Traffic-Aware Virtual Network Embedding with Joint Load Balancing and Datarate Assignment for SDN-Based Networks

Mario Minardi, *Student Member, IEEE,* Thang X. Vu, *Senior Member, IEEE,* Ilora Maity, *Member, IEEE,* Christos Politis, and Symeon Chatzinotas, *Fellow, IEEE*

Abstract—**Network virtualization is one of the key technologies to enable network slicing scenarios for the efficient resource management of beyond 5G and 6G networks. Mapping algorithms, such as Virtual Network Embedding (VNE), have been proposed to manage such a challenging scenario due to heterogeneous traffic requirements. Despite the numerous VNE contributions, the literature is currently lacking VNE implementation platforms, which can exploit real-time traffic statistics and jointly optimize the routing and resource assignment. In general, VNE solutions foresee complex optimizations with worst-case resource dimensioning, without considering the real-time traffic statistics. In this paper, we propose a statistics-collection aware (SCA) link mapping algorithm for VNE, named SCA-VNE. SCA-VNE is formulated as a Mixed Binary Linear Programming (MBLP) problem which jointly minimizes the load balancing and the data rate assignment to each Virtual Network Request (VNR). VNRs are differentiated based on priority level, i.e., tolerated queuing delay and user satisfaction probability. Due to the exponential complexity of SCA-VNE, we propose a relaxed version, named SCA-R, to significantly reduce the computation time. We show via testbed experimental results that, compared to four baseline schemes, the proposed algorithm increases the acceptance ratio up to 11% and drastically minimizes the average queuing delay, in highly-loaded scenarios.**

Index Terms—**Virtual Network Embedding (VNE), Software Defined Networking (SDN), traffic statistics.**

I. Introduction

NETWORK virtualization has opened up promising scenarios and technologies to counteract the increasing traffic demand in beyond 5G networks. Network slicing [1] is one of the most attractive scenarios where the substrate network is optimally shared between different virtual networks, also known as slices, which can offer different Quality of Service (QoS) requirements, e.g., tolerated latency, serving time or data rate requirements. As included in the standardization document [2] from 3GPP, a network slice is defined as ”A logical network that provides specific network capabilities and network characteristics”. From the network virtualization perspective, instead, a virtual network (VN) is the combination of active and passive network elements (network nodes and network links) on top of a substrate network. Clearly, there is a common pattern between the virtual network and a network slice. Both are logical entities which run on top of the physical infrastructure. In the specification [3], the features of the slice are described such as ”Priority, end-to-end latency, service availability requirement”. This brings to the conclusion that the VN, or Virtual Network Request (VNR), is the simplification of a network slice, with the same logical concept. For this reason, we use throughout this contribution the term VNR to refer to a set of interconnected nodes and links with specific network requirements (e.g., datarate). To reap the benefit of network slicing, it is crucial to develop efficient Virtual Network Embedding (VNE) algorithms. Given a defined slice or sub-slice, generally known as Virtual Network Request (VNR), composed of nodes and links, VNE is the optimized process to embed both nodes and links of virtual slices into the substrate networks. As VNE started to be considered more than a decade ago, a multitude of implementations has been already presented in the literature [4], [5]. VNE is traditionally computed considering that the resource requirements are known. While this was a reasonable assumption when the optimization strategies were only at the initial stage, it became an obsolete approach, considering the dynamicity and load of the current information networks.

To fully optimize the embedding algorithms, the traffic from each embedded VNR should be periodically collected to, firstly, keep checking the actual current utilization of the assigned resources and, secondly, use a stochastic traffic description to optimize the assigned resources. From the infrastructure and implementation perspectives, promising technologies such as Software Defined Networking (SDN) [6] can support this method by collecting, in pre-defined intervals of time, the traffic statistics, and describing each VNR’s traffic as a random variable with average and variance. In addition, real-time routing decisions and/or VNE adjustments can be executed by the SDN controller based on the current network utilization. In this paper, we propose a VNE algorithm for an SDN-based network, where an initial embedding is computed in the traditional way for each VNR for the first period of time. After the collection period, the traffic of each VNR is described as a random variable and a joint load-balancing and data rate assignment optimization is performed. The SDN controller supports the VNE algorithm by sending the traffic usage in an almost real-time manner.

This work is supported by Fond National de la Recherche Luxembourg (FNR) via, Industrial Partnership Block Grant (IPBG), ref. FNR/IPBG19/14016225/INSTRUCT, and the ASWELL Project, ref. FNR/C19/ IS/13718904/ASWELL/Chatzinotas

M. Minardi, T. X. Vu, I. Maity and S. Chatzinotas are with the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Luxembourg, Luxembourg (email: {mario.minardi, thang.vu, ilora.maity, symeon.chatzinotas}@uni.lu).

C. Politis is with SES Techom, Betzdorf, Luxembourg (email: christos.politis@ses.com).

A. Related Works

Traditionally, VNE is seen as the combination of two sub-problems, node and link mapping. Detailed surveys summarize the VNE proposed solutions [4], [5], [7]. Due to the NP-Hardness of VNE, the two sub-problems are decoupled and solved separately. The scope of this paper is to investigate one of the two sub-problems, the link mapping. In fact, in this section, some of the main link mapping algorithms, currently proposed in the literature, are analyzed. While mapping a VNR onto the substrate network, any proposed scheme has its own set of features/resources which drives the solution. The choice of the considered resources makes the proposed schemes different one each other. To facilitate the literature review, we consider a subset of features.

The first main trend of VNE was to consider the VNRs as static requirements, both in the network graph and/or the requirements. This was the first approach used when the priority was given to elaborate on different techniques to solve the VNE problem [8]–[10]. Considering VNRs as static resource assignment is very limiting, especially in dynamic and heterogeneous scenarios such as 5G and satellite communications. In fact, several VNE implementations proposed a dynamic VNR description. The dynamicity has been introduced either in the substrate network [11]–[13], in the VNR graph [14], [15] or in the online migration of some node and/or link mappings [16]–[18]. On one side, these works foresaw a real-time reconfigurability of the VNE, which make them a good support to several scenarios, from satcom to 5G well-known use cases, e.g. enhanced Mobile Broadband (eMBB), Ultra Reliable Low Latency Communications (URLLC) and massive Machine Type Communications (mMTC). However, they lack one of the main dynamic features of the current telecom networks, i.e., the time-varying behavior of the traffic. Authors in [19], [20] investigated this aspect. In these works, VNRs were simulated with time-varying requirements which might bring to reconfiguration of some embeddings over time. The periodic re-computations are proposed also in [21], where authors constantly re-compute the VNE for the allocated VNRs to increase the overall utilization of the substrate resources. Although the solutions therein are close to realistic scenarios, they performed the network dimensioning in a rather static way, where requirements of VNR(s) are fulfilled 100% all the time. In practice, this is normally not the case for modern networks [22]. Some works introduced already the idea of directly considering the traffic of each VNR via the probabilistic description [23], which is one of the most realistic approaches to be used in VNE because it follows the nature of the traffic.

Despite that, two topics have not been yet addressed in the literature. The first one is related to the description of the traffic. Authors in [23] described the traffic with a probabilistic approach, however, each VNR's traffic was supposed to be known a priori, via the mean and variance. This assumption is not always realistic since, often, the traffic stochastic process is non-stationary and thus their statistical parameters evolve over time. The most realistic approach would be to observe the traffic for each VNR and extrapolate its mean and variance. The second relevant missing feature is related to the probabilistic approach, when embedding VNRs, introduced by the User Satisfaction Probability (USP). USP is defined throughout this contribution as the probability to satisfy the traffic demand. While, traditionally, VNRs are constantly assigned with what they have required, in the reality, on the opposite, any network operator does not plan to provide maximum service 100% of the time to each user. Indeed, a relevant missing feature in VNE literature is the introduction of a guaranteed USP. This would increase the efficiency of the substrate network resources and, consequently, the acceptance ratio. Finally, VNR might be treated differently for their resource requirements or importance, and, consequently, be assigned with different USPs. Differentiating VNRs is not a new concept since some works already defined VNE with a priority scheme [24]–[26]. However, these works focused only on the priority when sorting the VNRs to be embedded, without considering any optimization technique to be matched with different QoS that VNRs of different priority may have, e.g guaranteed USP, queuing delay. To the best of our knowledge, a link mapping algorithm for VNE which jointly optimizes the load-balancing and the assigned resources with a guaranteed USP, after describing the real-time traffic as a stochastic event, has not been presented before.

B. Motivations

As mentioned in section I-A, the first main motivation of this work comes from the fact that traffic is never static or stationary in real-world scenarios. We aim to simulate, as realistically as possible, a scenario where several and different QoS VNRs send varying traffic over the network and to run the VNE algorithm in a realistic and dynamic context.

Secondly, as the infrastructure provider aims at optimizing its operation and profit, while guaranteeing the requirements of each VNR, real-time traffic monitoring is crucial. This approach requires a flexible network management, which can differentiate VNRs, monitor traffic from any mapped VNR and real-time change routing decisions based on internal policies, if needed. SDN is the promising candidate to provide the VNE with these features and improve the overall performances. In fact, SDN controller(s) can efficiently support VNE algorithms to manage the incoming traffic by describing the current utilization for each VNR and almost instantaneously (physical transmission time, between SDN controller and nodes, delays the process) translating the mapping decisions, coming from VNE, into forwarding rules, to let the traffic flow into the network.

In general, it is not sufficient to get a precise and realistic description of the traffic, for every VNR, but it is also important to differentiate VNRs, when providing them with the required resources, with a stochastic approach of USP. In fact, considering different priority VNRs, not only should be applied as the order which they are embedded in, as proposed in a few works in the literature, but also in the USP, which is likely to be different depending on whether the VNR has a high priority, e.g., emergency services, or lower priority, e.g., sensor networks, earth observations.

This probabilistic approach reduces the assigned data rate, while keeping the USP under control. Some techniques to facilitate the embedding of high priority VNRs, especially in congested scenarios, such as the congestion ratio [27] are also introduced to increase the probability of having available capacity for higher priority VNRs. Finally, we specify that, while normally VNE includes both node and link mapping, in this contribution we prefer to focus on link mapping and consider the node mapping already computed. Different reasons drove this choice. In the literature there are several near-optimal solutions which compute node and link mapping in a coordinated manner. However, these solutions are time-consuming. In fact, simpler solutions, where node and link mapping are computed sequentially and independently, are typically proposed to speed up the computation process. As we intend to provide a solution computed in low time due to the traffic requirements, we followed the same approach. Consequently, since the focus of our algorithm is on traffic analysis and real-time link monitoring, it is convenient to prioritize the link mapping. We noticed that the analysis was complex and to simplify the readiness of our paper, we decided to keep the node mapping known a priori.

C. Contributions

Considering the highlighted trends of VNE's literature in Sec. I-A and the motivations for this work in Sec. I-B, our contribution is the following:

- As the VNE literature lacks of considering USP as one of the main KPI, we propose a SCA-VNE which minimizes the average link capacity utilization, given a minimum guaranteed USP. We consider different priority classes of the traffic and evaluate the performance in terms of acceptance ratio, average queuing delay and the USP, for each priority class.
- To tackle the exponential computation complexity of the formulated MBLP problem, we propose a relaxed version, SCA-R, which relaxes the binary constraint of the flow embedding variables with a penalty function to balance the convergence-performance trade-off. We show that SCA-R is a more scalable solution, with respect to SCA-VNE, since the time complexity of SCA-R grows significantly slower than the SCA-VNE's one, as the number of network nodes increases.
- Finally, the proposed VNE algorithm is validated via testbed experimental results. We build an emulated SDN testbed to collect traffic statistics from each VNR in real-time scenarios. Based on the current utilization of each VNR, SCA-VNE releases resources that are not utilized and assigns them to other incoming VNR. Both simulation and implementation results show that SCA-R outperforms existing solutions by reducing the acceptance ratio, in highly-loaded scenarios, and, by considerably minimizing the average queuing delay, especially for higher priority VNRs.

This paper is organized as follows. Section II formulates the problem and Section III-D describes the SCA-VNE algorithm and the relaxed version SCA-R. Some initial simulations are presented in Section IV and the performance evaluation, as well as the testbed description/results, are presented in Section V. Finally, Section VI concludes the paper.

Table I: Variables and parameters in eq. (1) - (9)

$\boldsymbol{z_{uv}}$	binary variable to indicate if the VNR is embedded in $uv \in E_s$
$\boldsymbol{x}$	data rate
E_S	set of substrate links
$c(uv)$	availability capacity on the link uv
R	congestion ratio
N_S	set of substrate nodes
P_g	minimum USP to be guaranteed
μ, σ^2	average and variance of simulated traffic

II. Network Model and Problem description

A. Network Model

We model the substrate network as a directed weighted graph $G_s = (N_s, E_s)$, where N_s is the set of substrate nodes and E_s is the set of substrate edges. Given u and v a pair of substrate nodes in N_s, we define $c(uv)$ as the available capacity of the link.

Every VNR is modelled as a directed graph: $G_v = (N_v, E_v)$, with N_v the set of virtual nodes and E_v the set of virtual edges. Each VNR is described via a fixed data rate b, a priority level p, a lifetime L time slots and a start time slot t_a. Given the defined priority level, a minimum USP P_g is guaranteed. The VNR will leave the network at $t_a + L$. The VNR generation follows a Poisson distribution [14] with average arrival rate λ VNRs per time slot. VNRs are randomly generated as end-to-end connections between any pair of substrate nodes in the set N_s, given m the source node and d the destination node. We assume that the topology of VNRs does not vary over the considered time frame. For each VNR, we define a counter t_{count} which keeps track of whether the VNR has registered a very low value of traffic and possibly the lifetime. For all VNRs, a common collection period $t_{collect}$ is defined. After the collection period, the traffic of each VNR is modelled as a random variable X, with mean value μ and variance σ^2 (we omit the VNR subscript index for ease of presentation). The simulation time is subdivided into time slots of duration t_{slot}. In addition, let us denote θ as the minimum required serving percentage of the VNR and t_{exp} as the number of time slots the VNR experiences traffic smaller than θ. Finally, we define R_h, R_m and R_l as the congestion ratios for the high, medium and low priority VNRs, respectively.

B. Problem Formulation

In the following, the objective and constraints of the problem are presented. The objective function (1) jointly optimizes the load-balancing, which aims at spreading the traffic in the substrate network as more as possible, while minimizing the assigned data rate. To simplify the reading, we summarized the variables and parameters in Table I. The variables are in bold.

The initial joint objective can be written as:

$$\min_{\boldsymbol{z_{uv}},\boldsymbol{x}} \frac{1}{|E_s|} \sum_{uv\in E_s} \frac{\boldsymbol{z_{uv}}\cdot \boldsymbol{x}}{c(uv)+\epsilon}, \tag{1}$$

with ϵ a very small number. Given the following constraints:

$$\boldsymbol{z_{uv}}\cdot \boldsymbol{x} \le R\cdot c(uv),\ \ \forall (uv)\in E_s, \tag{2}$$

$$\sum_{w\in N_s} \boldsymbol{z_{mw}} - \sum_{w\in N_s} \boldsymbol{z_{wm}} = 1, \tag{3}$$

$$\sum_{w\in N_s} \boldsymbol{z_{dw}} - \sum_{w\in N_s} \boldsymbol{z_{wd}} = -1, \tag{4}$$

$$\sum_{v\in N_s} \boldsymbol{z_{vu}} - \sum_{v\in N_s} \boldsymbol{z_{uv}} = 0,\ \ \forall u\in N_s\setminus\{m,d\}, \tag{5}$$

$$\boldsymbol{z_{uv}}\in\{0,1\},\ \ \forall (uv)\in E_s, \tag{6}$$

$$\boldsymbol{z_{uv}}+\boldsymbol{z_{vu}}\le 1,\ \ \forall (uv)\in E_s, \tag{7}$$

$$\frac{1}{2}[1+erf(\frac{\boldsymbol{x}-\mu}{\sigma\sqrt{2}})]\ge P_g, \tag{8}$$

$$\mu < \boldsymbol{x} < \mu+\sigma\sqrt{2}. \tag{9}$$

In (2), for each substrate link uv, the embedded data rate is upper-bounded by a portion of the residual substrate capacity. The value $R\in\{R_h,R_m,R_l\}$ is the congestion ratio ([27]). Constraints (3) and (4) ensure that, given the source and destination substrate nodes, the outgoing flow from the source node m and the total incoming flow in the destination node d is equal to the required data rate, with a negative sign for the latter one. For all intermediate nodes, (5) states that the sum of incoming flows has to equal the sum of outgoing flows (flow's conservation law). Equation (6) states the integrity constraint for the flow variable and constraint (7) guarantees the absence of loops. Unlike the traditional VNE approaches, we additionally introduce constraint (8) as a new measure to control the actual USP. In particular, (8) states that the probability of the actual traffic is lower than the assigned optimized data rate x (i.e., integration of the probability density function of the random variable X, over the interval [0, x]) should be greater or equal to the minimum guaranteed USP. Since the random variable x is within the erf function, the explicit formula of constraint (8) is obtained under the condition that the argument of the erf function is in the interval $[-1,1]$. In addition, it should be convex, so the interval is further restricted to $[0,1]$. This is why we introduce the additional constraint (9).

III. Statistics collection-aware VNE

Problem (1) is a mixed binary non-linear programming problem due to the binary VNR assignment variables and the non-convex objective function. In this section, we will first transform the original problem into a difference-of-convex (DC) form, which will be solved via an iterative algorithm. Subsequently, we relax the optimization problem presented in DC form using a proper penalty function that removes the binary constraint (6) and increases the scalability of the solution.

A. Difference of convex

The expression presented in (1) and in constraint (2) is not convex due to the multiplication of the flow variable times the assigned data rate variable. We will transform them into a preferable format which can be efficiently solved by DC. By using the equivalent representation of a product:

$$2xz=(x+z)^2-(x^2+z^2), \tag{10}$$

the objective function can be subdivided into two terms. The first one is already convex, while the second one is concave. We use the first order of approximation to iteratively solve the concave term as:

$$(x^2+z^2)=(2xx_i^2-x_i^2)+(2zz_i^2-z_i^2), \tag{11}$$

where x_i and z_i are the feasible values at the i-th iteration.

Thus, the objective function at the i-th iteration can be written as:

$$\min_{z_{uv},x} \frac{1}{|E_s|} \sum_{uv\in E_s} \frac{(x+z_{uv})^2-(2xx_i-x_i^2)-(2zz_i-z_i^2)}{c(uv)+\epsilon}. \tag{12}$$

Solving (12) with respect to constraints (2)-(9) provides link mapping from source m to destination d as specified in the decision variable vector z.

B. SCA-VNE: Statistics Collection-Aware VNE algorithm

The pseudo-code of SCA-VNE is introduced in Algorithm 1. As mentioned earlier, the simulation is subdivided into time slots. At each time slot, a set of operations is performed, in the order as explained in Algorithm 1. Initially, in case of expired VNRs, the substrate resources will be released. For any mapped VNRs, the SDN controller checks and saves the current utilization. If the analyzed VNR's traffic is below θ and this has been registered for t_{exp} time slots, then the VNR is considered as expired. Otherwise, the counter for that VNR is increased by 1. In addition, if already t_{coll} samples have been obtained for the selected VNR, the VNR is included in the vector "to jointly optimize". Then, the new arrived VNR(s), if any, are initially mapped with DiVNE [8], restricted to the load balancing objective. We underline that we can not apply our model as initial mapping of any arrived VNR because the model needs to have a collected statistics. This is why we use a traditional VNE algorithm, such as DiVNE, which is purely based on initial demanded resources. If the solution is found, the current substrate network resources are updated, and the traffic of the VNR starts to be transmitted. Otherwise, the VNR is inserted into the failed VNR(s) vector. This is done to initially assign a path, where the traffic can start to flow. The jointly optimization is performed later on, when t_{coll} samples of traffic have been collected, and, therefore, the actual value of traffic can be used to jointly optimize the assignment. This is what happens in the following for cycle, where, for all the VNRs which satisfy the condition of enough collected samples, the proposed joint optimization algorithm is proposed. All these described processes take a certain interval

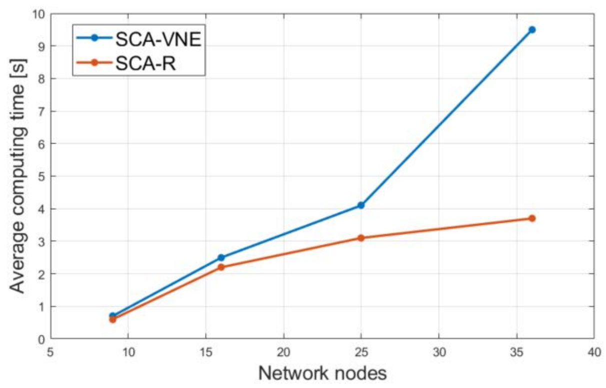


Figure 1: Comparison of average computing time vs network nodes between integer formulation (SCA-VNE) and relaxed one (SCA-R)

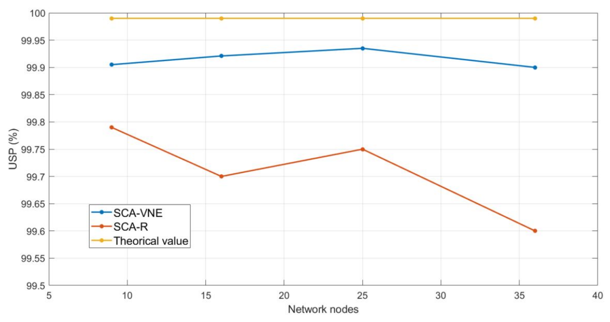


Figure 2: Comparison of average USP vs network nodes between integer formulation (SCA-VNE) and relaxed one (SCA-R)

of time t_{comp}. For each time slot, given the fixed duration to t_{slot} seconds, in the interval $t_{slot} - t_{comp}$, the system will try to re-embed the failed VNRs, starting, once again, from the highest priority ones. This is important especially in congested scenarios, to try to minimize the delay experienced by the highest priority VNRs.

C. SCA-R: LP Relaxation of the SCA-VNE Formulation

In the previous subsection, we have presented the SCA-VNE formulation as a joint optimization and we have also showed that a non convex solution was transformed in a convex solution, using an iterative process. However, the exponential complexity of MBLP formulation limits the scalability of SCA-VNE. Figure 1 shows the exponential growth of the average computing time of SCA-VNE with the increasing number of the substrate nodes. This is due to the binary constraint (6) of the flow variable. For this reason, we propose a relaxed version of SCA-VNE, named SCA-R algorithm. SCA-R is obtained by the relaxation of the binary flow variable in constraint (6) to continuous values. To do that, we introduce the penalty function

$$P(\bar{z}) = \sum_{uv \in E_s} z_{uv} - z_{uv}^2, \tag{13}$$

Algorithm 1: SCA-VNE Algorithm

```
Initialization
Substrate network $G_s, E_s$, $c(uv)$ and $p(uv)$ $\forall uv \in E_s$;
VNR parameters (required physical resources, arrival
  distribution, end-to-end nodes);
for $t \in [t_0, t_{sim}]$ do
    save current status of substrate network;
    for expired VNR(s) do
        release assigned resources;
    end
    for any mapped VNR(s) do
        check data rate usage with SDN traffic
          statistics collection;
        save current utilization;
        if $x < \frac{\theta * \mu}{100}$ and $t_{count} = t_{exp}$ then
            add to expired VNR(s);
        else
            increase $t_{count}$ by 1;
        end
        if $t = t_a + t_{coll}$ then
            add in to jointly optimize;
        end
    end
    for any arrived VNR(s) do
        solve load-balancing [8] (starting from the
          highest priority VNR(s));
        if successful then
            update current network resources;
            send link mapping and demanded data rate
              to SDN controller;
        else
            add to failed VNR(s)
        end
    end
    for VNR(s) in to jointly optimize do
        solve (12) s.t. (2) - (9);
        if successful then
            update substrate residual capacities with
              optimized data rate $x$;
            send $x$ to SDN controller;
        else
            add to failed VNR(s)
        end
    end
    for any failed VNR(s) do
        solve load-balancing as in [8] starting from the
          highest priority VNR(s);
        if successful then
            update current network resources;
            send mapping and assigned data rate to
              SDN controller;
        else
            try the subsequent VNR (if any)
        end
    end
end
```

to penalize the values of the flow variable, far from the extremes 0 and 1. The algorithm runs an iterative process, where f_i is the solution at the i-th iteration. The process iterates until $|f^{i-1} - f^i| < \varepsilon$, with ε a very small number.

Since (13) is not convex, we introduce the first order approximation which results as

$$P(\bar{\boldsymbol{z}}) = \sum_{uv \in E_s} z_{uv} + \bar{z}_{uv}^2 - 2z_{uv}\bar{z}_{uv}, \tag{14}$$

being $\bar{z}_{uv}$ the solution at the previous iteration. With the addition of $P(\bar{\boldsymbol{z}})$ to the objective function (12) of SCA-VNE, the final objective of SCA-R at the i-th iteration can be written as:

$$\min_{z_{uv},x} \Big(\frac{1}{|E_s|} \sum_{uv \in E_s} \frac{(x + z_{uv})^2 - (2x\bar{x} - \bar{x}^2) - (2z_{uv}\bar{z}_{uv} - \bar{z}_{uv}^2)}{c(uv) + \epsilon} + \gamma \cdot P(\bar{\boldsymbol{z}}) \Big), \text{ s.t. } (2)-(5), (7)-(9) \tag{15}$$

with $\bar{x}$ the solution for x at the previous iteration and $\gamma > 0$ as the penalty weight parameter. Each iteration is solved by cvx. We underline that, while the relaxed SCA-R has a relevant impact on the computing time, the difference in USP performance, as highlighted in Figure 2, is not dependent on the number of substrate nodes. Accordingly, the relaxed version SCA-R worsens the average USP of $\sim 1\%$ with respect to the integer version SCA-VNE.

D. Complexity Analysis

In this subsection, we analyze the worst-case scenario for the solution of (15). For each iteration, the total number of scalar variables is given by the sum of the cardinality of the flow variable z_{uv}, i.e. $|E_s|$, and the data rate variable x, i.e., 1. The total number of constraints is given by $(|E_s| + |N_s| + |E_s| + 2)$. Given N_i as the number of iterations to reach a locally optimal solution, the global complexity to solve (15) is defined as $N_i((|E_s| + 1)^3(|E_s| + |N_s| + |E_s| + 2))$.

IV. Initial Simulations

In this section, we describe the simulation setup and we provide several performance evaluations to initially set problem parameters such as interval time slot duration and congestion ratio values for different priority VNR. Some trade-offs are discussed.

A. Simulation setup

The substrate network is a grid of 16 nodes (SDN-enabled switches) and we simulate VNRs of different priorities. While considering a static network might seem a strong assumption, we clarify that, due to the complexity of the problem and the testbed, our intent in this work is to present the optimization and the logic behind the proposed mapping algorithm. Considering dynamic links would have further complicated the equations with the risk of confusing the reader, as this is a novel approach. However, we consider the interval time slot duration as low as 5-15 s. to reduce the impact of topology changes on the assumption of static network. As highlighted in Figure 3, among the services of beyond 5G and 6G networks are expected to be very heterogeneous. The same integrated non-terrestrial/terrestrial substrate network might accommodate emergency services (from IoT devices to Emergency vehicles), broadband and broadcasting accesses or very low priority and resource demanding services such as earth observations. This intends to motivate the reasons to consider several priority services with different QoS requirements.

Table II: Simulation setup parameters

Parameter	Value
Simulation time	1 hour
Initial substrate capacity c(uv)	10000 bps
Required data rate b	$\in$ [1000, 4000] bps
Lifetime L	$\in$ [100,150] time slots

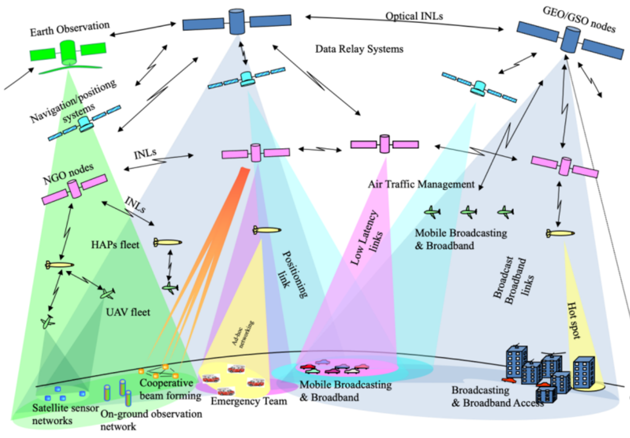


Figure 3: 6G Integrated Non-Terrestrial Networks [28]

The VNRs are randomly generated between any pair of hosts with equal random probability (33.3 %) to be either high, medium or low priority requests. We initialize $\lambda = 1$ VNR/time slot. Note that we are generating high values of resource requirements and for a long period of time to congest the network after a few tens of time slot, considering that the congested scenarios are where the proposed algorithm shows the higher gain. VNRs are differentiated on 3 different priority levels as follows:

- High priority e.g. the emergency connections;
- Medium priority e.g. the backhaul link for broadband connectivity;
- Low priority e.g. the earth observation missions.

For each level, we define $P_g = 99.99\%, 97\%$ and 90% for high, medium and low priority VNRs, respectively. We summarize the remaining parameters in Table II.

B. Duration of time slot

In this subsection, we analyze the impact of the single time slot duration t_{slot}. Unlike other parameters such as congestion ratio or USP, t_{slot} affects not only the acceptance probability but, mainly, the average queuing delay. In fact, in each time slot, as described in Algorithm 1, not only the embedding for the new arrived VNRs is computed, but

Table III: interval time slot duration impact on average queuing delay

t_{slot}	High	Medium	Low
5 s.	2.79 s.	4.51 s.	4.61 s.
10 s.	2.9 s.	5.12 s.	4.99 s.
15 s.	3.549 s.	4.048 s.	5.4198 s.

additional operations are performed, such as checking whether any VNR has expired, jointly optimizing any mapped VNR which traffic has been sufficiently collected for and trying to run again algorithm [29] for those VNRs which have failed the previous time slots. This latter operation is performed starting from the highest priority VNRs until the duration of the time slot is over.

Clearly, on one side, the longer the time slot, the more the probability to try to embed more previously failed VNRs (if any). However, on the other side, if one VNR fails to be mapped, it will have to wait the following time slot to try to be mapped again. This corresponds to a waiting time of at least t_{slot}. If the network load is low, this effect is not relevant because the system will usually manage to map all VNRs as soon as they arrive or few failed VNRs will be present each time slot. However, this might not be obvious when the load increases because the probability of having failed VNRs to be re-mapped increases as well.

To understand a reasonable value for t_{slot}, the average queuing delay should be considered. For the analysis we have taken three possible and meaningful interval values, 5, 10 and 15 seconds. In Table III, we can notice that, as expected, independently on the priority, the higher the t_{slot}, the higher the average queuing delay. Note that the values are averaged over the accepted VNRs but, in general, the low priority VNRs are the most affected by a lower t_{slot} because they might suffer large delays.

After this analysis, $t_{slot} = 10s$. is taken as the most reasonable choice for the following simulations.

C. Acceptance ratio vs time

The acceptance ratio evaluation is the performance used to analyze and obtain reasonable values of the parameters t_{coll} and t_{exp}. To better understand the value and the impact of these two parameters, we present in Figure 4 all steps each VNR goes through. Whenever a new VNR arrives, certain requirements are defined. In a real scenario, this can be seen as the resource requirements that a VNR has paid for. For the link mapping, the resource requirements are considered as the data rate and the USP. Initially, like the traditional approach, the VNR goes through an optimization algorithm where the path is optimized, given the required data rate as static. At this point, we solve the DiVNE [8].

If the mapping is successful, the SDN controller starts to collect data from the VNR. After t_{coll} collected samples, the proposed joint optimized SCA-R, as mentioned in (15), is solved. The SCA-R is solved every collection window, i.e. every t_{coll} time slots. When the lifetime of the VNRs expires, the assigned substrate resources will be released. This is done because in the reality we know that the traffic is never static.

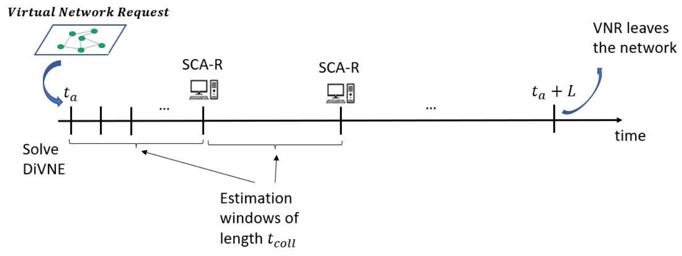


Figure 4: Lifetime of a VNR

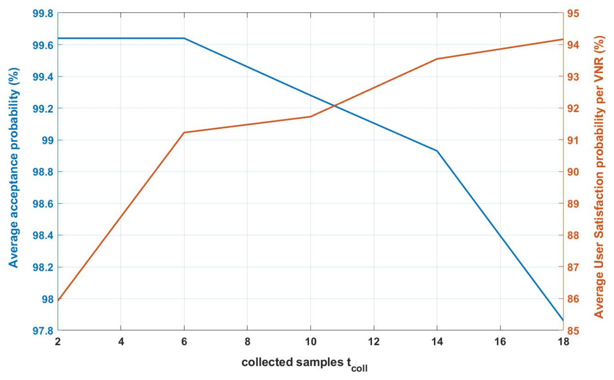


Figure 5: Average acceptance probability and Average USP vs collection window's length

In fact, after a set of collected samples, the SDN controller describes the traffic from each VNR, given the mean and the variance values. The number of the collected samples t_{coll} drives the first trade-off. The more the collected samples, the more precise the traffic estimation (higher USP) but the more the time where the VNR is assigned with a high value of data rate, which corresponds to less available resources for other VNRs. On the opposite, the sooner we get the traffic estimation, the sooner the joint optimization algorithm runs avoiding to waste substrate capacity, bringing to higher acceptance ratio.

Acceptance ratio, or acceptance probability, is a well-known performance analysis in VNE literature [8]–[12], defined as the ratio between the mapped VNRs and the arrived ones. It can be defined at each time, in case the simulation is subdivided into time slots and, by definition, is a number in the range $[0, 1]$. The second analysis, which drives the above-mentioned trade-off, is the USP. In fact, the lower t_{coll}, the worse the traffic estimation and the lower the USP. This effect is emphasized also by the fact that, during the interval of time where the traffic is being collected, the VNR is assigned with the maximum data rate, which means USP = 1 (from VNE perspective) in this scenario (with the assumption that there are no peaks of traffic higher than the required data rate). For this scenario, we have selected only one random type of VNR (medium) to avoid any worsening of performance due to mixing of priority mapping.

In Figure 5, the impact of t_{coll} on the trade-off between average acceptance probability is shown. In fact, on the x-axis, t_{coll} values are highlighted. On the left y-axis, we measure the average acceptance probability. The more the collected

Table IV: Congestion ratio impact on acceptance probability

Congestion ratio scenarios	High	Medium	Low
$R_h = R_m = R_l = 0.945$	91.4 %	97.75 %	98.99 %
$R_h = 1, R_m = 0.98, R_l = 0.90$	95.65 %	94.19 %	92.23 %
$R_h = 1, R_m = 0.95, R_l = 0.90$	90.62 %	96.67%	83.87%
$R_h = 1, R_m = 0.95, R_l = 0.88$	96.88 %	100 %	93.55%
$R_h = 1, R_m = 0.92, R_l = 0.90$	96.88%	100 %	87.10%
$R_h = 1, R_m = 0.90, R_l = 0.85$	97.8 %	94.12 %	86.36 %

samples, the more the time spent with higher capacity consumption, so less available resources and therefore worse acceptance probability. On the other side, the more precise the traffic description, i.e. the system can find a more accurate optimized value, the higher the USP (right y-axis). This effect is emphasized also by the fact that for the interval of time where the traffic is being collected, the VNR is assigned with the maximum data rate, which means USP = 1 in this scenario (with the assumption that there are no peaks of traffic higher than the required data rate). This is the reason why the curve has a steep decrease.

After the first analysis, it looks like it is better to have $t_{coll} \geq 16$ time slots to avoid having low USP values.

It is worth underlining that the t_{coll} values that we have found is strictly dependent also on the type of simulated traffic. In this case, we are considering an easier scenario, where the VNR generates traffic described as a normal variable with average and variance equal to a percentage of the peak data rate (e.g. 90% of the peak data rate and 4%, respectively). We underline the fact that, depending on this choice, different results will be obtained. For example, if the traffic varies more, longer estimation times will be needed to collect reliable traffic description.

D. Congestion ratio impact and analysis

In this subsection, we analyze the impact of congestion ratio on the acceptance probability and we aim to set a reasonable value for congestion ratios for the three different priority values. As previously mentioned, congestion ratio is not a new concept. In fact, it has been introduced by authors in [27]. If, on one side, congestion ratio is efficient to spread the traffic over the network and to leave available substrate capacity in case of need, on the other side, when the congestion ratio is too low, the acceptance probability is drastically reduced, hence, more free capacity is needed. Authors of CEVNE [27] have found out that a reasonable value for congestion ratio would be 0.945 in the presented scenario. This means that each substrate link can be filled up to 94.5 % of its capacity.

In this context, we apply the congestion ratio to the three types of VNRs. Initially, we use the same congestion ratio proposed in CEVNE, i.e. 0.945.

Results in Table IV show that, considering the same congestion ratio for any type of VNRs is not efficient. In fact, the higher the priority of VNR (which corresponds to a higher USP), the lower the acceptance probability. Thus, in this scenario, the choice proposed for CEVNE is not efficient because, being the VNRs treated equally, high priority VNRs will be less likely to be accepted than the others, due to more need of capacity since the guaranteed USP is close to 100%. Clearly, the disadvantage introduced by the USP should be counteracted by an unbalanced congestion ratios, in favor of higher priority VNRs. It is worth underlining that these considerations hold in scenarios where the traffic demands have the same order of magnitude among the three types of requests (i.e. the traffic demand is randomly generated within the same range of values). In fact, in case of unbalanced situations, e.g. low priority VNRs with very low data rate, they would be even more likely to be mapped with respect to highly demanding requests.

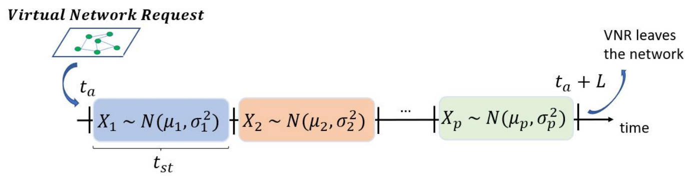

Virtual Network Request
VNR leaves the network
t_a
$t_a + L$
$X_1 \sim N(\mu_1, \sigma_1^2)$
$X_2 \sim N(\mu_2, \sigma_2^2)$
...
$X_p \sim N(\mu_p, \sigma_p^2)$
time
t_{st}

Figure 6: Non-stationary traffic

We compare different combinations for R_h, R_m and R_l. The choice usually depends on the service level agreements (SLAs) and the load of traffic. Independently from the type of VNR selected, the first consideration is that the lower the congestion ratio, the lower the acceptance probability. In addition, using different values of congestion ratio depending on the priority, increases the probability of available capacity for higher priority VNRs. So the choice is whether to decide to penalize all VNRs at the same way, or to give importance for example to high and medium priority and accept a considerable lower values of acceptance probability for low priority VNRs. For example, if this is the case, considering $R_h = 1, R_m = 0.95, R_l = 0.88$ would make more sense. On the opposite, if we want to increase the chances for higher priority VNRs and decreasing the acceptance probability for low priority VNR is still within the SLAs, $R_h = 1, R_m = 0.90, R_l = 0.85$ would be a valid option, which offers a low acceptance probability (86.36%) for low priority but the maximum for high priority (97.8%). For the following simulations, we take this latter option.

E. Average USP for non-stationary traffic

For the sake of simplicity, in the above initial analyses, we have considered a stationary traffic, i.e. static average and variance over time. However, given that the statistical distribution is by nature non-stationary, in this subsection we aim to test our algorithm in different scenarios of non-stationarity.

Let us assume that the VNR's traffic can be subdivided into epochs of length t_{st} where the statistic can be described with average μ and variance σ^2. As shown in Figure 6, let μ_i and σ_i^2 be the average and variance of the traffic during epoch i, respectively. We aim to test our algorithm with varying ratio $\frac{t_{coll}}{t_{st}}$, which considerably affects the final USP. We recall that t_{coll} is the number of collected samples to obtain the average and variance. Intuitively, the higher $\frac{t_{coll}}{t_{st}}$, the less prompt the reaction of the algorithm is to quick changes of traffic statistics. On the opposite, the lower t_{coll} with respect to t_{st}, the more reactive the algorithm is. However, if t_{coll}

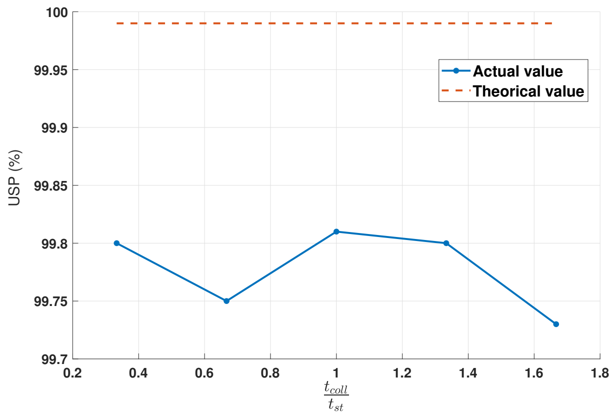
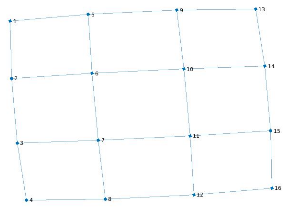
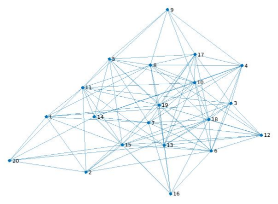


Figure 7: USP vs t_{coll}/t_{st}



(a) Scenario A



(b) Scenario B

Figure 8: Comparison of network models

is too small, the number of samples might not be enough to obtain a reliable statistical description. We run this experiment only for one type of VNR, high priority, because the same applies to the other types of VNRs. Figure 7 shows the impact of $\frac{t_{coll}}{t_{st}}$ on the USP. The theorical value represents the ideal case, i.e. when the estimation is always perfect and the traffic is stationary. When $\frac{t_{coll}}{t_{st}}$ is relatively low ($\frac{1}{3}$), the algorithm performs well because the statistics collected at the previous window, most likely apply to the current one, since for each t_{st}, three estimations are taken. The USP remains similar when $t_{coll} = t_{st}$ because the estimation is better due to more samples. However, a gain with respect to the previous case is not shown because the probability that the statistics between consecutive windows have considerably changed, i.e. previous estimations degrade the performance, compensates the final USP. Then, when $\frac{t_{coll}}{t_{st}} > 1$, the USP keeps decreasing because the algorithm is less reactive. This analysis highlights the efficiency of our algorithm when the collection window is comparable or lower to the interval of time where the statistics can be considered static.

F. Scalability of the substrate network

In this section, we test the algorithm with larger and more complex network. We take a substrate network composed of 20 nodes, with probability of being connected with each other equal to 0.5. We summarized in Figure 8 the network used for all the previous discussions, Scenario A, and the more complex network, Scenario B.

We aim to compare the three main KPIs, Table V, to check the impact of the size of the substrate network. For this experiment, we considered only high priority VNRs in a low-load configuration mode. This is the reason why the acceptance ratio is in both cases 100%. This is done to avoid that the average delay is also impacted by the waiting time due to unavailability of the resources. It can be seen that the actual average outage in both cases is not as the expected one, i.e., 0.01%, due to error in the traffic description for limited samples. However, we should not expect that this is impacted on the network size. On the contrary, the average delay, which is the average computing time in this scenario, increases considerably with increasing network size.

Table V: Comparison KPIs

Performance	Scenario A	Scenario B
Average outage	0.34 %	0.26 %
Average delay [s]	5.4	22.17
Acceptance ratio	100 %	100 %

As the results in Section V are obtained with the testbed, the configuration in Scenario B would generate so much complexity in the emulation side that the results are impacted from the hardware limitations of the tools, in particular network emulator and controller.

V. TESTBED EXPERIMENT RESULTS

After defining some relevant parameters, we compare the SCA-R to the following baseline approaches:

- CEVNE [27] restricted to link mapping (load-balancing approach);
- SP-1 [30];
- MIQCP [19], restricted to link mapping;
- SP-2, Shortest path with statistics collection.

We have chosen these 4 baselines for the following reasons. CEVNE and SP-1 are well-known VNE algorithms which belong to the categories of algorithms that do not collect any statistics and the embedding is executed purely on the required data rate for each VNR. MIQCP and SP-2, instead, represent the set of algorithms that consider also a traffic statistical description (considered to be known a priori) but without a joint optimization (probabilistic approach when optimizing the assigned data rate).

A. Testbed setup

For the previous results and discussion, we have used Matlab to run initial simulations to fix parameters such as collection time, congestion ratio and interval time slot duration. From now on, we validate SCA-R into a developed version of the testbed presented in [31]. The schematic graph is shown in Figure 9. The testbed is composed by four main elements. The network emulator, Mininet, the SDN controller, RYU, the VNE algorithm running in Matlab and the traffic generator OSTINATO. We underline that the VNE might eventually run inside the SDN controller but, as the purpose of our testbed is to be tested with different VNE algorithms, we find it convenient to keep the two entities separated for a faster integration.

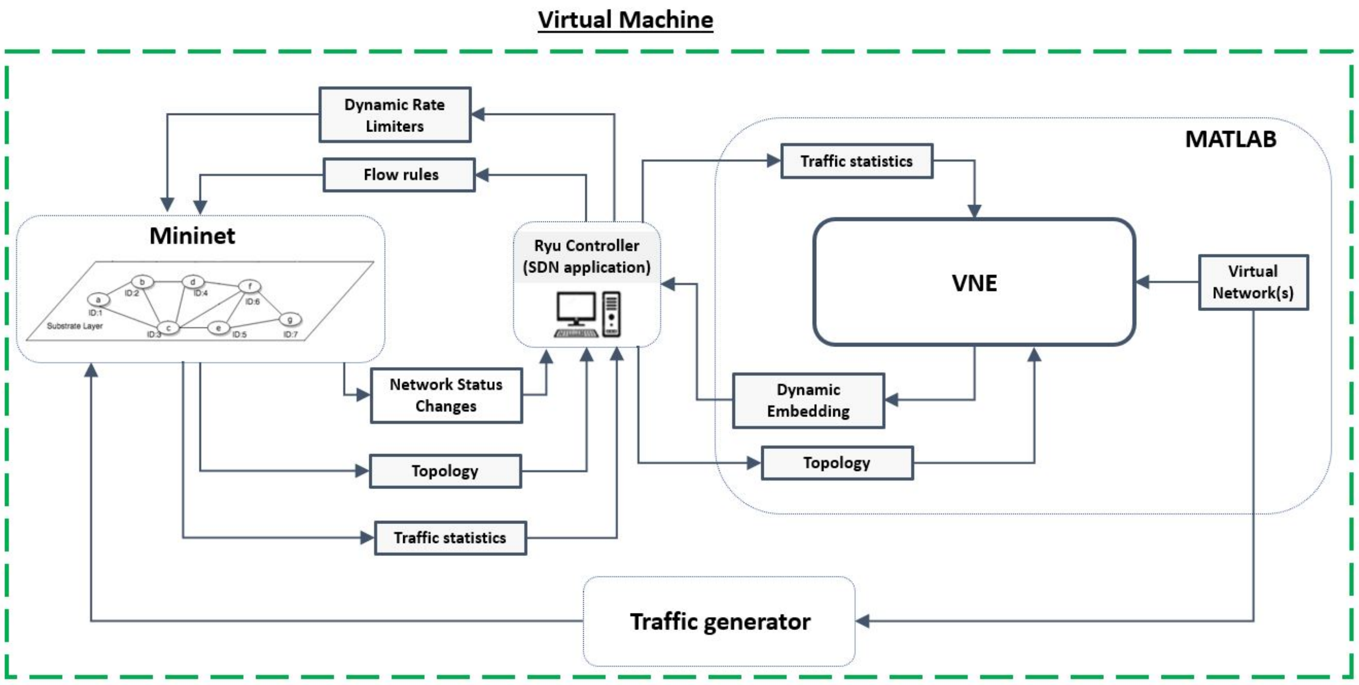


Figure 9: Testbed setup

Table VI: Ostinato traffic description

L1	VLAN	L2	L3	L4	L5
Mac	Tagged	Ethernet II	IP	UDP	Empty Text

We use OSTINATO to simulate traffic demands, with a varying data rate over time depending on the traffic model we have selected. OSTINATO [32] creates strings of traffic with multiple choices for each ISO/OSI layer, starting from physical up to application layer. Table VI summarizes the input settings, as defined in Ostinato. UDP traffic is used for each VNR, with vlan identifier to differentiate the VNRs. Then, the stream is initiated with the datarate and lifetime.

To collect statistics from every VNR, the process is the following. As shown in Figure 10, once the mapping is performed, the traffic flows from source to destination. The SDN controller, which is connected to all switches between source and destination, has access to the number of bytes transmitted for each installed flow of every switch (”*byte_count*”). So, given an interval of time Δt, the current data rate at time t in bps for the VNR j and for each link is passing through, is computed as:

$$\text{data rate(t)} = 8 \cdot \frac{d_j(t) - d_j(t-\Delta t)}{\Delta t}, \tag{16}$$

given $d_j(t)$ the transmitted bytes of VNR j until time t. It is worth underlining that, although the SDN controller computes the actual data rate for each link, we only select the data rate at the last switch, before the destination host, to have only one value per VNR.

After some offline analysis, $\Delta t = 5s$. was taken as interval time for collecting the statistics. Lower Δt gave more unstable values which would have impacted the overall performance.

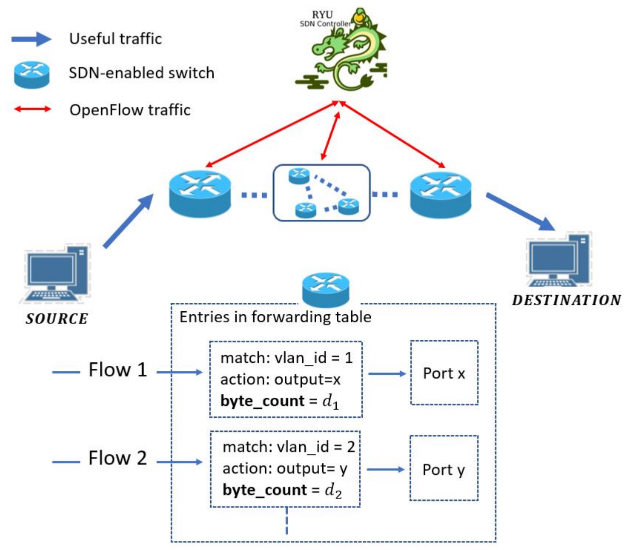


Figure 10: Traffic statistics collection

On the other side, higher values would have slowed down the time slotting strategy of the VNE algorithm.

B. Acceptance ratio

In this subsection, the acceptance ratio comparison is shown. For all following experiments and results, we are using the parameters value obtained in the previous subsections, t_{coll}, congestion ratios R_h, R_m and R_l and interval time slot duration t_{slot}. Two scenarios are considered. The first one (Figure 11) is longer and with higher available substrate capacity (20000 bps instead of 10000 bps) to decrease the load of

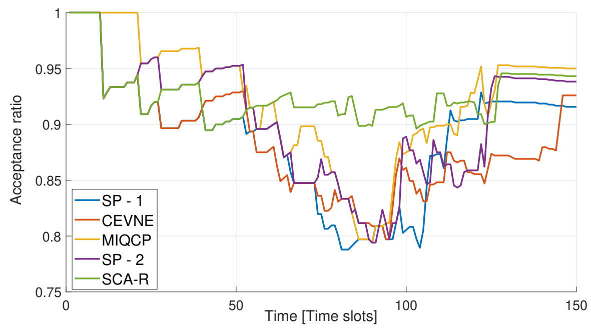


Figure 11: Acceptance ratio vs time

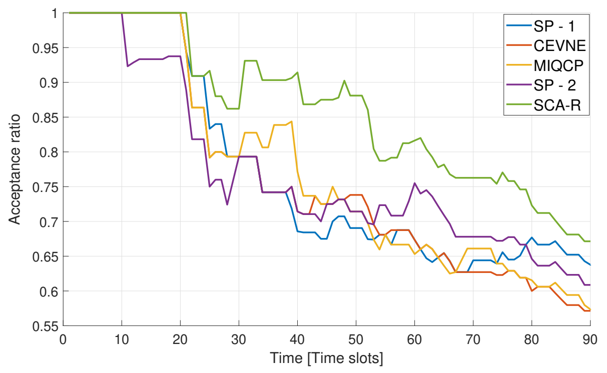


Figure 12: Acceptance ratio vs time in congested network

the network. We plot the acceptance ratio over time for the four baseline algorithms and the proposed SCA-R. In this case, we that there is an initial part where the proposed algorithm performs even worse than most of the baselines. This is expected because SCA-R is using a congested ratio which reduces the acceptance ratio. However, as soon as the traffic is collected and the joint optimization is computed, more substrate capacity will be free, which brings to higher acceptance ratio (intermediate area). Afterwards, since we limit the number of generated VNRs to 100, the acceptance ratio will grow again since VNRs are also expiring, releasing capacity for the previously failed VNRs.

We also show that even better results are obtained with higher network load, as presented in Figure 12. Even in this case, at the beginning, the network is not congested and the efficiency of SCA-R cannot be appreciated because the probability that the VNR is not being accommodated is higher, since a lower portion of capacity can be used (congestion ratio). When the network is more congested, the efficiency of accumulating the statistics and jointly optimizing the embedding is visible and brings to higher acceptance ratio compared to baselines.

C. Average queuing delay

In this section we analyze the average queuing delay for each type of VNR. Firstly, we only consider SCA-R and we show that on a very low time scale, as presented in Figure 13, for each time slots, the lower the priority, the higher the average queuing delay. It is worth underlining that when in a

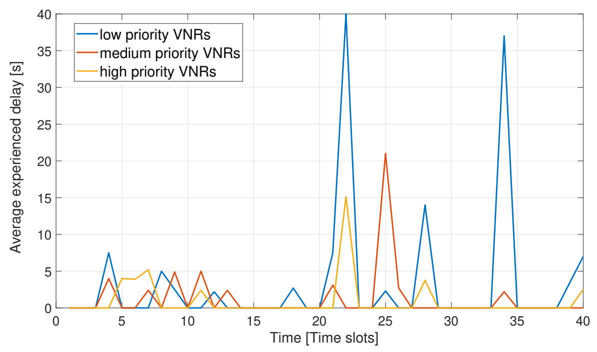


Figure 13: Average queuing delay

Table VII: Average queuing delay vs types of VNRs

Algorithm	High	Medium	Low
SP - 1	106.82 s.	276.14 s.	78.27 s.
CEVNE	198.74 s.	184.09 s.	112.06 s.
MIQCP	140.05 s.	152.64 s.	89.66 s.
SP - 2	154.90 s.	146.77 s.	125.27 s.
SCA-R	22.04 s.	115.48 s.	97.52 s.

time slot the delay of a high priority is higher than a lower one, this is because the highest priority VNR has failed to be mapped the previous time slots. Thus, its queuing time will be counting also the previous time slot duration. This is why we should look at the average queuing delay for each priority level. In addition, in the previous simulation, we showed that SCA-R improves, after a certain amount of time, the acceptance ratio compared to the baselines. As a consequence, on average VNR, independently of their priority, will have to wait more time before being embedded when baselines are considered. In fact, in the following table, we present the comparison of average queuing delay for the four baselines and for SCA-R, over a simulation time of 90 time slots, as presented in Figure 12. Due to decreasing acceptance ratio after 20 time slots, VNRs are more likely to wait. This is the reason why the value described below might seem high.

Table VII shows that when comparing the same priority level between SCA-R and baselines, the former always performs better. The fact that there are lower values for low priority VNRs with respect to medium priority is because the system is considering only the accepted VNRs, ignoring the ones which are not accepted. Thus, since the system starts to embed from the higher priority VNRs, few low priority VNRs are counted in this computation, which brings the average lower. This is also the reason why for low priority VNRs, it looks like SCA-R performs worse than some baselines. This is due to the fact that there are more accepted low priority VNRs than some baselines, and, therefore, the final average is counting more VNRs with respect to those baselines which have lower average queuing time.

D. Average USP

In this subsection, the average USP metric, computed as the percentage of time where the traffic is lower than the assigned data rate, is considered. It is worth noting that the first two baselines do not consider any probabilistic approach so, unless

Table VIII: Average USP

	High	Medium	Low
Expected	99.99%	97%	90%
Real	97.7%	95.6%	89.4%

the traffic is higher than the maximum agreed level (which is outside the scope of this paper), the USP is always equal to 1. For the baselines 3 and 4, there is a collection period, where the traffic is described as a random variable, which can bring to some incorrect values, but afterwards there is no joint optimization as SCA-R. Thus, if USP < 1, this is due to an error introduced by the number of analyzed samples. Even this matter is outside the scope of this paper. For the above-mentioned reasons, it does not make sense to compare SCA-R to baselines, but we still provide a brief description of the results achieved. For each type of VNR, SCA-R optimizes the assigned data rate based on a defined minimum USP. Thus, in this section, we check the results of the obtained USP, which should be close to the expected values P_g.

It is worth adding that the congestion ratio does not impact/reduce the USP since, if the constraint on available capacity (lowered by congestion ratio) is not respected, the network will not be mapped. Table VIII shows the results for the SCA-R, for the 3 different priority values. The results are not exactly equal to the expected values due to the error, in the traffic description, introduced by the limited number of samples.

VI. Conclusion

In this paper, we proposed a formulation for link mapping in VNE with joint optimization of load balancing and assigned data rate (SCA-VNE). Unlike the traditional approaches for VNE which assign to each VNR the resource requirements, either static or dynamic, in a deterministic way, SCA-VNE jointly optimizes the embedding for each VNR with a probabilistic approach. In fact, instead of assigning 100% of resource requirements for 100% of the time, SCA-VNE optimizes the assigned data rate with the guarantee that the USP is kept under control and lower-bounded by the minimum guaranteed QoS. Minimizing the assigned data rate to each VNR, increases the acceptance ratio due to more available substrate capacity. The effect of traffic's non-stationarity on SCA-VNE has also been discussed, showing a worsening of the performance with an increase of non-stationarity. SCA-VNE is supported by SDN, which describes the real traffic sent. Due to the exponential complexity of SCA-VNE, we proposed the relaxed version, named SCA-R. We compared SCA-R to four baseline approaches, and we proved that, the further the time goes, i.e. the more the load over the substrate network, the higher the acceptance ratio of SCA-R compared to baselines. In fact, initially SCA-R uses the traditional approach since there are no statistics collected on the traffic. However, when statistics are collected, SCA-R can run the joint optimization problem and the advantage starts to be noted. We differentiate VNRs based on a priority, which not only sorts them when being embedded, but also differentiates them when guaranteeing a defined USP. In fact, the higher the priority, the higher the USP. We also used known techniques such as congestion ratio, to increase the probability of accepting high priority VNRs. SCA-R has also shown a lower average queuing time for all priority levels compared to baselines. We have finally shown that the average USP values are close to the expected values for the 3 different priority levels. We plan to extend this work to an integrated satellite-terrestrial network, simulated in STK, with dynamic links and real-time mapping decisions based, not only on the current traffic demand, but also on the available network connections.

Acknowledgment

This work is supported by Fond National de la Recherche Luxembourg (FNR) programme, Industrial Partnership Block Grant (IPBG), ref. 14016225, and the FNR project, Autonomous Network Slicing for Integrated Satellite-Terrestrial Transport Networks (ASWELL), ref. 13718904.

References

[1] H. Zhang, N. Liu, X. Chu, *et al.*, "Network Slicing Based 5G and Future Mobile Networks: Mobility, Resource Management, and Challenges," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 138–145, 2017.

[2] 3GPP, "" 3rd Generation Partnership Project (3GPP), Technical Report (TR) 23.501, Oct. 2017, Version 18.2.2. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144.

[3] ——, "" 3rd Generation Partnership Project (3GPP), Technical Report (TR) 28.530, Oct. 2019, Version 15.3.0. [Online]. Available: https://www.3gpp.org/ftp/tsg_sa/WG5_TM/TSGS5_128/SA_86/28530-f30.doc.

[4] H. Cao, S. Wu, Y. Hu, *et al.*, "A survey of embedding algorithm for virtual network embedding," *China Communications*, vol. 16, no. 12, pp. 1–33, 2019.

[5] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual Network Embedding: A Survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[6] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[7] A. Hashmi and C. Gupta, "A Detailed survey on Virtual Network Embedding," in *2019 International Conference on Communication and Electronics Systems (ICCES)*, 2019, pp. 730–737.

[8] M. Chowdhury, M. R. Rahman, and R. Boutaba, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 206–219, Feb. 2012.

[9] Y. Xue, J. Peng, K. Han, and Z. Zhu, "On table resource virtualization and network slicing in programmable data plane," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 319–331, Mar. 2020.

[10] H. Cao, Y. Zhu, G. Zheng, and L. Yang, “A novel optimal mapping algorithm with less computational complexity for virtual network embedding,” *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 356–371, Nov. 2017.

[11] J. Liu, X. He, T. Chen, X. Wang, R. Luo, and T. Huang, “SN-VNE: A Virtual Network Embedding Algorithm for Satellite Networks,” in *2019 IEEE/CIC International Conference on Communications Workshops in China (ICCC Workshops)*, 2019, pp. 1–6.

[12] D. Yang, J. Liu, R. Zhang, and T. Huang, “Multi-Constraint Virtual Network Embedding Algorithm For Satellite Networks,” in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[13] I. Maity, T. X. Vu, S. Chatzinotas, and M. Minardi, “ D-ViNE: Dynamic Virtual Network Embedding in Non-Terrestrial Networks,” in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, 2022, pp. 1–5.

[14] C. K. Dehury and P. K. Sahoo, “DYVINE: Fitness-Based Dynamic Virtual Network Embedding in Cloud Computing,” *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1029–1045, 2019.

[15] V. A. Gang Sun hongfang Yu and L. Li, “A cost efficient framework and algorithm for embedding dynamic virtual network requests,” *Future Generation Computer Systems*, vol. 16, no. 12, pp. 1–13, 2012.

[16] M. Lu, Y. Lian, Y. Chen, and M. Li, “Collaborative Dynamic Virtual Network Embedding Algorithm Based on Resource Importance Measures,” *IEEE Access*, pp. 1–17, 2018.

[17] D. Chen, X. Qiu, Z. Qu, S. Zhang, and W. Li, “Algorithm for virtual nodes reconfiguration on network virtualization,” in *2011 International Conference on Advanced Intelligence and Awareness Internet (AIAI 2011)*, 2011, pp. 333–337.

[18] Q. Lin, Y. Huang, and Y. Li, “A New Algorithm for Virtual Networks Reconfiguration with Adaptive Interval,” in *2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC 2018)*, 2018, pp. 1–5.

[19] K. Qu, W. Zhuang, Q. Ye, X. Shen, X. li, and J. Rao, “Dynamic Flow Migration for Embedded Services in SDN/NFV-Enabled 5G Core Networks,” *IEEE Transactions on Communications*, vol. 68, no. 4, pp. 2394–2408, 2020.

[20] Y. Yuan, C. Wang, C. Wang, B. Zhang, S. Zhu, and N. Zhu, “ A Novel Algorithm for Embedding Dynamic Virtual Network Request,” in *2015 2nd International Conference on Information Science and Controller Engineering*, 2015, pp. 1–5.

[21] S. R. Chowdhury, R. Ahmed, N. Shahriar, *et al.*, “Revine: Reallocation of virtual network embedding to eliminate substrate bottlenecks,” in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2017, pp. 116–124. DOI: 10.23919/INM.2017.7987271.

[22] F. Xu, Y. Li, H. Wang, P. Zhang, and D. Jin, “Understanding mobile traffic patterns of large scale cellular towers in urban environment,” *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 1147–1161, 2017.

[23] L. Yu, H. Shen, Z. Cai, L. Liu, and C. Pu, “Towards bandwidth guarantee for virtual clusters under demand uncertainty in multi-tenant clouds,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 2, pp. 450–465, 2018.

[24] S. Xu, P. Li, S.-Y. Guo, and X. Qiu, “Fiber-wireless network virtual resource embedding method based on load balancing and priority,” *IEEE Access*, vol. 6, pp. 33 201–33 215, 2018.

[25] J. Cai, X. Nian, H. Gu, and L. Zhang, “A user priority-based virtual network embedding model and its implementation,” in *2013 IEEE 4th International Conference on Electronics Information and Emergency Communication*, 2013, pp. 33–36.

[26] F. Sadia, N. Jahan, L. Rawshan, M. T. Jeba, and T. Bhuiyan, “A priority based dynamic resource mapping algorithm for load balancing in cloud,” in *2017 4th International Conference on Advances in Electrical Engineering (ICAEE)*, 2017, pp. 176–180.

[27] M. Pham, D. B. Hoang, and Z. Chaczko, “Congestion-Aware and Energy-Aware Virtual Network Embedding,” *IEEE/ACM Transactions on Networking*, vol. 28, no. 1, pp. 210–223, 2020.

[28] A. Vanelli-Coralli, G. E. Corazza, M. Luglio, and S. Cioni, “The isicom architecture,” in *2009 International Workshop on Satellite and Space Communications*, 2009, pp. 104–108.

[29] M. Chowdhury, M. R. Rahman, and R. Boutaba, “ViNE-Yard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping,” *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 206–219, 2012.

[30] D. Chemodanov, F. Esposito, P. Calyam, and A. Sukhov, “A constrained shortest path scheme for virtual network service management,” *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 127–142, 2019.

[31] F. Mendoza, M. Minardi, S. Chatzinotas, L. Lei, and T. X. Vu, “An SDN Based Testbed for Dynamic Network Slicing in Satellite-Terrestrial Networks,” in *IEEE International Mediterranean Conference on Communications and Networking (MEDITCOM) 2021*, Athens, Greece, 2021.

[32] [Online]. Available: https://ostinato.org/.