive demands that can be kicked to favour the new demand. Accordingly, demand $\# : 15_{2,3}(3)$ is expelled from the network to allocate new higher priority demand as shown in column seven (execution). Finally, after verifying all potential paths that can allocate the demand, the algorithm determines the optimal path based on the available resources, as shown in the last column from Table 2. Table 3 shows the results of the online proposed algorithm in terms of the $\mathcal{U}_c$, $\mathcal{U}$, $\mathcal{AR}_c$, $\mathcal{AR}$, $\mathcal{LB}$, and $\mathcal{L}_{ov}$. From the results, slice 3, accepted two demands until the observation time $\#2 : 10_{3,2}(2)$, $\#3 : 10_{3,5}(2)$ across the network. Please note that the low priority slice demand $\#1 : 15_{2,3}(3)$ has been kicked to satisfy the higher priority slice demand $\#3 : 20_{3,5}(2)$.

V. SIMULATION AND ANALYSIS

In this section, the performance analysis of the proposed algorithm is discussed including the resource allocation algorithms, the different scenarios that are considered in this work, simulation settings and the obtained results.

A. Simulation Scenarios and Compared Algorithms

Table 4 provides a high-level comparison between SKM, Smart Alloc, AllocTC, RDM and MAM algorithms, listing their used strategies, and how they embed the demands onto different priority slices along the path. The algorithms are compared considering a number of simulation scenarios with each scenario intended to meet a given objective. In all simulations, algorithms were developed using Eclipse IDE for Java Developers, version: Mars.2 Release (4.5.2) and conducted on a desktop computer running Windows operating system with the following specifications: Intel(R) Core(TM) 2 CPU 6400 @ 2.13GHz Memory 6GB. The scenarios considered for the performance analysis are as follows:

1. Scenario 1: SKM's overall performance in terms of acceptance rate, resource usage, as well as load balancing and link overload will be compared with the latest online algorithm such as Smart Alloc from Bahnasse *et al.* (2018) [22] as described in the Section V.B.1 considering the online case with different arrival rates (λ). The objective of this scenario is to assess the effect of λ on SKM against the state of art algorithms.
2. Scenario 2: This scenario involved an online simulation under mesh network topology and various generated traffic load (same load in all slices, high load in lower priority slices and high load in higher priority slices) for traffic slices of all priorities as detailed in Section V.B.2. The objective of this scenario is to assess the impact of mesh topology in which all nodes are reachable in a single hop from each other node on the performance of SKM against the state of the art algorithms considering different load distributions.
3. Scenario 3: This scenario involved an online simulation under NSF network topology and various generated traffic load for traffic slices of all priorities as detailed in Section V.C. The objective of this scenario is to analyze the impact of NSF network on our proposed algorithm performance against the state of the art algorithms considering different load distributions. This is because NSF topology faces more bottlenecks which further complicates resource allocation and QoS management compared to mesh topology.

B. Simulation Settings and Obtained Results

This section presents simulation settings, the results obtained from the different scenarios and their analysis.

B.1 Scenario 1: Impact of arrival rate

In this scenario, we assess the impact of different arrival rates on the performance of SKM strategy against other sates of the art algorithms in the network topology adopted in [22], with the aim of optimizing resource utilization while improving acceptance rate in higher priority slices.

We used in this scenario a substrate network with 8 nodes and 9 links, the link bandwidth resources are given as real numbers, chosen as 150 or 300 units, and the delay in each substrate link was set to 1 ms. The number of slices per link is equal to 3 slices in links that contain 150 units have the same capacity and are equal to 50 units, and slices in links that contain 300 units have the same capacity and equal to 100 units. We assume that the demands arrive with an exponentially distributed lifetime with an average of 100 time units. In this evaluation scenario, the choice of both source and destination nodes for each request is randomly determined. The arrival rate of incoming demands λ is varied from 1 to 4 per 100 time units, over simulation time of 20,000 units. The size of demands were real numbers uniformly distributed between 1–20 units, while the delay for each demand was randomly selected between 1 and 5. For the routing step, using the k-shortest path, the maximum value of k was set to 5. Table 5 summarizes all simulation scenarios parameters.

From the results in Figs. 4(a) and 4(c), the average links utilization and acceptance ratio for SKM, Smart Alloc and AllocTC resulted in 78.5% for $\mathcal{U}$ and 59.70% for $\mathcal{AR}$, which were higher than MAM and RDM by 5.09% and 3.2% for $\mathcal{U}$ and by 3.98% and 2.3% for $\mathcal{AR}$ respectively. As expected, SKM outperforms the rest of the algorithms in terms of average $\mathcal{U}_3$ and average $\mathcal{AR}_3$ by 11% and 8% respectively, for the different arrival rates (see Figs. 4(a) and 4(c)). In case of including E2E delay, SKM links' utilization is less than without delay, since considering delay constraints leads to lower $\mathcal{AR}$, resulting in low resource utilization, thus has less utilized substrate links (see Figs. 4(b) and 4(d)).

Fig. 4(e) results show that SKM, AllocTC and Smart Alloc

Table 2. Numerical example showing the basics of our proposed deployment algorithm.

#of demand: $d_{p,\delta}(t)$ & path selection	**Allocation (SKM on-line)**						
3 PRIORITY SLICES	**(Unit time 1) Paths to be checked/sorted ($P_{A,B,C,D}, P_{A,B,F,D}$)**						
$\#1:15_2(3)$ **The selected path after checked all paths is $P_{A,B,C,D}$**	**Paths to be selected**	**Expired demands**	**New demands to be processed**	**Available Resources in each link of path**	**Alive demands after sorting**	**Execution**	**Evaluated metric**
	$P_{A,B,C,D}$ in this path: [4]nodes and [3] links	–	$\#1:15_{2,3}(3)$	$A-B$ (10,10,10) $B-C$ (10,10,10) $C-D$ (10,10,10)	–	Accepted delay 3 ms (10,0,5) RDM (10,0,5) RDM (10,0,5) RDM	Ra_(A,B) =30-15 = 15 units Ra_(B,C) =30-15 = 15 units Ra_(C,D) =30-15 = 15 units Min available resources of Ra given slice for links along the path (Min Ra) = 15 units. Consuming resources along the path = 15+15+15 = 45 units
	$P_{A,B,F,D}$ in this path: [4]nodes and [3] links			$A-B$ (10,10,10) $B-F$ (10,10,10) $F-D$ (10,10,10)		Accepted delay 3 ms (10,0,5) RDM (10,0,5) RDM (10,0,5) RDM	Ra_(A,B) =30-15 = 15 units Ra_(B,F) =30-15 = 15 units Ra_(F,D) =30-15 = 15 units Min available resources of Ra given slice for links along the path (Min Ra) = 15 units. Consuming resources along the path = 15+15+15 = 45 units
3 PRIORITY SLICES	**(Unit time 2) Paths to be checked/sorted ($P_{A,B,E}, P_{A,B,C,E}$)**						
$\#2:10_{3,2}(2)$ **The selected path after checked all paths is $P_{A,B,E}$ (the least consumed resources path)**	**Paths to be selected**	**Expired demands**	**New demands to be processed**	**Available Resources in each link of path**	**Alive demands after sorting**	**Execution**	**Evaluated metric**
	$P_{A,B,E}$ in this path: [3] nodes and [2] links	–	$\#2:10_{3,2}(2)$	$A-B$ (10,0,5) $B-E$ (10,10,10)	$\#2:10_{3,2}(2)$ $\#1:15_{2,3}(2)$	Accepted delay 2 ms (5,0,0) $SL1$_(A,B) (10,10,0) $SL1$_(B,E)	Ra_(A,B) = 5 units Ra_(B,E) = 20 units Min available resources on the other priority slices for links along the path =5 units. Consuming resources along the path = 25+10 = 35 units
	$P_{A,B,C,E}$ in this path: [4] nodes and [3] links			$A-B$ (10,0,5) $B-C$ (10,0,5) $C-E$ (10,10,10)		Rejected delay 2 ms	Discarded path due to rejected delay
3 PRIORITY SLICES	**(Unit time 3) Paths to be checked/sorted ($P_{A,B,F}, P_{A,B,C,F}$)**						
$\#3:20_{3,5}(4)$ **The selected path after checked all paths is $P_{A,B,F}$ (the least consumed resources path)**	**Paths to be selected**	**Expired demands**	**New demands to be processed**	**Available Resources in each link of path**	**Alive demands after sorting**	**Execution**	**Evaluated metric**
	$P_{A,B,F}$ in this path: [3] nodes and [2] links	–	$\#3:20_{3,5}(4)$	$A-B$ (5,0,0) $B-F$ (10,10,10)	$\#3:20_{3,5}(4)$ $\#2:10_{3,2}(1)$ $\#1:15_{2,3}(1)$	(0,0,0) $K2$_(A,B) (10,0,0) $SL2$_(B,F)	Ra_(A,B) = 0 units Ra_(B,F) = 10 units Min available resources on the other priority slices for links along the path = 0 units. Consuming resources along the path = 30+20 = 50 units
	$P_{A,B,C,F}$ in this path: [4] nodes and [3] links			$A-B$ (5,0,0) $B-C$ (10,0,5) $C-F$ (10,10,10)		(0,0,0) $K2$_(A,B) (0,10,0) $K2$_(B,C) (10,0,0) MAM	Ra_(A,B) = 0 units Ra_(B,C) = 10 units Ra_(C,F) = 10 units Min available resources on the other priority slices for links along the path = 0 units. Consuming resources along the path = 30+20+20 = 70 units
3 PRIORITY SLICES	**(Unit time 4) Paths to be checked/sorted ($P_{A,B,F}, P_{A,B,C,F}$)**						
$\#4:24_{1,4}(6)$ **The demand is rejected**	**Paths to be selected**	**Expired demands**	**New demands to be processed**	**Available Resources in each link of path**	**Alive demands after sorting**	**Execution**	**Evaluated metric**
	$P_{A,B,F}$ in this path: [3] nodes and [2] links	$\#2:10_{3,2}$ (0)	$\#4:24_{1,4}(6)$	$A-B$ (5,0,5) $B-F$ (10,0,0)	$\#3:20_{2,5}(3)$ $\#4:24_{1,4}(6)$	Accepted delay 3 ms (5,0,5) Rejected (10,0,0) Rejected	Discarded path due to rejected allocation
	$P_{A,B,C,F}$ in this path: [4]nodes and [3] links			$A-B$ (5,0,5) $B-C$ (10,10,10) $C-F$ (10,10,10)		Accepted delay 3 ms (5,0,5) Rejected (0,0,6) RDM (0,0,6) RDM	Discarded path due to rejected allocation

have very close performance with and without delay when increased load in terms of average $\mathcal{LB}$ and average $\mathcal{L}_{ov}$ due to the algorithms have similar utilization. On the other hand, MAM and RDM showed lowest performance with and without delay among other algorithms in terms of $\mathcal{LB}$ and $\mathcal{L}_{ov}$ by 0.011, 0.0027 for $\mathcal{LB}$ and by 0.025, 0.015 for $\mathcal{L}_{ov}$ respectively, where more links not being fully used across the network. In the case of including E2E delay, SKM links' load balancing performance is

Table 3. Results of the performance metrics after applying the proposed deployment algorithm in an online example scenario.

Links utilization:	**Utilization per slice:**	**Accepted demands per slice:**
Utilization for link $(A - B)$ = (20) / 30 = 66.67% Utilization for link $(B - -C)$ = (0) / 30 = 0% Utilization for link $(C - -D)$ = (0) / 30 = 0% Utilization for link $(B - -E)$ = (0) / 30 = 0% Utilization for link $(B - -F)$ = (20) / 30 = 66.67% Utilization for link $(C - -E)$ = 0% Utilization for link $(C - -F)$ = 0% Utilization for link $(E - -D)$ = 0% Utilization for link $(F - -D)$ = 0%	Utilization for slice (1) = 0 / (9*30) = 0% Utilization for slice (2) = (0) / (9*30) = 0% Utilization for slice (3) = (20) / (9*30) = 7.40%	For slice (1): 0 Demand(s) of 1 – acceptance = 0% For slice (2): 0 Demand(s) of 1 – acceptance = 0% For slice (3): 2 Demand(s) of 2 – acceptance = 100.00%
Average utilization of the Network = (20/30 + 20/30) / 9 = 14.81%		
Average acceptance ratio = 2/ 4 = 50%		
$\mathcal{LB}$ = [(66.67% – 14.81%)^2 + (66.67% – 14.81%)^2 + (0% – 14.81%)^2 + (0 – 14.81%)^2 + (0% – 14.81%)^2 + (0 – 14.81%)^2 + (0 – 14.81%)^2 + (0 – 14.81%)^2 + (0 – 14.81%)^2]/ 9= 0.26		
$\mathcal{L}_{ov}$ = (66.67% – 14.81%) = 0.52		
Number of preempted demands = 1 ($\#1 : 15_{2,3}(3)$)		

Table 4. Comparing SKM to Smart Alloc, AllocTC, RDM and MAM algorithms.

Item	**SKM**	**Smart Alloc**	**AllocTC**	**RDM**	**MAM**
Scenario	Online				
Goal	maximize overall resource utilization				
Strategy	Confirm slices' constraints. Check available links capacities then allocate				
Resource allocation	The squatting strategy allows sharing unused resources between all $\mathcal{CT}s(l)$	It is classified demands based on their threshold. Whatever the priority of the resources required belonging to the high threshold, the latter can benefit from the loans of the other slices	It allows an opportunistic sharing of the link resources among the different slices	The lower priority $\mathcal{CT}s(l)$ can reuse the free resources of higher priority $\mathcal{CT}s(l)$ and no the reverse	Each $\mathcal{CT}_c(l)$ has its private resources, and if the latter is not used, it cannot be allocated to another $\mathcal{CT}_c(l)$
Best path selection	Select the highest available resource path. If two or more paths have the same resources, determine which resource is the least consumed path.				
$\mathcal{P}_{\mathcal{HTL}}$	Yes	Yes	Yes	Yes	No
$\mathcal{P}_{\mathcal{LTH}}$	No	No	Yes	No	No
$\mathcal{K}_q(l)$	Yes	No	No	No	No
E2E delay	Yes				

less than without delay, mainly since SKM with E2E delay utilized less network resources, thus has more overloaded substrate links.

As shown in Fig. 4(f), SKM outperforms RDM, Smart Alloc and AllocTC in terms of average $\mathcal{P}_{re}$ by 5, 10, and 23 demands respectively due to kicking operation. In the case of including E2E delay, number of preempted demands of SKM is less than without delay, mainly since SKM with E2E delay accepted less number of demands.

Impact of delay on all algorithms: The impact of E2E delay on all algorithms, was negative in general overall simulations as shown in Fig. 4. The results reflect how the algorithms are performing better without delay than when they were included.

B.2 Scenario 2: Performance considering mesh topology

In this scenario, we assess the impact of mesh topology on the performance of SKM strategy with and without delay against MAM, RDM and AllocTC under various traffic loads. Note that the demands are generated in this scenario with a fixed demands lifetime equal to 1-time unit and the size of each demand is also fixed equal to 1 unit as the minimum granularity for allocation. Each demand has single priority generated in a random manner from (1 to 3) with a generation rate of demands per each unit time equal to 2500 demand. The total number of demands among slices generated until 10 unit time is 25,000 for each experiment (see Table 5). Moreover, we consider in this scenario three experiments in order to analyze the performance of SKM under several metrics and different load distributions between different priority slices. Please note that in all experiments, the capacity of each slice along the whole network is 500 unit ($\mathcal{RC}_c(l)$ * 10 links = total size of the slice across the network). The evaluation experiments are as follows:

- Experiment 1: More traffic load in lower priority slices.
- Experiment 2: Same traffic load in all priority slices.
- Experiment 3: More traffic load in higher priority slices.

The objective of experiment one is to illustrate that SKM has similar behaviour to RDM and AllocTC at high loads for lower priority slices across the network. The simulation experiment

Table 5. Simulation scenarios parameters.

Substrate network			
Parameter	**Value**		
	Scenario 1	**Scenario 2**	**Scenario 03**
Nodes/Links	8/9	Mesh 5/10	NSF 14/21
Link delay	1 ms	1 ms	1 ms
Link capacity	fixed.[150 units or 300 units]	150 units	150 units
Traffic slices capacity	fixed,[50 units or 100 units]	50 units	50 units
Traffic slices priorities	Unif,[1 – 3]	Unif,[1 – 3]	Unif,[1 – 3]
Time units	0 – 20,000	0 – 25,000	0 – 40,000
Generation rate	Unif,[1 demand – 4 demand]/100 unit times	2500/1 unit time	4000/1 unit time
Demands/lifetime	500 time units	1 time unit	1 time unit
k **value for** k**-shortest path**	5	5	10
Demands			
	Scenario 1	**Scenario 2**	**Scenario 3**
Source	Random	Random	Random
Destination	Random	Random	Random
Demands/size	Unif, [1 unit – 20 units]	1 unit	1 unit
Demands/lifetime	500 time units	1 time unit	1 time unit
Delay	Unif,[1 ms – 5 ms]	Unif,[1 ms – 5 ms]	Unif, [1 ms – 10 ms]
Load volume traffic for each slice per-each unit time in each scenario experiments	-	Experiment 1, high load in lower priority slices: 1250, 833, 417 Experiment 2, same load in all priority slices: 833, 833, 833 Experiment 3, high load in higher priority slices: 417, 834, 1250	Experiment 1, high load in lower priority slices: 2000, 1500, 500 Experiment 2, same load in all priority slices: 1333, 1333, 1334 Experiment 3, high load in higher priority slices: 500, 1500, 2000

enforces the share or squatting strategy that is inherent to RDM across the network. The objective of experiment two is to illustrate that the SKM guarantees to accept more demands for higher priority slices than AllocTC, RDM and MAM in case of same loads in traffic slices across the network. The objective of experiment three is to illustrate that SKM has an similar behaviour to AllocTC before the saturation case when the load is high for higher priority slices across the network. This is verified by enforcing the share strategy of AllocTC or squatting strategy. Also, SKM achieves more accepted demands than AllocTC and RDM at high loads for higher priority slices, which is due to being stricter on priorities than the other algorithms after saturation case.

Fig. 5 shows the results for each algorithm with and without delay in terms of $\mathcal{U}$, $\mathcal{AR}$, $\mathcal{U}_c$, $\mathcal{AR}_c$, $\mathcal{P}_{re}$, $\mathcal{LB}$ and $\mathcal{L}_{ov}$ using different traffic load according to experiments 1–3.

Experiment 1, considering high load in lower priority slices: In terms of $\mathcal{U}$ and $\mathcal{AR}$, Figs. 5(a) and 5(c) show that SKM, AllocTC and RDM resulted in 100% $\mathcal{U}$ and 59% $\mathcal{AR}$ where 1475 demands are accepted from 2500 demands per each unit time. On the other hand, MAM achieved the lowest performance and resulted in 95.2% $\mathcal{U}$ and 56% $\mathcal{AR}$ where 1400 demands are accepted from 2500 demands per each unit time. As expected, SKM, AllocTC, RDM and MAM have similar behaviour in terms of $\mathcal{U}_3$ and $\mathcal{AR}_3$ by achieving 25.60% and 100% (417/417) $\mathcal{AR}_3$, respectively. This is due to the fact that the load distributions on slice 3 across the network was lower than its capacity (the demanded resources for slice 3 was 417 unit). Moreover, SKM outperforms AllocTC, RDM and MAM by 18.02%, 18.34%, and 22.54% in terms of $\mathcal{U}_2$ due to kicking operation. The same trend of performance is observed in Fig. 5(c) in terms of $\mathcal{AR}_c$.

Fig. 5(e) illustrates that SKM, AllocTC and RDM have a very close performance in terms of $\mathcal{LB}$ and $\mathcal{L}_{ov}$, this is because these algorithms have a similar value of network resource utilization which is 100% $\mathcal{U}$ (almost all links are fully used). Moreover, MAM gives the lowest performance in terms of $\mathcal{LB}$ and $\mathcal{L}_{ov}$ resulted in 0.0011 $\mathcal{LB}$ and 0.047 $\mathcal{L}_{ov}$ where more links not being fully used across the network. Moreover, as shown in Fig. 5(f), SKM, AllocTC and RDM resulted in 740, 739, and 745 in terms of $\mathcal{P}_{re}$, respectively.

Experiment 2, considering same load in all priority slices: Figs. 5(a) and 5(c) illustrate that the SKM, AllocTC, RDM and MAM resulted in 100% $\mathcal{U}$ and 58.88% $\mathcal{AR}$ where 1472 demand from 2500 are accepted per each unit time. Moreover, SKM outperforms MAM, RDM and AllocTC in the highest priority slice by 20.47% in terms of $\mathcal{U}_3$ and 41.17% in terms of $\mathcal{AR}_3$. Also, from the results, SKM outperforms MAM, RDM and AllocTC in slice 2 by 11.94% in terms of $\mathcal{U}_2$ and by 17.39% in terms of $\mathcal{AR}_2$ (as the expected from the behaviours) due to the kicking operation.

Fig. 5(e) shows that SKM, AllocTC, RDM, and MAM have similar performance in terms of $\mathcal{LB}$ and $\mathcal{L}_{ov}$ and have resulted in almost zero since all links are used across the network. Moreover, as shown in Fig. 5(f), SKM outperforms AllocTC and RDM by 219 and 38 in terms of $\mathcal{P}_{re}$, respectively due to kicking operation.

Experiment 3, considering high load in higher priority slices: Figs. 5(a) and 5(c) illustrate that the SKM and AllocTC have similar performance in terms of $\mathcal{U}$ and $\mathcal{AR}$ by achieving 100% $\mathcal{U}$ and 59% where 1475 demand are accepted from 2500 demand per each unit time. On the other hand, MAM and RDM performance are the lowest one among the four strategies by achieving 94.5% in terms of $\mathcal{U}$ and 55.54% in terms of $\mathcal{AR}$. This is because there is no ability to share resources among the slices. Furthermore, SKM outperforms AllocTC, RDM and MAM in

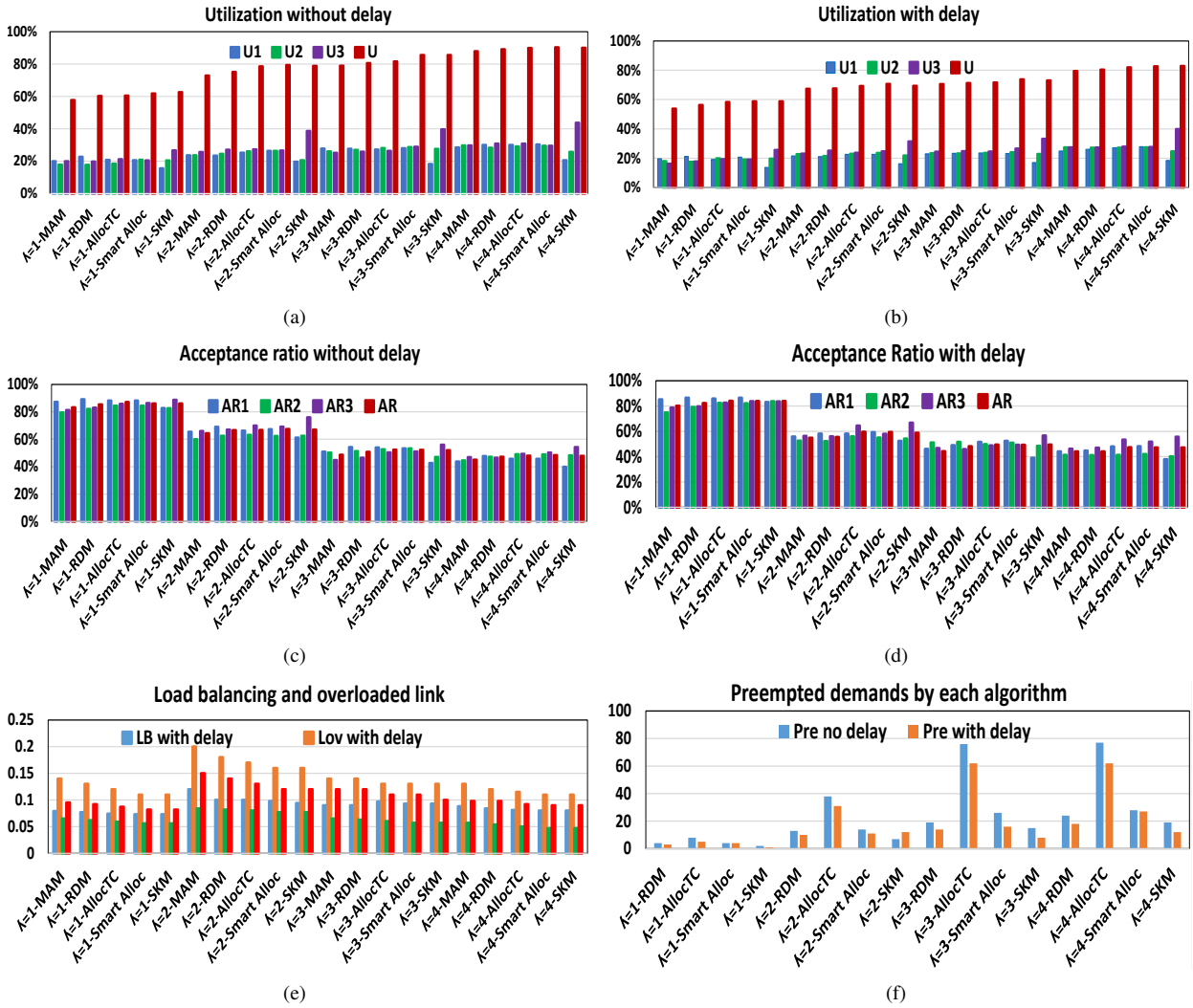

Fig. 4. Overall Performance of SKM with and without E2E delay compared to Smart Alloc, AllocTC, RDM and MAM in scenario 1. Figs. (a) and (b) show the average link utilization and average utilization per slice without delay and with delay, respectively. Figs. (c) and (d) show the average acceptance ratio and average acceptance ratio per slice without delay and with delay, respectively. Fig. (e), shows the load balancing and overloaded link with and without delay, and figure (d), shows the average number of preempted demands with and without delay.

the highest priority slice by 47.79%, 53.14%, 53.14% in terms of $\mathcal{U}_3$ and by 54.28%, 60.95%, 60.95% respectively in terms of $\mathcal{AR}_3$.

Fig. 5(e) illustrate that SKM and AllocTC have a similarly good performance by achieving zero in terms of both $\mathcal{LB}$ and $\mathcal{L}_{ov}$ since all links are fully used in the network. Moreover, RDM and MAM gave the worst performance in terms of $\mathcal{LB}$ and $\mathcal{L}_{ov}$ and resulted in 0.0117 and 0.0474, respectively, where more number of links are not fully used across the network. Further, from the results of Fig. 5(f), SKM outperforms RDM and AllocTC by 657 and 43 respectively in terms of $\mathcal{P}_{re}$ since the load was too low in lower slices so, no need to use the kicking operation.

Impact of delay on the performance of different algorithms: All simulation results showed that impact of delay on resource allocation process was clearly the most significant parameter among all varied metrics while testing SKM. Specifically, referring to Figs. 5(b) and 5(d), SKM's average utilization and average acceptance ratio with delay, were less than when it was not included by 3%, and 5%, respectively. Similar trends can be seen by referring to SKM's results for average $\mathcal{U}_c$, $\mathcal{AR}_c$, $\mathcal{LB}$, $\mathcal{L}_{ov}$, and $\mathcal{P}_{re}$ (see Figs. 5(e) and 5(f)).

C. Scenario 3: Performance considering NSF topology

In this scenario, we assess the impact of NSF topology on the SKM performance against MAM, RDM and AllocTC under different traffic loads and under fixed demands lifetime, in terms of $\mathcal{U}$, $\mathcal{AR}$, $\mathcal{U}_c$, $\mathcal{AR}_c$, $\mathcal{P}_{re}$, $\mathcal{LB}$, and $\mathcal{L}_{ov}$. Moreover, in this scenario we used the same experiments that were considered in the second scenario. Please note that in all experiments, the capacity of each slice along the network is 1050 unit ($\mathcal{RC}_c(l)$ * 21 links = total size of the slice across the network).

Fig. 6 shows the results by each algorithm in terms of $\mathcal{U}$, $\mathcal{AR}$, $\mathcal{U}_c$, $\mathcal{AR}_c$, $\mathcal{P}_{re}$, $\mathcal{LB}$ and $\mathcal{L}_{ov}$ using different traffic load according to experiments 1–3.

Experiment 1, considering high load in lower priority slices: From Figs. 6(a) and 6(c) results, SKM, AllocTC and RDM resulted in 88.93% $\mathcal{U}$ and 41.97% $\mathcal{AR}$ where 1679 de-

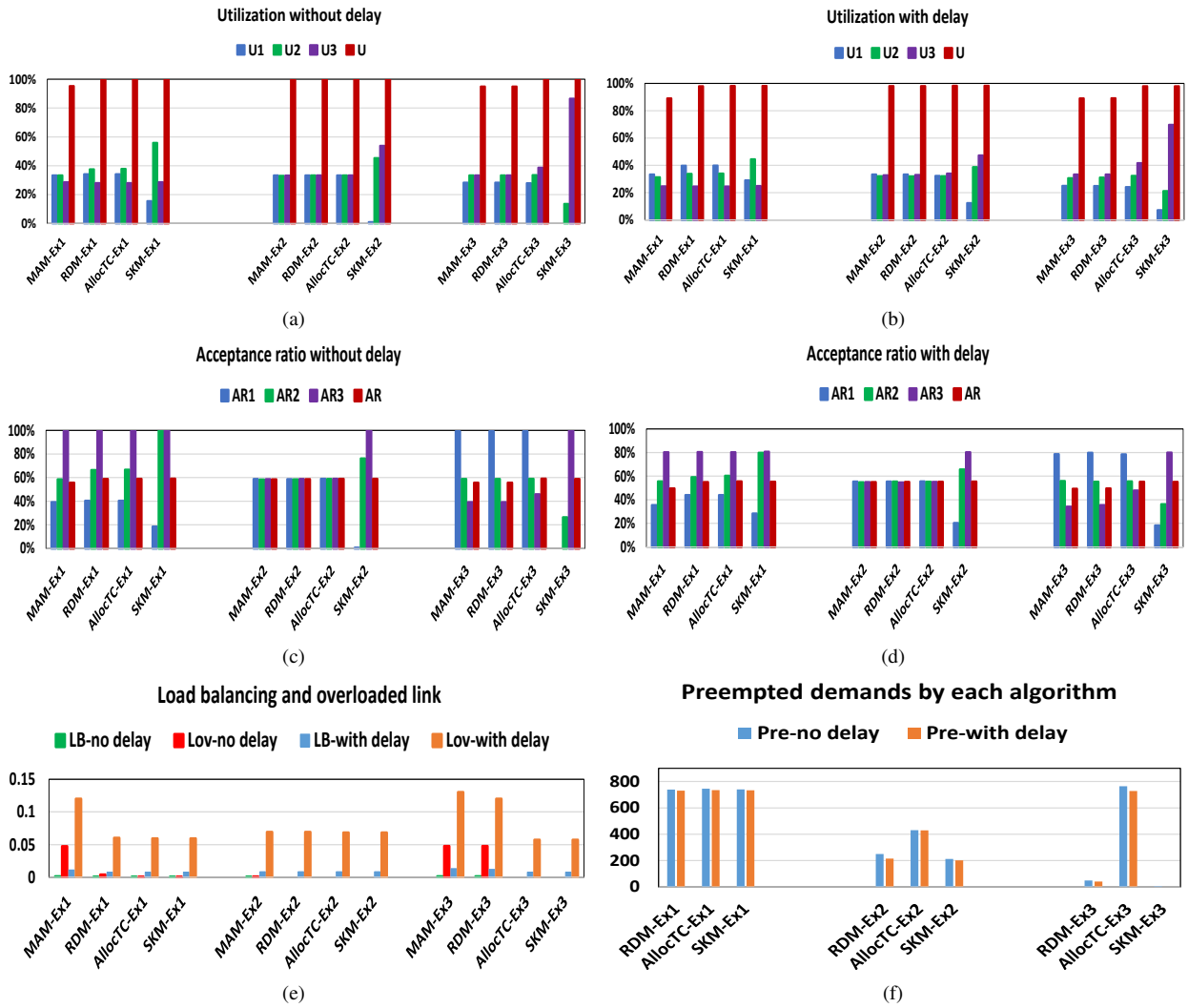

Fig. 5. SKM with and without E2E delay performance compared to MAM, RDM and AllocTC in scenario 2. Figs. (a) and (b) show the average link utilization and average utilization per slice without delay and with delay, respectively. Figs. (c) and (d) show the average acceptance ratio and average acceptance ratio per slice without delay and with delay, respectively. Fig. (e), shows the load balancing and overloaded link with and without delay, and figure (d), shows the average number of preempted demands with and without delay.

mands are accepted from 4000 demands per each unit time. On the other hand, MAM achieved the lowest performance and resulted in 77.24% $\mathcal{U}$ and 37.30% $\mathcal{AR}$ where 1492 demands are accepted from 4000 demands per each unit time. Moreover, SKM, AllocTC, RDM and MAM have simiar behaviour in terms of both $\mathcal{U}_3$ and $\mathcal{AR}_3$ by achieving 23.98% and 73.72%, respectively because the load distributions on slice 3 across the network was lower than its capacity. Furthermore, SKM outperforms AllocTC, RDM and MAM by 20.14%, 20.61%, 24.32% in terms of $\mathcal{U}_2$. Further, in terms of $\mathcal{AR}_c$, SKM, achieved 12.5% for slice 2 more than MAM, RDM and AllocTC by 15.29%, 15.71%, 20.63%, respectively due to kicking operation (see Figs. 6(a) and 6(c)).

Fig. 6(e) illustrate that SKM, AllocTC and RDM have a very close performance in terms of $\mathcal{LB}$ and $\mathcal{L}_{ov}$ resulting in 0.025 and 0.12 respectively, due to the fact that the algorithms have similar utilization performance. On the other hand, MAM gives the lowest performance in terms of $\mathcal{LB}$ and $\mathcal{L}_{ov}$ resulting in 0.028 and 0.17 respectively, where more links are not being fully used across the network. Moreover, Fig. 6(f) reveals that SKM, AllocTC and RDM resulted in 1601, 2012, and 1331 in terms of $\mathcal{P}_{re}$ respectively due to kicking and preemption operations as we explained earlier.

Experiment 2, considering same load in all priority slices: From Figs. 6(a) and 6(c) results, SKM, AllocTC, RDM and MAM resulted in 88.72%, 88.07%, 87.66%, 87% of $\mathcal{U}$ and 40.62%, 40.54%, 40.52%, 40% of $\mathcal{AR}$, respectively since the load was same in all slices. Moreover, SKM outperforms MAM, RDM and AllocTC in the highest priority slice by 30.14% in terms of $\mathcal{U}_3$ and 29.26% in terms of $\mathcal{AR}_3$ due to the kicking operation. Furthermore, Fig. 6(e) shows that the performance of SKM, AllocTC, RDM and MAM are similar in terms of $\mathcal{LB}$ and $\mathcal{L}_{ov}$ resulted in average 0.024 and 0.12, respectively due to the algorithms have similar utilization performance. Further, Fig. 6(f) shows that the SKM outperforms AllocTC and RDM by 298 and 71 in terms of $\mathcal{P}_{re}$, respectively due to kicking operation.

Experiment 3, considering high load in higher priority

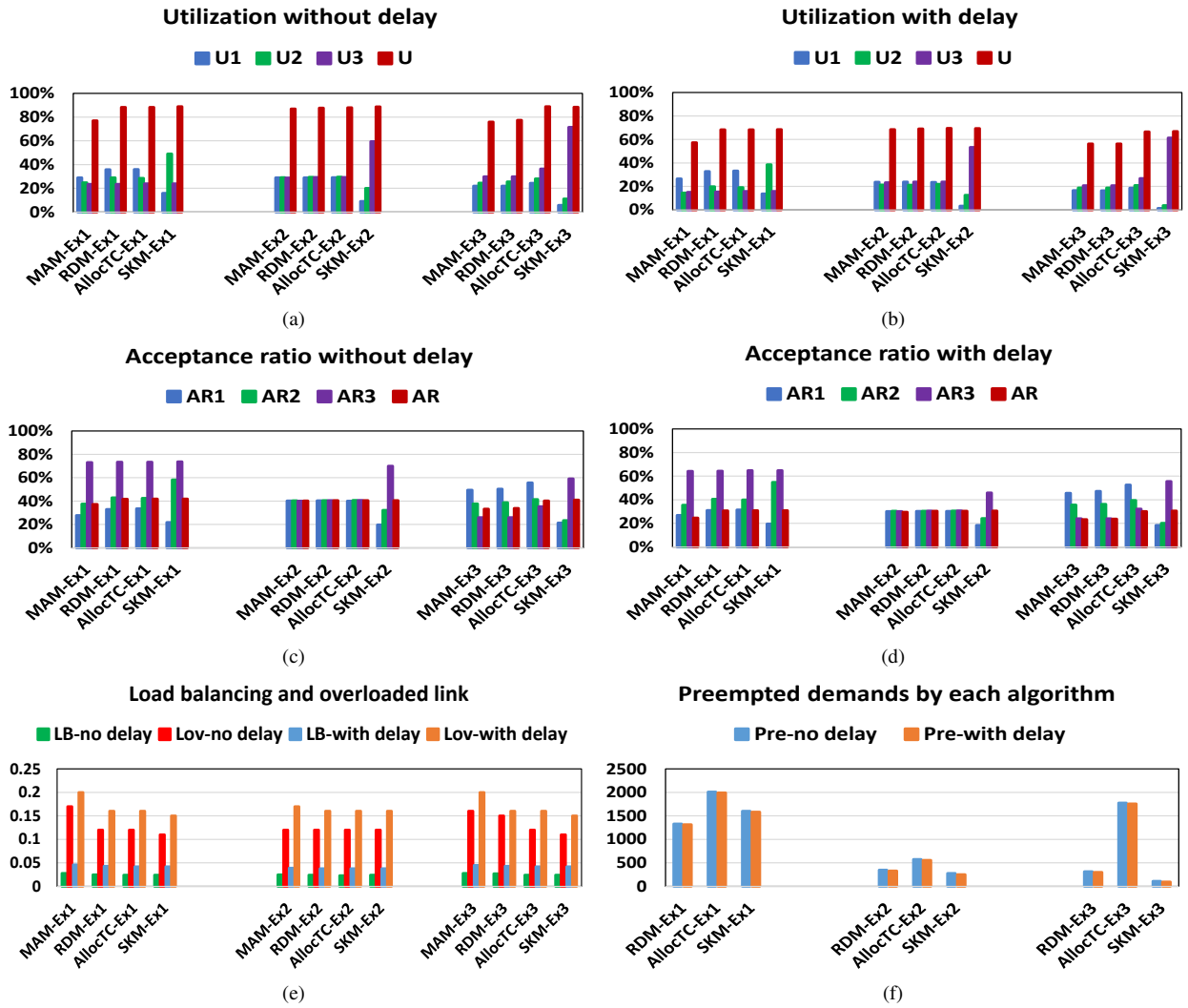

Fig. 6. SKM with and without E2E delay performance compared to MAM, RDM and AllocTC in scenario 3. Figs. (a) and (b) show the average link utilization and average utilization per slice without delay and with delay, respectively. Figs. (c) and (d) show the average acceptance ratio and average acceptance ratio per slice without delay and with delay, respectively. Fig. (e), shows the load balancing and overloaded link with and without delay, and figure d, shows the average number of preempted demands with and without delay.

slices: From Figs. 6(a) and 6(c) results, SKM and AllocTC have similar performance in terms of $\mathcal{U}$ and $\mathcal{AR}$ by achieving 88.65% $\mathcal{U}$ and 41.05% where 1642 demand are accepted from 4000 demand per each unit time. On the other hand, RDM and MAM performance are the lowest one among the four strategies by achieving 77.36%, 76% in terms of $\mathcal{U}$ and 33.90%, 33% respectively in terms of $\mathcal{AR}$. This is because there is no ability to share resources among the slices. From Figs. 6(a) and 6(c) illustrate that the SKM outperforms AllocTC, RDM and MAM in the highest priority slice by 35.17%, 41.71%, 41.71% in terms of $\mathcal{U}_3$ and by 23.8%, and 33.19%, 33.19% respectively in terms of $\mathcal{AR}_3$ (as the expected from the behaviours) due to kicking operation.

From the results of Fig. 6(e) illustrate that SKM and AllocTC have a similarly good performance by achieving 0.025, 0.12 in terms of both $\mathcal{LB}$ and $\mathcal{L}_{ov}$. Moreover, RDM and MAM gave the worst performance in terms of $\mathcal{LB}$ and $\mathcal{L}_{ov}$ and resulted in 0.028 and 0.16, respectively, where more number of links are not fully used across the network. Moreover, Fig. 6(f), SKM outperforms RDM and AllocTC by 1665 and 205 respectively in terms of $\mathcal{P}_{re}$ since the load was was too low on the lower priority slices so, no need to use kicking operation of SKM.

Impact of delay on the performance of different algorithms: All simulation results showed that impact of delay on resource allocation process was clearly the most significant parameter among all varied metrics while testing SKM performance across NSF network. Specifically, referring to Figs. 6(b) and 6(d), SKM's average utilization and average acceptance ratio with delay, were less than when it was not included by 20.42%, and 10.19%, respectively. Similar trends can be seen by referring to SKM's results for average $\mathcal{U}_c$, $\mathcal{AR}_c$, $\mathcal{LB}$, $\mathcal{L}_{ov}$, and $\mathcal{P}_{re}$ (see Figs. 6(f) and 6(e)).

These values confirm the importance of including E2E delay as a main constraint when solving resource allocation problem, as a direct evaluation metric for real world 5G networks.

D. Analysis of Simulation Results

Through the simulation analysis of all the algorithms in the considered scenarios mentioned in Sections V.B.1, V.B.2, and V.C, the following points can be obtained:

1. Addressing delay problems: Incorporation of E2E delay constraint had a vital impact on the resource allocation process, as displayed by lower resource utilization and acceptance ratios across all simulations in the range of 10% and 4% respectively when compared to the cases without delay.
2. Overall online performance of SKM, measured by resource utilization and acceptance ratio were in average in the range of 78.5% and 59.70% respectively, which is similar to the latest referenced online algorithm by Bahnasse et al., (2018). Moreover, SKM outperforms all algorithms in terms of average $\mathcal{U}_3$ and average $\mathcal{AR}_3$ by 11% and 8% respectively, for the different arrival rates (see Figs. 4(a) and 4(c)).
3. The impact of the topology on the SKM performance and other algorithms is discussed below: Our analysis shows that the all algorithms achieve worse performance in terms of links utilization in NSF topology compared to Mesh, even though NSF have more nodes and links. This is because the mesh topology exhibits a low betweenness centrality value compared to NSF, as a result, the mesh topology experiences fewer bottlenecks compared to NSF topology. This is due to the fact that under mesh topology, all nodes are reachable in a single hop from each other, hence, bottlenecks are minimal since most demands are mapped on single edge paths. In addition, the mesh topology has a high closeness implying on average, mapping of demands from source to destination uses fewer links (shortest path length). All the above issues account for the better performance in terms of $\mathcal{AR}$, $\mathcal{U}$, load balancing, delay, resource consumption and number of preempted demands among others for the mesh topology across all the algorithms. In online experiment 2 (same load in all slices), the performance improvement of all algorithms such as SKM under mesh network in terms of $\mathcal{U}$ and $\mathcal{AR}$ is 9.48% for $\mathcal{U}$ and 14.69% for $\mathcal{AR}$ compared to NSF. Note that, we found similar attribute for the other experiments.
4. Usage recommendations: Under online scenario: SKM is a suitable algorithm to be used under different topologies but from our experiments we found that SKM provides high performance in terms of $\mathcal{AR}$, $\mathcal{U}$, $\mathcal{LB}$, $\mathcal{L}_{ov}$, and $\mathcal{P}_{re}$ under topolgies with fewer bottlenecks such as mesh topology irrespective of the load distributions. Moreover, SKM performance gain was more significant with high load in higher priority slices compared to other strategies in terms of accepting higher priority demands. In addition, SKM can reproduce the behaviour of MAM, RDM, and AllocTC in a single model and, as such, generalizes the inherent behaviour of these BAMs in a single implementation in case of unsaturated network.
5. Execution time: We investigated the impact of processing and time costs. The proposed algorithm performance has a sorting step, which needs slightly more memory, but we did not calculate and focus on the cost in terms of memory because our focus was the run time of the algorithms. For example, when the E2E delay was not included, SKM achieved 10 h, 7 min and 56 s as average runtime to assign the demands after running the algorithms 10 times using experiment 3 of scenario 3. RDM and AllocTC have a slightly lower run time complexity (60 and 20 min, respectively) than SKM. Nevertheless, SKM gave very high utilization and acceptance ratio in higher priority slices. Also, when we compare the proposed algorithm with MAM, SKM's run time complexity is approximately 1 h and 45 min more than MAM. Incorporating E2E delay constraints is expected to increase run time due to additional need to search for more paths, however, since we adopted the k-shortest path approach, those additional paths are few in number, the additional run time was insignificant compared to the case without delay constraint.

VI. CONCLUSION

This paper introduced a new algorithm based on SKM that efficiently allocates, manages and controls the slice resources under several constraints for 5G and beyond networks, such as priority, bandwidth and E2E delay in real-time while aiming to maximize the overall resource utilization in the substrate network.

In a practical scenario, the computational needs suggest to be executed inside NFV to provide intelligent decisions regarding admission control, routing path computation and resource allocation with a goal of dynamic resource management and guarantee QoS constraint routing for intelligent network slicing management. Moreover, the algorithm proposed is stricter on priorities and significantly differentiates priorities, especially under congested scenarios to optimize usage and provide high acceptance for users of the higher traffic priority slice, which is critical to ensuring the quality of service.

The experimental results showed that the best available algorithm to handle slices until now, SKM without delay constraint managed to maximize the average resource utilization in the substrate multi-hop network by 20.42% and by 3% in the substrate single-hop network. Additionally, this algorithm achieved up to 100% acceptance ratio in higher priority user slices which can not be achieved by other algorithms in some scenarios. However, when the E2E delay constraint is considered, SKM performance is degraded across all evaluation metrics, suggesting that, introducing E2E delay as the main constraint had a clear impact on the whole resource allocation process, and so, it has to be one of the key metrics when evaluating real-world 5G networks.

As future work, the authors are planning to conduct further study in the context of multi-hop paths, considering an E2E delay for specific 5G applications. Moreover, we aim to implement a heuristic to reduce the computational needs and to provide faster response, which is essential in 5G applications.

REFERENCES

[1] "3GPP. System architecture for the 5G system (5GS). Technical specification (TS) 23.501," 3rd generation partnership project (3GPP), 2020. Version 16.4.0.

[2] "3GPP. Service requirements for the 5G system. Technical specification (TS) 22.261," 3rd generation partnership project (3GPP), 2020. Version 17.2.0.
[3] C. Xu, L. Tao, H. Wu, D. Ye, and G. Zhang, "Multiple constrained routing algorithms in large-scaled software defined networks," arXiv preprint arXiv:1902.10312. Feb. 2019.
[4] S. Tomovic and I. Radusinovic, "Fast and efficient bandwidth-delay constrained routing algorithm for SDN networks," in *Proc. IEEE NetSoft*, 2016, pp. 303–311.
[5] E. Pateromichelakis, K. Samdanis, Q. Wei, and P. Spapis, "Slice-tailored joint path selection & scheduling in mm-wave small cell dense networks," in *Proc. IEEE GLOBECOM*, 2017, pp. 1-6,
[6] W. Feng, Y. Li, D. Jin, L. Su, and S. Chen, "Millimetre-wave backhaul for 5G networks: Challenges and solutions," *Sensors* vol. 16, no. 6, June 2016.
[7] A. El-mekkawi, X. Hesselbach, J. R. Piney, "Squatting and kicking model evaluation for prioritized sliced resource management", *Elsevier Comput. Netw.*, vol. 167, Feb. 2020.
[8] A. El-mekkawi, X. Hesselbach and J. R. Piney, "A Novel Admission Control Scheme for Network Slicing based on Squatting and Kicking Strategies," in *Proc. JITEL*, 2019, pp. 1–8.
[9] A. El-mekkawi, X. Hesselbach and J. R. Piney, "network function virtualization Aware Offline Embedding Problem Using Squatting-Kicking Strategy for Elastic Optical Networks," in *Proc. ICTON*, 2018, pp. 1–10.
[10] A. M. Medhat, G. A. Carella, M. Pauls, and T. Magedanz, "Orchestrating scalable service function chains in a NFV environment," in *Proc. IEEE NetSoft*, 2017, pp. 1–5.
[11] K. Hejja and X. Hesselbach, "Online power aware coordinated virtual network embedding with 5G delay constraint," *Elsevier J. Netw. Comput. Applications*, vol. 124, pp. 121–136, Dec. 2018.
[12] W. Lai and F. L. Faucheur, "Maximum allocation bandwidth constraints model for diffserv-aware MPLS traffic engineering," *RFC 4125*, 2005.
[13] F. L. Faucheur, "Russian dolls bandwidth constraints model for diffserv-aware MPLS traffic engineering," *RFC 4127*, 2005.
[14] R.F. Reale, W. da C. P. Neto, and J.S.B. Martins, "AllocTC-sharing: A new bandwidth allocation model for DS-TE networks," in *Proc. IEEE/IFIP NOMS*, 2011, pp. 1–4.
[15] D. Adami, C. Callegari, S. Giordano, M. Pagano, and M. Toninelli, "G-RDM: A new bandwidth constraints model for DS-TE networks," in *Proc. IEEE GLOBECOM*, 2007, pp. 2472–2476.
[16] C. Tata and M. Kadoch, "CAM: Courteous bandwidth constraints allocation model," in *Proc. IEEE ICT*, 2013, pp. 1–5.
[17] R. Trivisonno, R. Guerzoni, I. Vaishnavi, and A. Frimpong, "Network resource management and QoS in SDN-enabled 5G systems," in *Proc. IEEE GLOBECOM*, 2015, pp. 1–7.
[18] J. Socrates-Dantas *et al*., "Novel differentiated service methodology based on constrained allocation of resources for transparent WDM backbone networks," in *Proc. IEEE SBRC*, 2014, pp. 420–427.
[19] N. Subhashini and A. B. Therese, "User prioritized constraint free dynamic bandwidth allocation algorithm for EPON networks," *Indian J. Sci. Technol.*, vol. 8, no. 33, pp. 1–7, 2015.
[20] W. da C. P. Neto and J. S. B. Martins, "A RDM-like bandwidth management algorithm for Traffic Engineering with DiffServ and MPLS support," in *Proc. IEEE ICC*, 2008.
[21] R. F. Reale, R. Freitas, R. M. da S. Bezerra, and J. S. B. Martins, "G-BAM: A Generalized bandwidth allocation model for IP/MPLS/DS-TE Networks," *International J. Comput. Inf. Syst. Ind. Manag. Appl.*, vol. 1, no. 6, pp. 625–643, Dec. 2014.
[22] A. Bahnasse *et al.*, "Novel SDN architecture for smart MPLS traffic engineering-DiffServ aware management," *Future Generation Computer Systems*, vol. 87, pp. 115–126, Oct. 2018.
[23] B. G. Józsa, Z. Király, G. Magyar, and Á. Szentesi, "An efficient algorithm for global path optimization in MPLS networks," *Optimization and Engineering*, vol. 2, pp. 321–347, Sept. 2001.
[24] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 7, pp. 1228–1234, Sept. 1996.
[25] Y. Yang, J. K. Muppala, and S. T. Chanson, "Quality of service routing algorithms for bandwidth-delay constrained applications," in *Proc. IEEE ICNP*, 2001, pp. 62–70.
[26] S. Chen, "Routing support for providing guaranteed end-to-end quality-of-service," Ph.D. thesis, University of Illinois at Urbana-Champaign, 1999.
[27] J. Y. Yen, "Finding the K-Shortest Loop=less Paths in a Network," *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.
[28] G. Kibalya *et al.*, "A novel dynamic programming inspired algorithm for embedding of virtual networks in future networks," *Computer Networks*, vol. 179, Oct. 2020.
[29] M. R. Garey and D. S. Johnson, "Computers and intractability: A guide to the theory of NP-completeness," San Francisco: freeman, 1979.
[30] A, A. Zoobi, D. Coudert, and N. Nisse, "Space and time trade-off for the k-shortest simple paths problem," in *Proc. SEA*, 2020.
[31] Y. Dinitz, S. Dolev, M. Kumar, "Polynomial time k-shortest multi-criteria prioritized and all-criteria-disjoint paths," arXiv preprint arXiv:2101.11514, 2021.
[32] M. Chowdhury, M. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping." *IEEE/ACM Trans. Netw.* vol. 20, no. 1, pp. 206–219, Feb. 2012.
[33] A. Fischer, J. Botero, M. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, Feb. 2013.

Ahmed El-mekkawi receieved his M.S. degree in Electronics Engineering and Communications from Arab Academy for Science & Technology and Maritime Transport (AASTMT), Alexandria, Egypt in 2016. He previously worked with the Egyptian Interior Ministry between 2007-2016, as the director of the information network at the department of passports and immigration. He is currently working toward his Ph.D. degree in the Network Engineering Department, Universitat Politecnica de Catalunya (UPC), Barcelona, Spain. His current research interests are focused upon networks virtualization, resources management, broadband networks, quality of service, software-defined networks and bandwidth allocation mechanisms.

Xavier Hesselbach (Senior Member, IEEE) Associate Professor at the Department of Network Engineering (Dept. Enginyeria Telematica) at the UPC, and IEEE Senior Member, received the M.S. degree with honors in Telecommunications Engineering in 1994 and the Ph.D. degree with honors in 1999, from the Universitat Politecnica de Catalunya (UPC). In 1993, he joined the Design, modelling and evaluation of broadband networks group in the Network Engineering Department of the UPC. His research interests include networks virtualization, resources management, broadband networks, quality of service and green networking. He has been involved in several national and international projects, and is author in 4 books and more than 100 national and international publications in conferences and journals. In 1994 he received the award from the COIT/AEIT of Spain for the best Master Thesis on Networks and Telecommunication Services. He has participated in the technical program committees of several conferences, he has been the Information Systems and Internet Chair in Infocom 2006, and guest editor of the Ad Hoc Networks Journal and the Journal of Electrical and Computer Engineering. He has taken part in several European and Spanish research projects, such as the EuroNGI/FGI/NF Network of Excellence, COST293, Mantychore and All4Green, being main UPC researcher in the Mantychore and All4Green projects.

JOSE RAMON PINEY received the B.S. in Electronic Engineering and Communications from the Universidad de las Américas, Puebla, México in 1989, the M.S. in electrical engineering from the University of Southern California, Los Angeles in 1993 and the Ph. D. degree in telematics engineering from de Technical University of Catalonia, Barcelona, Spain in 2006. Since 1999 he has been an Associate Professor in the Telematics Engineering, Technical University of Catalonia. His research interests include passive optical networks, software defined networks and bandwidth allocation mechanisms.