Network slicing with load-balancing for task offloading in vehicular edge computing

Khaled Hejja, Sara Berri, Houda Labiod

VEHCOM:100419

Vehicular Communications ••• (••••) ••••••

Contents lists available at ScienceDirect

Vehicular Communications

www.elsevier.com/locate/vehcom

Network slicing with load-balancing for task offloading in vehicular edge computing

Khaled Hejja [a], Sara Berri [b,*], Houda Labiod [a]

[a] INFRES, Telecom Paris, Institute Polytechnic of Paris, 91120 Palaiseau, France
[b] ETIS UMR 8051, CY Cergy Paris University, ENSEA, CNRS, 95000, Cergy, France

ARTICLE INFO

Article history:
Received 5 March 2021
Received in revised form 10 September 2021
Accepted 29 September 2021
Available online xxxx

Keywords:
Edge computing
Network slicing
Vehicular offloading
Load-balancing
Network function virtualization

ABSTRACT

The support of edge computing for vehicular technologies gained increasing momentum with 5G to fulfill efficient offloading tasks from vehicles towards the edge nodes. Accordingly, vehicles demanding powerful computation and large storage resources will be directed to communicate with the nearest edge computing nodes hosted at a wireless 5G new generation nodes (gNBs) or a road side units (RSUs). To efficiently utilize the edge nodes' resources, network slicing and load-balancing features can greatly help in that, therefore, this paper proposes an algorithm for Vehicular Edge Computing (VEC) with network slicing and load-balancing based on resources utilization, denoted as VECSlic-LB, specifically dedicated for offloading tasks from vehicles to edge nodes at gNBs or RSUs. The algorithm can holistically view and manage the whole network, and use network function virtualization framework to manage the data plane. VECSlic-LB can handle a mix of slicing configurations, capable of balancing the loads between various slices per node, and can support multiple edge computing nodes. Several simulations were conducted comparing the performance of the proposed algorithm to the optimal solution, resulting on very close acceptance ratios as the optimal solution, and also was evaluated against a recent reference algorithm, providing more efficient resource utilizations ratios, saving up to 48% of the resources better than the state-of-art algorithm.

1. Introduction

The support of 5G for edge computing and vehicle-to-everything (V2X) communication, facilitated the creation of VEC as an emerging technology to support several vehicular applications such as vehicles platooning, advanced-driving and remote-driving applications, to enable automated driving, where longer inter-vehicle distance is assumed, in addition to allowing vehicles to share and coordinate their trajectories and intentions for safer traveling, collision avoidance, and improved traffic efficiency [1]. Moreover, the 3rd generation partnership project (3GPP) standardization activities opened new era to use VEC technologies for vehicles' extended sensors applications such as exchanging large amounts of data from sensors for post processing and analysis at the edge nodes. For example, sensors in vehicle-to-vehicle (V2V), or vehicles-to-infrastructure (V2I) can offload their data towards the VEC nodes residing at road side units (RSU), Internet-of-things (IoT) devices, or wireless base stations such as 5G's Next Generation NodeBs, for better precision, fast analysis, or to store the data for future access [2].

However, most existing works in literature about offloading in vehicular edge computing networks did not consider network slicing to differentiate such sensitive vehicular applications from each other, nor considering balancing the loads between the edge nodes themselves or within the edge computing servers to benefit from any available excessive computation resources that can be utilized by the vehicles. Consequently, vehicular networks' operators may face complications in designing their VEC networks, forcing them to apply solutions of higher costs and limited support to the new emerging vehicular services [3].

Network slicing was specified in [4] to support network service differentiation and diversification based on the requirements from variety of industries, including V2X applications. Accordingly, 3GPP highlighted in [5] the importance of abstracting the required network slice information to support V2X applications and multiple public land mobile network operator (PLMN). Technically this means that, network slicing can be considered as a complete and separate logical network, to provide specific network capabilities and network characteristics, similar to any other physical network operation. Therefore, several V2X use cases and classes such as

* Corresponding author.
E-mail addresses: khalid.a.hijjeh@gmail.com (K. Hejja), sara.berri@ensea.fr (S. Berri), houda.labiod@telecom-paris.fr (H. Labiod).

https://doi.org/10.1016/j.vehcom.2021.100419
2214-2096/

vehicles' cooperative maneuvering and safety, autonomous navigation, and remote driving; which have diverse and conflicting computing, storage, latency, reliability, and throughput requirements, can leverage the use of network slicing to allow the service providers to design one or more network slices, and bundle them to support multiple V2X key performance indicators and quality of service requirements [6,7].

Regarding load-balancing, there are several strategies widely employed in cloud environments [8], such as random access load-balancing technique, which assumes that the demanding users need to connect randomly to any available edge server from a list of available servers. Another technique is load-balancing based on number of users to determine the capability of edge servers, therefore, if the capacity of the first server is overloaded, a new server will be activated to meet the demand. Similarly, the load-balancing technique based on the utilization of the central processing unit (CPU) or memory of the cloud or edge servers, in this case if the requested CPU or memory load did not meet the total capability of the target server, the load-balancer will instantiate a new server at that cloud or edge node [9]. Lastly, throughput based load-balancing techniques which usually relies on evaluating the network quality of service and performance metrics, such as the efficiency of throughput, bandwidth, and latency [10,11].

In light of that, the authors propose an algorithm for vehicular edge computing with network slicing and load-balancing, denoted as (VECSlic-LB), specifically designed to handle offloading tasks from vehicles towards wireless nodes (i.e. RSUs and gNBs), while balancing the loads between the sliced edge computing servers hosted at these nodes. VECSlic-LB follows the spirit of Software Defined Networking (SDN) in providing a centralized management of the VEC network usage. Moreover, based on the clarifications of [12], VECSlic-LB applies the network function virtualization (NFV) architecture on the data plane at the RSUs and gNBs, to decouple the virtualized network functions from the physical devices of the RSUs and gNBs, and to efficiently manage and utilize the physical resources at the edge computing servers. Moreover, VECSlic-LB integrated the network slicing feature to allow V2X service providers to have their own slices with separate computing, storage, and networking resources that can be assigned according to the V2X service type. Finally, the new addition in this paper is the load-balancing feature, which allows VECSlic-LB to distribute the overall load between the different slices at an edge node, and to enhance the efficiency in utilizing the edge computing resources.

Results of VECSlic-LB will be evaluated against the optimal VEC solution, and to a recent cloud computing algorithm in literature, the power aware network function virtualization (PaNFV) developed for cloud and edge computing [13,14]. Notice that PaNFV does not support load-balancing, and only works for single slice configuration. Moreover, in all evaluations, VECSlic-LB will be compared to VECSlic without load-balancing, to highlight the benefits of VECSlic-LB, specifically the impacts of adding the load-balancing.

The main contributions of this paper are as follows:

1. VECSlic-LB is designed to support vehicular offloading tasks towards the edge computing servers hosted at wireless nodes, which is differentiated from other algorithms by:
   - Supporting sliced edge computing servers.
   - Supporting load-balancing between the sliced edge computing nodes.
   - Virtualizing the data plane of the sliced edge nodes based on NFV framework.
   - Decoupling the control plane from data plane emulating SDN controllers.
2. Compared to the optimal solution, VECSlic-LB provided close acceptance ratios to the optimal, and
3. When compared to state-of-art algorithm, VECSlic-LB resulted on more efficient resource utilization by 48%.

Rest of the paper is organized as follows: Section 2 provides related work, Section 3 discusses the overall framework defining the physical network, and the problem formulation. Then Section 4 details in depth the proposed algorithm, including its computational complexity and the evaluation metrics that will be used in the simulations. This is followed by Section 5 which covers the simulation settings and discusses the results of the conducted experiments, and conclusions are presented in Section 6.

2. Related work

Task offloading techniques in vehicular edge computing are attracting the attention of a large number of applications in vehicular networks as a means to satisfy their increasing resource demands in terms of data storage and processing. A detailed survey about the recent advances in the task offloading techniques through vehicular ad-hoc networks is provided in [15], focusing on the communication patterns among vehicles and infrastructure, classifying them into task offloading through vehicle-to-vehicle communications, vehicle-to-infrastructure communications, and vehicle-to-everything communications. Another recent survey paper was conducted by [16], discussed the VEC architecture, smart vehicle services, communication, and applications, and concluded by some open research issues and challenges. In addition, [17] reviewed the state-of-art literature focusing on fog computing, mobile edge computing, cloud computing, Internet of things, autonomous automobiles, cloud-lets, and micro data centers for smart cities. They identified their key requirements, and listed and discussed some of the related open challenges, for instance privacy and mobility.

On vehicular task offloading, [18] proposed data offloading from cellular network's base station (BS) to IEEE 802.11p RSU or WiFi Access Points (AP) using the multiple hop vehicular to vehicular (V2V) path to increase the possibility of vehicular ad-hoc networks (VANET) data offloading. They proposed the mobile edge computing (MEC)-based method to perform the offloading, giving that each vehicle reports its context to the MEC server periodically, accordingly the MEC server uses a specific path selection method to find suitable offloading path for vehicle v before or after it enters into (has left) the signal coverage of the ahead or rear RSU or AP. Another research was conducted by [19] to investigate the scheduling problem in a vehicular edge computing scenario to minimize the offloading cost in terms of a trade-off between task latency and energy consumption. They provided an optimal solution to the offloading scheduling problem, modeling it as a Markov decision process using deep reinforcement learning to deal with the dynamic mobile state space. In the study of [20], the authors proposed a collaborative edge computing scheme for vehicular Internet-of-things. The proposed scheme enabled a networking and computing architecture by forming vehicular clusters based on the edge architecture, and concluded by providing an optimized offloading algorithm based on greener intelligent transportation system architecture. Similarly, [21] investigated the task offloading in vehicular edge computing environment, and provided an optimal solution for static and dynamic offloading tasks for time-varying fading channels of uncertain allocated bandwidths.

Another comprehensive study about 5G network slicing in [22], focused on the principles and models of resource allocation algorithms for network slicing. They introduced the basic ideas of the software defined networks and network function virtualization for network slicing. In addition they studied the fundamental framework of resource allocation algorithms for network slicing, and analyzed the resource types for slicing the radio access and core networks. Furthermore, the authors categorized the mathematical

Table 1
Notations.

Notations	Description.	Notations	Description.
T	Simulation total time.	P	Weighted directed graph representing physical network.
N	Set of nodes in the physical network.	L	Set of links in the physical network.
$Path^P$	All paths in the physical network.	P_{sd}	Physical path between nodes s and d.
R	Set of all Road Side Units.	G	Set of all gNB node sites.
S	Set of all Slices representing PLMNs.	s	Slice s from the set of slice in S
$PLMN$	Set of PLMNs.	$plmn$	The $plmn$ from the set $PLMN$.
V	Set of all vehicles.	v	Index of vehicle $v \in V$.
l_{vr}	Link between vehicle v and RSU r.	SFC^v	Weighted directed graph representing SFC^v.
F	Set of all VNF types.	$vCPU$	VNF representing virtual CPUs.
$vNIC$	VNF representing virtual Router.	vHD	VNF representing virtual storage.
v_{plmn}	vehicle's v PLMN number.	num_v	Number of demands.
v_{cur}	Current location of vehicle v.	v_{des}	Destination location of vehicle v.
J_v	Total number of data chunks to be offloaded from vehicle v.	dc_v	Size of one data chunk in Bytes from vehicle v.
c_v^s	Demanded CPU by vehicle's v VNFs on slice s	c_{r_s}	Available CPU by RSU r in Slice s
c_{g_s}	Available CPU by gNB g in Slice s	m_v^s	Demanded storage by vehicle's v VNFs on slice s
m_{r_s}	Available storage by RSU r in Slice s	m_{g_s}	Available storage by gNB g in Slice s
b_v	Demanded data rate by vehicle v	b_{vr}	Available data rate by RSU r for vehicle v
b_{vg}	Available data rate by gNB g for vehicle v	d_v	Maximum acceptable delay for vehicle v
d_{vr}	Current delay in the link between v and r	d_{vg}	Current delay in the link between v and g
AR^v	Accumulated sum of the accepted offloading tasks		

models of slicing resource allocation algorithms, and concluded by discussing some of the open research issues and their potential solutions. Moreover, [23] provided a comprehensive review and solutions related to 5G network slicing, started by presenting 5G service quality and business requirements followed by a description of 5G network softwarization and slicing paradigms including essential concepts, history and different use cases. Then the authors provided network slicing technology enablers including 5G technologies such as software define networking, network function virtualization, mobile edge computing, cloud and fog computing, network hypervisor, virtual machines and containers. Afterwards the authors compared the various 5G architectural approaches in terms of practical implementations, technology adoptions and deployment strategies to support network slicing. Moreover, they investigated the standardization efforts in 5G networks regarding network slicing and softwarization.

On load-balancing at edge computing nodes, the authors in [24] provided a detailed survey of current load-balancing techniques, and discussed and analyzed the utilized methodology for load-balancing, implementation tools, and evaluation metrics at edge environments. They reviewed the optimization technique, traffic load distribution and dynamic load based techniques, and concluded by identifying the current research gaps and future directions. The authors in [25] addressed the offloading problem for vehicular edge computing networks, by introducing a load-balancing scheme to minimize the processing delay of the vehicles' computation tasks. The authors proposed a software defined networking technology to centralize network and vehicle information management, and to achieve the load balancing of the computation resources at the edge servers, they assumed that all edge servers are candidates for offloading the tasks from a certain vehicle.

Moreover, the authors in [26] proposed integrating load-balancing with offloading for a multi user and multi server vehicular edge computing system. They formulated the joint load balancing and offloading problem as a mixed integer nonlinear programming problem to maximize system utility. Then, they developed an algorithm to jointly make vehicular edge computing server selection, and optimize offloading ratio and computation resources together. [27] proposed an approach for optimal placement of road side units based on load balanced routing, to improve the stability and battery lifetime for the individual nodes in vehicular ad-hoc networks. The authors assumed energy levels of transmission in each vehicle as a variable, accordingly, they balanced the loads on road side units by imposing some upper bounds on their received energy levels based on the separation distance from the vehicles, then they selected the node that utilizes the least power to handle the traffic from vehicles.

As a conclusion from the literature section, it was challenging to find a study integrating network slicing and load-balancing into edge computing environments, let alone applying centralized control plane (as in SDN) and virtualized data plane (as in NFV) for vehicular offloading tasks. Based on that, the recent publications by [13,14] will be used as the main references for the developed algorithm in this paper, since they support offloading on cloud and edge computing environments for single slice, emulate the SDN and NFV technologies, but do not support load-balancing. Important to point out that the works from [13,14] were originally evaluated against the state-of-art algorithms from [28–30] and improvements were shown.

3. Framework

In this section, we first develop the physical network model of vehicular network with a set of RSUs and 5G gNBs equipped with a physical edge server, and a set of vehicles which might request for task offloading. Then, we formulate the problem of task offloading as an optimization problem.

In order to evaluate the proposed algorithm for offloading tasks, this section presents a framework including a detailed description for the physical network composed of wireless base stations (RSUs and gNBs) hosting edge computing resources and their connecting links. In addition, the characteristics of virtual network function demands from the vehicles are descried focusing on the demands for specific edge computing resources such as processing power, storage, and bandwidth between the vehicles and the edge nodes. The parameters used in defining the proposed model are summarized in Table 1.

3.1. Physical network model

The physical network is modeled as a weighted directed graph $P = (N, L)$, where N and L are the sets of physical nodes and links respectively. The graph will be formulated as a set of network paths $Path^P$ connecting the nodes and links, giving that P_{sd} is a path between the source node s and the destination node d, and can be formulated of one or more nodes and links. N is composed of a set of RSUs denoted as R, and another set of 5G gNBs denoted as G, while the set of links L are used to connect the RSUs and gNBs together. Each gNB $g \in G$ can be connected to nearby gNBs and RSUs, but RSUs are only allowed to be connected to their

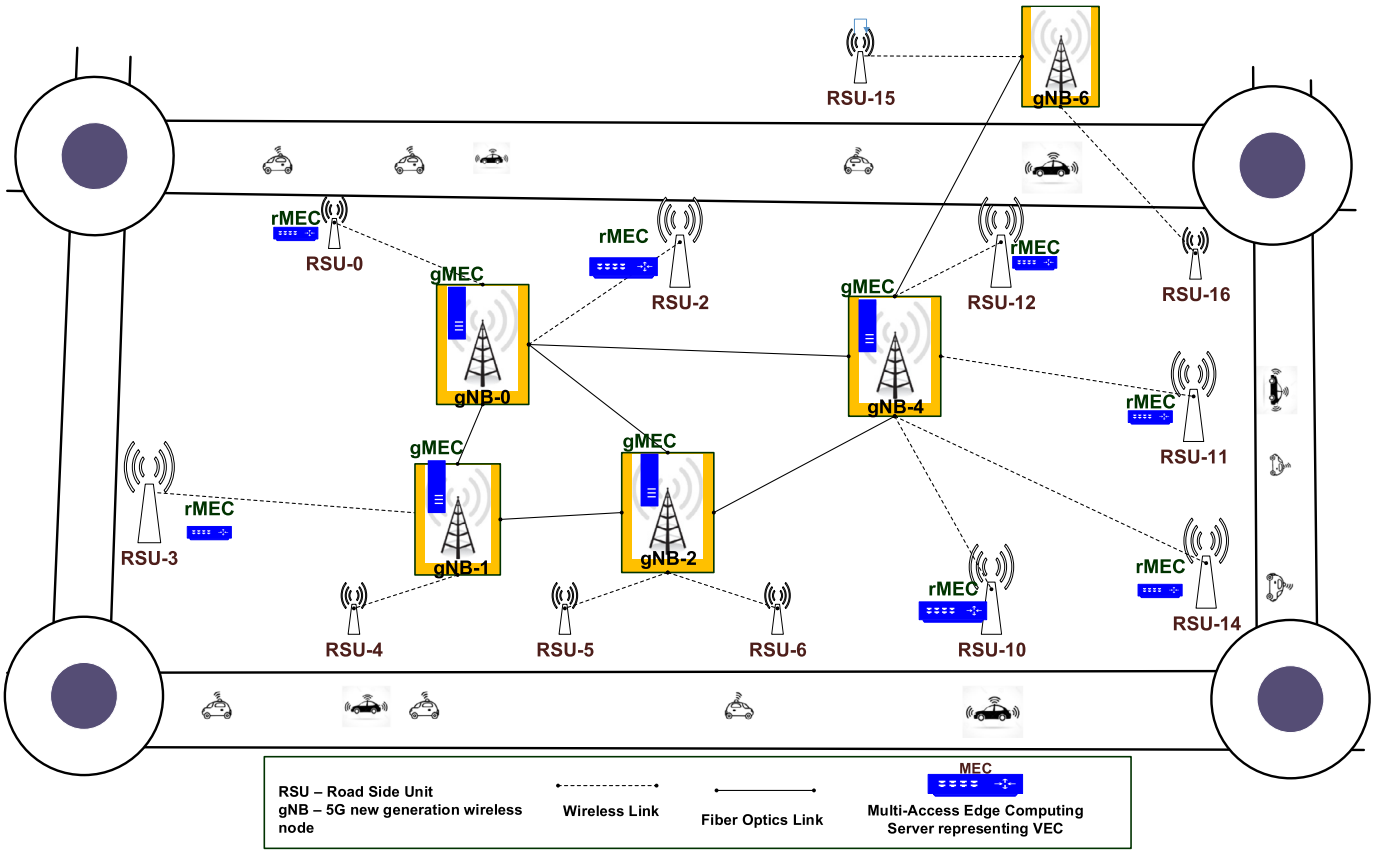

Fig. 1. Sample Network Model.

nearby gNBs as shown in the sample network in Fig. 1. Moreover, each RSU $r \in R$ can be connected to one or more vehicle denoted as v, and each vehicle will have a delay denoted by d_{vr} and data rate b_{vr} towards the RSU. For each gNB $g \in G$ it also can be directly connected to multiple vehicles, while each vehicle will have a delay d_{vg} and data rate b_{vg} towards the gNB, and the gNB can be linked to one or multiple RSUs with delay d_{rg} and data rate b_{rg}.

In this paper, the gNBs and the sliced edge core are assumed working based on standalone scenario, which is based on pure 5G core and access components only, and following 3GPP Rel.16 NR-V2X (new radio vehicle-to-everything) standards [31]. In the proposed physical network, a virtualized and sliced edge server at RSU r or gNB g are modeled to represent a physical hardware platform and a host operating system. The hardware platform will be managed by the NFV infrastructure manager (VIM) to provide the physical resources such as, processing power, network interface cards (NIC), and storage [31].

The maximum resources of RSU r and gNB g for processing power are denoted as C_r and C_g, which will be divided into slices denoted as $s \in S$, representing the public land mobile operators (PLMNs), each has available capacity denoted as c_{r_s} and c_{g_s}. NICs are represented by their data rates denoted as B_r and B_g for maximum data rate capacities, where b_r and b_g denoting the available capacities, and storage is denoted as M_r and M_g for maximum capacities, and similar to the processing power capacities, the storage resources are also divided between the different slices which will be denoted as m_{r_s} and m_{g_s} for available capacities.

The proposed algorithm in this paper, VECSlic-LB, is assumed residing at the gNBs, since in 5G [31], gNBs can be allocated with a standardized 5G user plane function (UPF), a V2X application server, and allows applying network function virtualization frameworks on the gNBs data plane. Accordingly, the proposed algorithm emulates the services of an SDN controller in coordinating network

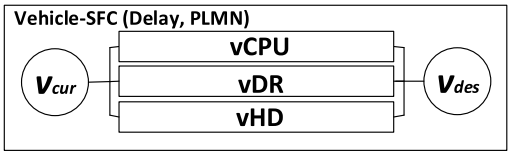

Fig. 2. SFC.

resources for edge-based applications and the lifecycle management of virtual network functions (VNFs) and network services. Therefore, it could be considered as a host operating system. In this way, VECSlic-LB, will represent the management and orchestration (MANO) center running the network's control plane, and will have an overall knowledge of the used and free resources at the edge computing servers to carryout the offloading and virtualization processes, and to activate or terminate the physical resources.

3.2. Service function chain request model

Demands of each vehicle v will be constructed as a service function chain requests, denoted as SFC^v shown in Fig. 2. Each SFC^v will be modeled as a weighted and directed graph $SFC^v = (N^v, L^v)$, where N^v and L^v are the sets of logical VNF nodes and their connecting logical links respectively. $VNF_n \in N^v$ is a VNF node n in the vehicle's v SFC, and l^v_{no} is a virtual logical link in L^v connecting VNFs n and o in SFC^v.

Each SFC^v will be represented by the total end-to-end delay threshold d_v, required public land mobile operator number v_{PLMN} (representing slice s in the RSUs or gNBs), demanded virtual processing power c^s_v (vCPU), demanded virtual storage m^s_v (vHD), demanded virtual data rate b_v (vNIC), and current vehicle location

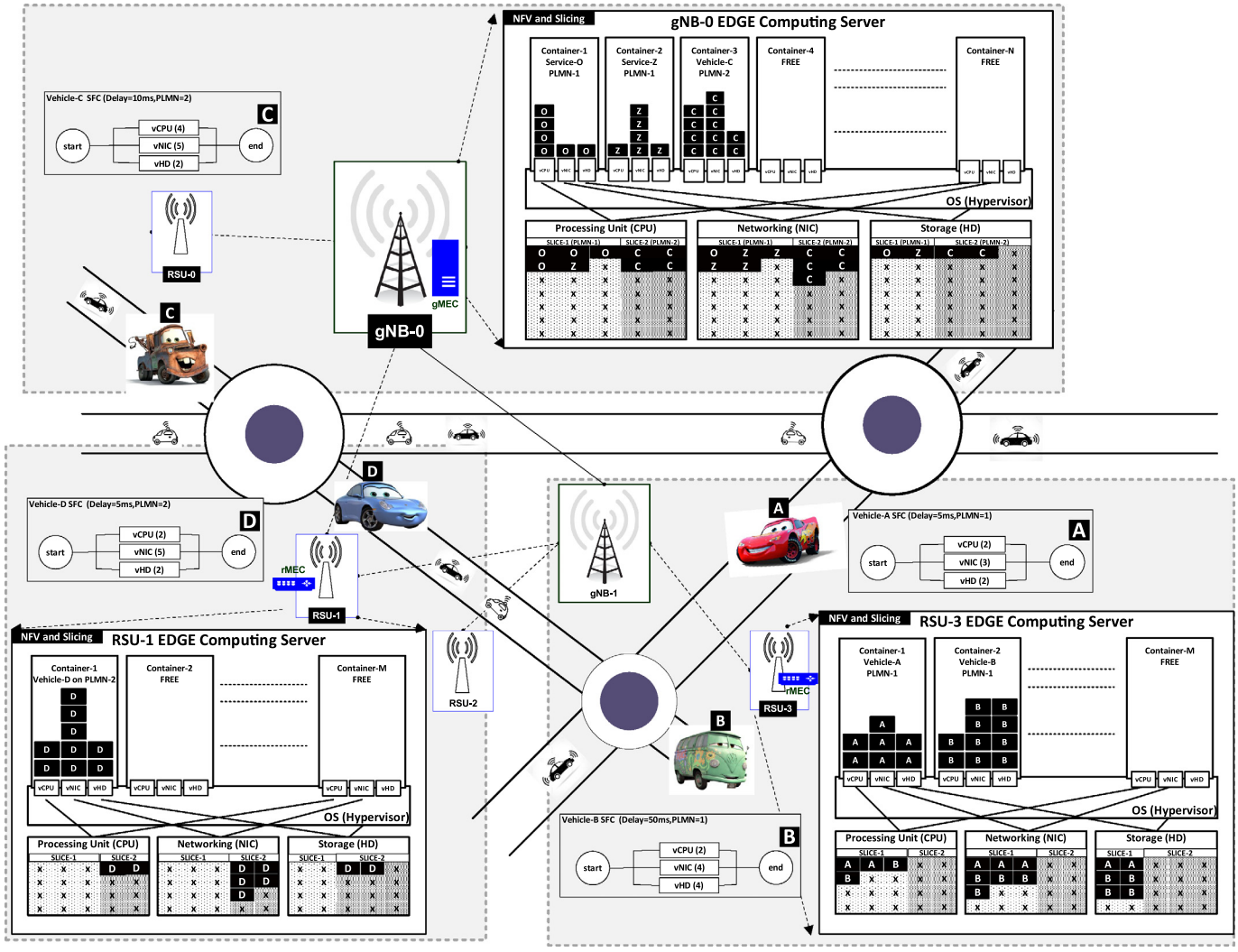

Fig. 3. Example.

v_{cur}, and destination location v_{des} as in Fig. 2 representing the vehicle's trajectory.

3.3. Demonstration example

An example to demonstrate the main activities that will be carried out by the proposed vehicular edge computing algorithm in this paper is shown in Fig. 3. Four vehicles are demanding virtual resources from the network, giving that their demands are constructed as shown in SFC^A, SFC^B, SFC^C, and SFC^D. For example vehicle A demands virtual 2 CPUs, 3 NIC, and 2 HD units to be allocated at PLMN-1 slices on RSU or gNB that has an end-to-end delay less than 5 ms. As shown in the figure, vehicle-A will be best served by RSU-3 since it is the nearest and will fit the 5 ms constraint. Therefore, Container-1 in the rMEC (multi-access edge computing at RSU 'r') of RSU-3 will translate the demands of vehicle-A and virtualize them on network slice-1 of PLMN-1. Same procedure for vehicle-B will be virtualized by Container-2 in the rMEC of RSU-3 on network slice-1 of PLMN-1. However, the demands of vehicle-C supposed to be served by RSU-0 giving that its the nearest to the vehicle, but since RSU-0 does not have rMEC service, the demands of vehicle-C will be hosted at Container-3 on the gMEC of gNB-0 on the network slice-2 of PLMN-2. Finally, vehicle-D arriving the network and the closest node to it happens to be RSU-1, which will host the demands of vehicle-D on Container-1 on the physical resources of slice-2 reserved for PLMN-2.

3.4. Problem formulation

In this paper the offloading problem from the vehicles towards other destinations in the context of vehicular edge computing environment is modeled as an integer linear programming (ILP) problem, of optimization objective function maximizing the total accepted offloading requests from the demanding vehicles.

3.4.1. Objective function and formulation

The objective function will target maximizing the number of accepted offloading tasks. Each task demands resources for offloading from vehicle $v \in V$ to RSU $r \in R$, or to gNB $g \in G$, located on specific path $P_{sd} \in Path^P$. Consequently, maximizing the acceptance ratio for each demand requires selecting either a slice s in specific RSU through activating binary variable $x^s_{vr} = 1$ to guarantee a slice was reserved in the RSU, or selecting a slice s in a gNB through activating binary variable $x^s_{vg} = 1$ to guarantee a slice was reserved in the gNB. Note that x^s_{vr} requires that the binary variable $x_{l_{vr}} = 1$ to ensure that the link between v and r is active, and x^s_{vg} requires that binary variable $x_{l_{vg}} = 1$ to ensure that the link between v and g is active too. For load-balancing feature the decision variables $x^{s'}_{vr}$ and $x^{s'}_{vg}$ must be activated and have a value equal to binary value one, to ensure that the next slice (s') in RSU r or gNB g is ready to handle the demands from the vehicles if the current slice s does not have free resources to host the vehicle's demand, giving that $s' \in S'$ represents the index of next slice to s. Note that

S' is a set that includes all the remaining slices than slice s in the selected node, i.e. $s \notin S'$

The mathematical formulation of the objective function is represented as follows:

$$\forall v \in V \; max \sum_{r \in R, g \in G} (x^s_{vr} + x^s_{vg}) \tag{1}$$

3.4.2. Constraints formulation

The solution of the objective function will be controlled by delay and data rate constraints for the links, processing and storage capacity constraints for the nodes, and domain constraints as presented below. Note that the constraints are designed to be generic and can support various V2X scenario:

1. *Constraints on the physical links*
   Checking delay between a vehicle and an RSU or a gNB: to ensure that end-to-end delay in the link connecting vehicle v in the selected path P_{sd}, d_{vr} for a vehicle towards an RSU, or d_{vg} for a vehicle towards a gNB is less than or equal to the demanded delay d_v by the vehicle, the following constraint must be fulfilled:

   for $v \in V$, for $r \in R$, $g \in G$

   $$(d_{vr} x_{l_{vr}} + d_{vg} x_{l_{vg}}) \leq d_v \tag{2}$$

   Reserving data rate capacity for a vehicle at an RSU or a gNB in the selected path P_{sd}: To ensure that the data rate to be reserved for a vehicle v through the network nodes formulating the selected P_{sd}, b_{vr} for data rate available for the vehicle v from an RSU, or b_{vg} from a gNB is at least equal or greater than the demanded data rate by vehicle v, b_v, the following constraint must be fulfilled:

   for $v \in V$, for $r \in R$, $g \in G$

   $$(b_{vr} x_{l_{vr}} + b_{vg} x_{l_{vg}}) \geq b_v \tag{3}$$

2. *Constraints on the physical nodes*
   To ensure that the demanded processing power for vehicle v is available at the hosting destination (an RSU or a gNB in the selected path P_{sd}), the following set of equations are formulated as follows:

   Reserving CPU capacity at an RSU: check if the demanded processing power by vehicle v, denoted as c^s_v, from the PLMN represented by the slice number s is less than or equal to the available processing power, denoted by c_{r_s}, at the desired RSU r slice s

   $\forall s \in S, \forall r \in R$

   $$c^s_v x^s_{vr} \leq c_{r_s} \tag{4}$$

   If load-balancing is activated (i.e. $x^{s'}_{vr} = 1$, where $s' \in S'$ represents another slice at the same RSU r reserved for another $PLMN'$), check if the demanded processing power by vehicle v, denoted by c^s_v, from the PLMN represented by the slice number s is less than or equal to the available processing power, denoted by $c_{r_{s'}}$, at the same RSU r on slice s' representing $PLMN'$.

   $\forall s \in S, \forall s' \in S', \forall r \in R$

   $$c^s_v x^{s'}_{vr} \leq c_{r_{s'}} \tag{5}$$

   Reserving CPU capacity at a gNB: If no RSU can satisfy the processing power demands, then check if the demanded processing power by vehicle v, denoted as c^s_v, from the PLMN represented by the slice number s is less than or equal to the available processing power, denoted by c_{g_s}, at the desired gNB g

   $\forall s \in S, \forall g \in G$

   $$c^s_v x^s_{vg} \leq c_{g_s} \tag{6}$$

   If load-balancing is activated (i.e. $x^{s'}_{vg} = 1$, where $s' \in S'$ represents another slice at the same gNB g reserved for another $PLMN'$), check if the demanded processing power by vehicle v, denoted by c^s_v, from the PLMN represented by the slice number s is less than or equal to the available processing power, denoted by $c_{g_{s'}}$, at the same gNB g on slice s' representing $PLMN'$

   $\forall s \in S, \forall s' \in S', \forall g \in G$

   $$c^s_v x^{s'}_{vg} \leq c_{g_{s'}} \tag{7}$$

   Reserving storage capacity at an RSU: check if the demanded storage capacity by vehicle v, denoted as m^s_v, from the PLMN represented by the slice number s is less than or equal to the available storage capacity, denoted by m_{r_s}, at the desired RSU r slice s

   $\forall s \in S, \forall r \in R$

   $$m^s_v x^s_{vr} \leq m_{r_s} \tag{8}$$

   If load-balancing is activated (i.e. $x^{s'}_{vr} = 1$, where $s' \in S'$ represents another slice at the same RSU r reserved for another $PLMN'$), check if the demanded storage capacity by vehicle v, denoted by m^s_v, from the PLMN represented by the slice number s is less than or equal to the available storage capacity, denoted by $m_{r_{s'}}$, at the same RSU r on slice s' representing $PLMN'$

   $\forall s \in S, \forall s' \in S', \forall r \in R$

   $$m^s_v x^{s'}_{vr} \leq m_{r_{s'}} \tag{9}$$

   Reserving storage capacity at a gNB: If no RSU can satisfy the storage capacity demands, then check if the demanded storage capacity by vehicle v, denoted as m^s_v, from the PLMN represented by the slice number s is less than or equal to the available storage capacity, denoted by m_{g_s}, at the desired gNB g

   $\forall s \in S, \forall g \in G$

   $$m^s_v x^s_{vg} \leq m_{g_s} \tag{10}$$

   If load-balancing is activated (i.e. $x^{s'}_{vg} = 1$, where $s' \in S'$ represents another slice at the same gNB g reserved for another $PLMN'$), check if the demanded storage capacity by vehicle v, denoted by m^s_v, from the PLMN represented by the slice number s is less than or equal to the available storage capacity, denoted by $m_{g_{s'}}$, at the same gNB g on slice s' representing $PLMN'$

   $\forall s \in S, \forall s' \in S', \forall g \in G$

   $$m^s_v x^{s'}_{vg} \leq m_{g_{s'}} \tag{11}$$

3. *Domain constraints*
   To ensure that the demands of a vehicle v are offloaded on only one RSU r or on one gNB g in P_{sd} the following constraint must be fulfilled:

   $$\forall v \in V \; \sum_{r \in R} x^s_{vr} + \sum_{g \in G} x^s_{vg} \leq 1 \tag{12}$$

To ensure that only one single link is activated between the vehicle v and one RSU r or one gNB g in P_{sd} the following constraint must be fulfilled:

$$\forall v \in V \sum_{r \in R} x_{l_{vr}} + \sum_{g \in G} x_{l_{vg}} \leq 1 \tag{13}$$

4. VECSlic-LB for offloading with network slicing and load-balancing

Optimal solution to solve the objective function in Eq. (1) subject to the constraints in Eq. (2)-Eq. (13) will be presented in the simulation in section 5.3, which implies allocating resources on the physical nodes (gNBs and RSUs) and links (connecting them to each other and to the vehicles) that are capable of meeting the demands of the vehicles. Theoretically, the optimal solution follows the strategy of introducing binary constraints to offload the demands of the SFC^v on one physical node, similar to the multi-dimensional Bin Packing problem [32]. Moreover, the optimal solution for Eq. (1) implies also connecting one link only for each node, and this is usually treated as a commodity between pairs of nodes, which is similar to finding an optimal flow for the commodity in any network model, and that was proved to be an NP-hard problem and not solvable in polynomial times even for small scale networks [32] as will be discussed in section 5.3.

Consequently, the majority of vehicular edge computing approaches followed heuristic or meta-heuristic algorithms to solve the optimization problem in a reasonable polynomial time. The following subsections will explain the proposed algorithm in this paper for vehicular edge computing supporting network slicing and load-balancing features, VECSlic-LB, which will be used to solve the problem of offloading tasks from vehicles towards edge computing servers hosted at the wireless gNBs or RSUs.

4.1. VECSlic-LB algorithm explained

VECSlic-LB heuristic is shown in Algorithm 1 and the flowchart in Fig. 4, which includes four major parts, initialization, ranking, offloading, and updating and evaluation.

4.1.1. Initialization

The physical network is constructed of an interconnected gNBs generated using Waxman generator [33], which produces random graphs using a probability function to interconnect any two gNBs based on the distance that separates them. Each gNB is allowed to connect to 3 RSUs max, and each RSU is connected to one gNB only. Each gNB and RSU will be constructed according to the NFV framework as shown in the example in Fig. 3.

Once the network of gNBs and RSUs are generated, VECSlic-LB will slice their processing power and memory capacities based on the PLMNs' required utilizations, then applies the path construction strategy developed by [13], and starts constructing and listing all the physical paths connecting the gNBs and RSUs as a source-to-destination path $P_{sd} \in Paths^P$. It is assumed that the locations of the physical network nodes (i.e. the gNBs and RSUs) are fixed. Therefore, the main elements formulating any path, such as number and connectivity of the nodes and links are also fixed and do not change, but only their capacities vary due to the consumption.

The initialization phase is performed in advance and ahead of handling any SFC. Consequently, VECSlic-LB algorithm will always have a full list of all the paths in the network, as well as a detailed information about the nodes and links of these paths, such as, number of nodes and links in the path, types of the nodes (gNBs or RSUs), maximum and consumed capacities of the resources of these nodes or their links, and end-to-end delay and data rate per each link in each physical path.

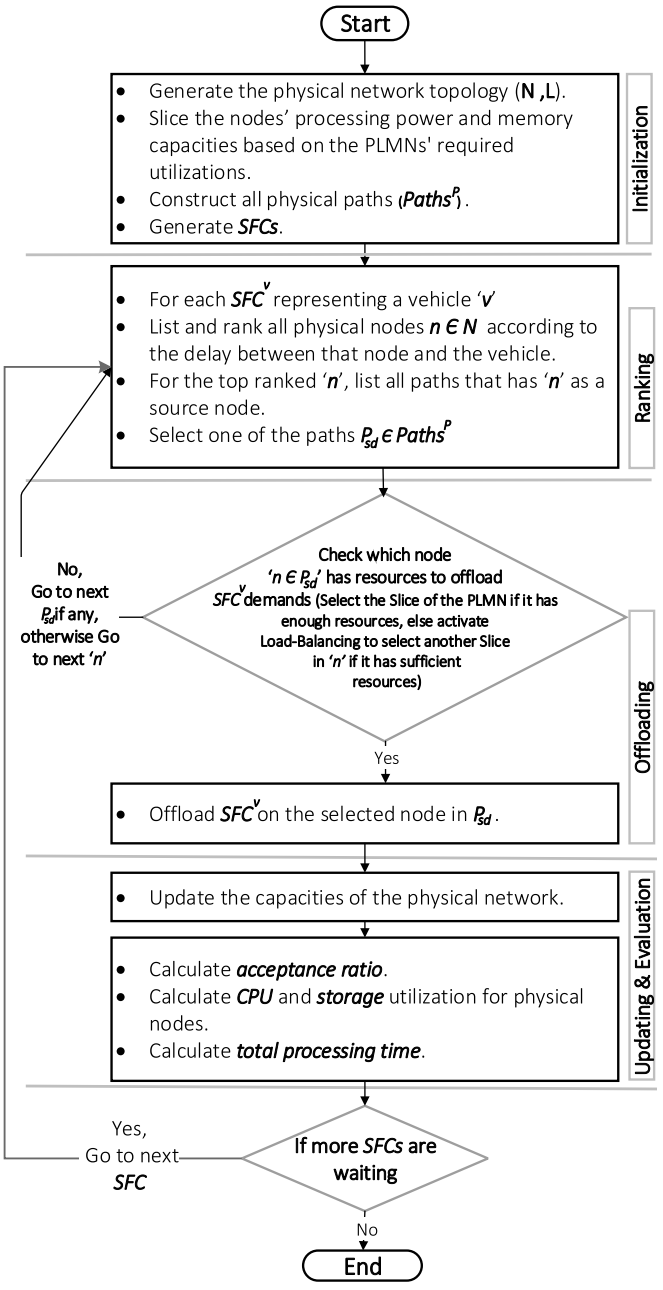

Fig. 4. VECSlic-LB Flowchart.

4.1.2. Ranking

In this phase VECSlic-LB algorithm lists all gNBs and RSUs, then lists all their PLMNs (i.e. all slices in the gNB or RSU), and identify the slice that was assigned to the required PLMN specified by the SFC^v. Next, VECSlic-LB calculates the delay using Eq. (18) between the vehicle and every gNB and RSU, which has the required PLMN among their listed PLMNs. Afterwards, VECSlic-LB will rank the gNBs and RSUs in descending order based on the shortest in delay to the vehicle v.

Accordingly, for each SFC^v representing the demands of $v \in V$, VECSlic-LB adopted (Bubble Sort) algorithm to sort and rank all physical network gNBs and RSUs in descending order [32] based on the least on delay towards the vehicle v, in addition to ranking the slices per each of these gNBs and RSUs based on their utilizations, which means that VECSlic-LB algorithm will have a quadratic computational time complexity in the order of $O(V * (N * S)^2)$, where $N \in P$ is the number of potential physical nodes for offloading the

SFCs, and S is number of slices per each node. Then once the gNBs and RSUs are sorted, the algorithm lists all the paths $P_{sd} \in Path^P$ which start by the top node (least in delay), and checks if a node in that path has a slice with enough resources to offload the traffic of SFC^v. This is done in computational time complexity in the order of $O(V * (O(|N^{P_{sd}}| + |L^{P_{sd}}|)))$ representing the order of complexity for ranking the paths (nodes and links) per each vehicle $v \in V$. Consequently, the total computational complexity of the VECSlic-LB to offload the traffic from all vehicles V is estimated to be in the order of $O(V * ((N * S)^2 + (O(|N^{P_{sd}}| + |L^{P_{sd}}|))))$ representing order of complexity for ranking the slices per nodes and paths per vehicles.

Algorithm 1 VECSlic-LB Pseudo-Code.

1. Input: P and V.
2. **For** the set of physical nodes $g \in G$ and $r \in R$ in P
   -Construct and list all physical paths $Paths^P$.
3. **List** all SFCs
4. **For** each SFC^v
   4.1- Select each gNB and RSU that has a slice for the demanded $PLMN$ by SFC^v.
   4.2- Calculate the delay between vehicle v and each gNB and RSU from (4.1) according to Eq. (18).
5. **Rank** the $gNBs$ and $RSUs$ from (4.2) based on the shortest in delay to v.
6. **For** the top ranked node in (5)
   6.1- List all physical paths $P_{sd} \in Paths^P$ which start by that node.
7. **For** the top listed path in (6.1), P_{sd}
   7.0- Start by the first node from the path.
   7.1– If delay and data rate constraints from Eq. (2) and Eq. (3), are satisfied, go to (7.2).
   – Else go to next path from (6.1) and continue to (7.0).
   7.2- Select the slice of the demanded PLMN
   – If CPU and storage constraints from Eq. (4), Eq. (6), Eq. (8), and Eq. (10) are satisfied, go to (8).
   – Else go to the next slice in (7.3).
   7.3- Rank all remaining slices in the selected node from (7.0) based on their utilizations.
   7.4- Select a slice starting by the top ranked one
   – If CPU and storage constraints from Eq. (5), Eq. (7), Eq. (9), and Eq. (11) are satisfied, go to (8).
   – Else go to next slice from (7.4).
   7.5– If no node in the selected path can satisfy the constraints, go to next path from (6.1)
   – Else go to the next node in (5).
8. **A suitable path and hosting node are found**
   - Allocate SFC^v on P_{sd} and
   - OFFLOADING is ACCEPTED.
   - Update CPU and storage in P_{sd} nodes.
   - Calculate the concerned evaluation metrics.
   - Go to (9).
9. **If** demands' list in (3) is not empty,
   Go to next SFC in (3).
10. **End**

4.1.3. Offloading

VECSlic-LB will start by selecting the top ranked node, then lists all the physical paths that start by that node without any specific ranking for the paths. Afterwards, VECSlic-LB will select the first path in the list, and starting by the first node in that path, it will check if it can satisfy the delay and data rate constraints in Eq. (2) and Eq. (3). Next, VECSlic-LB will check the slice belonging to the demanded PLMN in that node. If the node and the slice of the PLMN satisfy all the CPU and storage constraints in Eq. (4)-Eq. (11), VECSlic-LB will allocate the VNFs of SFC^v on the PLMN slice in that node, and the offloading is accepted.

Otherwise, if the PLMN slice has no resources to host SFC^v, VECSlic-LB will rank the remaining slices in the node based on their utilizations, and checks which one of them may have the demanded resources. If no slice can fulfill the demands of SFC^v, VECSlic-LB will jump to the next node in the path P_{sd}, do the same again until a slice is found. If no slice has resources and no other slice from the other nodes in the path can satisfy the demands, VECSlic-LB will jump to the next ranked path, and redo the checks again.

Important to clarify that VECSlic-LB will check if the path P_{sd} has multiple nodes between the vehicle and the hosting node (for CPU and storage), and will always ensure that the links connecting them satisfy the delay and data rate constraints.

However, in case no path can satisfy the demands of SFC^v, the algorithm jumps to the next ranked physical node and lists its paths, and the process keeps on going until no more SFC^v to be handled.

4.1.4. Updating and evaluation

Once a successful allocation occurs, the algorithm updates all changed resources on the hosting nodes and paths, calculates the evaluation criterion for acceptance ratio, utilization and processing times, then moves to the next SFC^v.

4.2. VECSlic-LB computational time complexity

Based on the size of the physical network P, VECSlic-LB constructs all types of paths in $O(|N| + |L|)$ processing time, considering the total number of nodes $N \in P$ and links $L \in P$ formulating the physical network [32]. This step is performed and saved only once before the arrival of any SFC, and it has no impact on the real computational time complexity of the Offloading process. However, to evaluate the computational time complexity of VECSlic-LB, the focal computational component of the heuristic is determined based on the time consumed while sorting all the listed gNBs and RSUs that belong to the demanded PLMN in the physical network.

4.3. VECSlic-LB algorithm running

The proposed VECSlic-LB algorithm is run offline in practice given the information on the vehicles' demands, which are assumed to be known or estimated using some historical demands. Moreover, VECSlic-LB algorithm allocates resources for an undetermined duration, and stops once the total capacities of the nodes (RSUs and gNBs) are achieved.

4.4. Evaluation metrics

4.4.1. Average acceptance ratio

Acceptance ratio (AR) represents how VECSlic-LB algorithm is performing and how successfully it managed to offload the demanded tasks from the vehicles. It is calculated by averaging and dividing the number of successfully offloaded vehicular demands (SFCs) by the total number of SFCs V.

$$AR = \frac{1}{V} \sum_{\forall v \in V} Accepted\ SFC^v \tag{14}$$

4.4.2. Processing power utilization

Represents the utilization trend of the gNBs and RSUs after each offloading attempt, denoted as CPU_{util}. It is defined as a ratio between consumed processing power given by $(C_n - c_n)$ and maximum processing power C_n of the physical node n, summed and averaged for all nodes in the physical network. c_n is the available processing resources in node $n \in N$.

$$CPU_{util} = \frac{1}{N} \sum_{\forall n \in N} \frac{(C_n - c_n)}{C_n} \tag{15}$$

Table 2
Settings.

Parameter	Values and Description.
R	3 RSUs per gNB.
G	15 gNB node sites.
S	1, 2, 4, 6, 8 Slices representing PLMNs.
$PLMN$	1, 2, 4, 6, 8 PLMNs, set of PLMNs.
V	1500 Set of all vehicles.
v_{PLMN}	$1-8$ Randomly selected for v PLMN.
v_{cur}, v_{des}	$0-1Km$.
c_v^s	15 Demanded CPU by vehicle v
C_r	100 Maximum CPU at each RSU r
C_g	300 Maximum CPU at each gNB g
m_v^s	15 Demanded storage by vehicle v
M_r	100 Maximum storage at each RSU r
M_g	300 Maximum storage at each gNB g.
b_v	10 Demanded data rate for vehicle v.
b_{vr}	By Eq. (16) data rate by RSU r.
b_{vg}	By Eq. (16) data rate by gNB g.
d_v	10 Demanded delay for vehicle v in ms.
d_{vr}	By Eq. (18) delay between v and r.
d_{vg}	By Eq. (18) delay between v and g.
W	$20MHz$ bandwidth of the wireless channel.
Pv	$23dBm$ transmission power of the vehicle.
No	$7dB$ variance of Gaussian noise.
h	$10m$ Antenna height of the gNB/RSU.
hv	$1.5m$ Antenna height of the vehicle.
fc	$2GHz$ carrier frequency in GHz.
α	0.7.
β	0.6.
p_{wax}	0.5.

5. Evaluation

In this section, we evaluate the performance of the proposed algorithm VECSlic-LB, by comparing it against the optimal solution, and the algorithm PaNFV proposed in [13] which does not consider network slicing feature.

5.1. Simulation settings

The simulation settings in this section are used for optimal solution, VECSlic-LB, and VECSlic without load-balancing algorithms. Accordingly, the physical network of gNBs topology was randomly generated using Waxman algorithm [33], setting $\alpha = 0.7$, $\beta = 0.6$, and mean probability of a pair of two gNBs being connected set equal to 0.5. The network includes 10 gNB nodes, each connected to 3 RSUs, giving that RSUs are connected to the gNBs only, and distributed on an urban area of 1 Km x 1 Km. The simulation handles 1500 vehicles that are distributed on the same area, and the distance between any vehicle v and RSU or gNB is less than 1 Km.

On network slicing, this paper generated the 1 slice simulations of PaNFV from [13], assigning the whole capacity of the $CPUs$ and MEM 100% as one slice for the gNB or RSU, supporting only one PLMN. For multiple slices, optimal, VECSlic-LB, and VECSlic without load-balancing supported 2 slices serving 2 PLMNs, dividing the resources of gNBs or RSUs as (60% for first slice assigned to the 1st PLMN, and 40% for the second PLMN), 4 slices for 4 PLMNs (35%, 25%, 20%, 20%), 6 slices for 6 PLMNs (30%, 25%, 20%, 10%, 10%, 5%), and 8 slices for 8 PLMNs (30%, 20%, 15%, 10%, 8%, 7%, 5%, 5%). For example, the 4th PLMN in the 8 slices configuration will be assigned 10% of the maximum capacity of the CPU and MEM in the hosting gNB or RSU. Table 2 summarizes all simulation parameters.

Maximum CPU and MEM for storage resources are given as real numbers, 300 for gNBs and 100 for RSUs, which are shared between the set of slices representing the PLMNs. For vehicles, the demanded CPU and MEM are set to 15 per each. The current data rate and delay between any gNB or RSU and a vehicle, will be calculated using the formulas in Eq. (16) and Eq. (18), which relay on multiple parameters including the Euclidean distance d between the vehicle and the gNB or RSU, carrier frequency fc, channel gain h_{V2N}, transmitted power by the vehicles Pv, channel bandwidth W, vehicle's antenna height hv, and gNB or RSU antenna height h. Finally, the demanded data rate and delay by the vehicles are set to 10.

To prove the case of vehicular edge computing and network slicing under controlled environment, and to show the strength of the proposed algorithm, this paper follows the offline scenario settings, where all demands from the vehicles are assumed known or estimated in advance. In addition to that, offline scenario will allow testing VECSlic-LB algorithm for variety of settings, and reliably calculate acceptance ratio, CPU utilizations, and the offloading times.

The performance of the offline algorithms in this paper was evaluated for consistency of results by running the simulations for 3 times. All parameters were fixed as shown in 2 to overcome any randomness. The target was to evaluate how efficient the algorithms were when repeating the offloading tasks for several times. For example, we focused on the results of the acceptance ratio for the offloading demands, and checked how the algorithm is calculating the paths and capacities each time, and in all cases, the algorithm resulted on the same results (or very close results) in terms of acceptance ratios. The only difference was on the time required by the algorithm when selecting the paths and the edge nodes capacities that can accommodate the requirements for each demand. Accordingly, the results shown represent the average of the 3 runs. The standard deviation between the acceptance ratio results were zero or near to zero for the 3 runs. Moreover, during the testing phase of the algorithms, we repeated the results for more than 10 times, and the results were as those of the 3 runs. Therefore, we reported the average results for the 3 runs to represent how stable the algorithm.

In all simulations, the two versions of VECSlic and the reference algorithm PaNFV, were developed using Eclipse IDE for Java Developers, version: Mars.2 Release (4.5.2). The used machine was Lenovo laptop, system model 20CLS2RG00, processor Intel(R) Core(TM) i7-5600U CPU, 2.60 GHz, 2 Cores, 4 logical processors, RAM 8 GB, and the operating system was Microsoft Windows 10 Enterprise.

5.2. Data rate and delay calculation

Available data rate by gNB or RSU for vehicle v is calculated by Eq. (16) which is the direct Shannon formula [34] as follows:

$$R = W.\log(1 + (Pv.h_{V2N})/No); \tag{16}$$

The channel gain h_{V2N} between the vehicle v towards gNB or RSU in the urban micro-cell wireless channel environment is calculated using Eq. (17) [35], where d in meters, is the distance between vehicle v and gNB or RSU:

$$h_{V2N} = 40log(d) + 9.45 - 17.3log(h) - 17.3log(hv) + 2.7log(fc/5) \tag{17}$$

Current delay in the link between v and gNB or RSU is calculated by Eq. (18):

$$d = (b_v/R); \tag{18}$$

5.3. Testing general behavior of VECSlic-LB

To evaluate VECSlic-LB with load-balancing feature activated, 5 tests were conducted and the results are shown in Fig. 5. The first test evaluated VECSlic-LB when the number of gNBs was changed

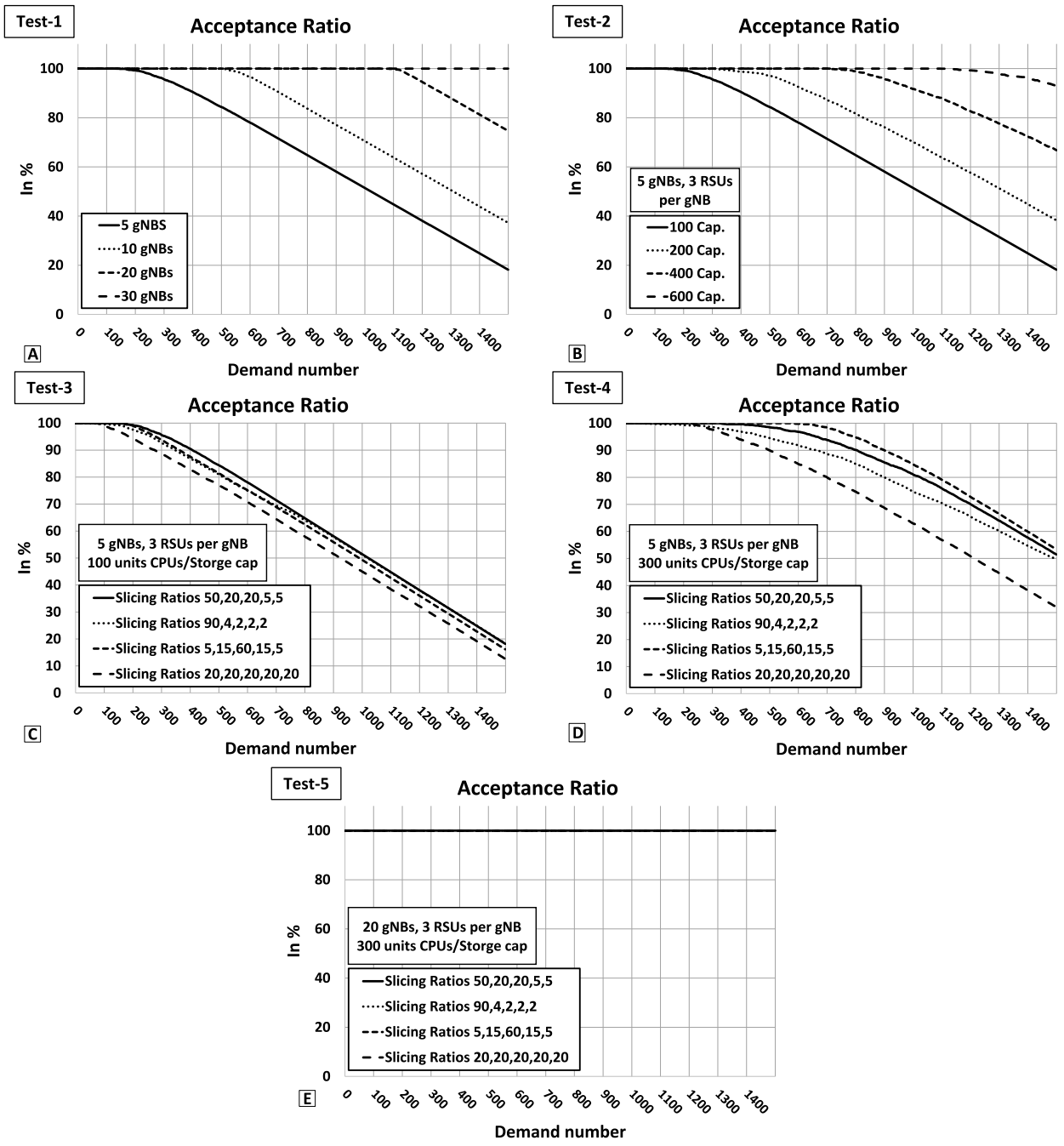

Fig. 5. Testing VECSlic-LB for multiple settings. (A) shows the results when number of gNBs and RSUs was fixed while changing their capacities, (B) shows impacts of changing gNBs and RSUs capacities, (C,D,E) show results due to changing slicing ratios.

between 5-30, while number of RSUs per each gNB was fixed to 3 only, and RSUs and gNBs CPU and storage capacities were fixed to 100 and 300 units per each respectively. 1500 demands were listed to be offloaded, each demanding a fixed 10 CPUs and 10 storage units. The acceptance ratio results in Fig. 5A shows that when number of gNBs was 5, VECSlic-LB provided least ratio scoring 18.2% after the last offloading demand was received. However, when the number of gNBs was increased to 10, acceptance ratio was 37.3%, for 20 gNBs it was much better scoring 74.8%, and when the number of gNBs was increased to 30, the acceptance ratio was 100%, indicating how VECSlic-LB can efficiently handle large number of nodes to reach an outstanding performance.

The second test fixed the number of gNBs to 5 and number of RSUs to 3 per each gNB, and evaluated the impacts of increasing the RSUs' CPU and storage capacities between 100-600 units, and for gNBs between 300-1800 units. The acceptance ratios of VECSlic-LB are shown in Fig. 5B, indicating that the higher the capacities the better the acceptance ratios. For example when comparing between the RSUs capacities of 600 units, VECSlic-LB scored 93%, while for 100 units VECSlic-LB provided 18% acceptance ratio.

For Test-3, Test-4, and Test-5, the aim was to evaluate the impacts of changing the slicing ratios on VECSlic-LB's acceptance ratio. Consequently, in Fig. 5C RSUs capacities were set to 100 units, while fixing the number of gNBs to 5, each is serving 3 RSUs.

Table 3
Comparing VECSlic-LB to PaNFV.

Item	VECSlic-LB	PaNFV
Scenario	Offline	Offline and Migration (optional)
Resource allocation	Offloading	Offloading, Caching (optional)
Edge Computing Support	Yes	Yes, plus cloud computing (optional)
Network Slicing Support	Multiple Slices	Single slice
Load-Balancing Support	Yes	No
Path Construction	As in [13]	Uses special technique called segmentation.
Ranking	Rank nodes on their distance from v	Rank nodes on their distance from v.
Offloading	All VNFs on one single node	All VNFs on one single node
End-to-end delay	Yes	Yes

Regardless of the different slicing ratios, the acceptance ratios were near to each other providing poor results for all of them. On the other hand, Fig. 5D provided much better results for the same slicing ratios, mainly due to increasing the capacities of the RSUs to 300 units, which allowed for more demands to be offloaded.

The best results were obtained from Test-5, when the number of gNBs was increased to 20, while fixing the capacities of the RSUs to 300, gNBs to 900, and connecting 3 RSUs to each gNB. Accordingly Fig. 5E shows VECSlic-LB managed to score 100% acceptance ratios regardless of the various slicing ratio.

Overall, these tests explored the various behaviors of VECSlic-LB when some of its main parameters were changed, highlighting its general capabilities, and how it can treat multiple scenarios and environments. The main outcome from these tests is that in terms of implementation costs, changing the capacities of the gNBs and RSUs are much efficient than adding more gNBs or RSUs, or changing the slicing ratios as well.

5.4. Optimal performance results

In order to validate the ultimate performance of the proposed algorithm with network slicing and load-balancing features, the following paragraphs will compare VECSlic-LB performance to the optimal solution of Eq. (1) subject to the constraints in Eq. (2)-Eq. (13), and to VECSlic without including the load-balancing feature. The simulations were conducted for 2, 4, 6, and 8 slicing configurations.

The optimal solution was solved by calling IBM ILOG CPLEX optimizer version 12.10.0.0, which is a mathematical programming solver for optimization problems, such as linear programming. It provides the best solution of the objective function in Eq. (1), which corresponds to a suitable RSU or gNB that can host the demands of a vehicle.

5.4.1. Acceptance ratio

The final accumulated acceptance ratios for optimal, VECSlic-LB, and VECSlic without load-balancing are shown in Fig. 6. For 2 slices configuration, the ratios for the optimal and VECSlic-LB were very close to each other, scoring around 48% after handling all demands, while VECSlic without load-balancing slightly differed from them around the demands $568 - 860$, but resulted on acceptance ratio around 48% as well. Notice that, both the optimal and VECSlic-LB started to degrade in their acceptance ratios after demands 701 and 714 respectively.

Similar trends were reported for 4 slices configurations, noting that the acceptance ratios for optimal solution, VECSlic-LB, and VECSlic without load-balancing resulted on 44.8%, 44.46%, and 44.4% respectively. They deferred on the starting point to degrade the acceptance ratio, where the optimal and VECSlic-LB started degrading around demands 627 and 615, and VECSlic started degrading around demand 535.

For the 6 slices configurations, the optimal solution scored acceptance ratio around 42.5% and started degrading around demand 583, VECSlic-LB acceptance ratio was around 42.1% but started degrading around demand 553, and VECSlic scored 41.3% acceptance ratio and started degrading so early around demand 152. Finally, the behavior of optimal, VECSlic-LB, and VECSlic for 8 slices configurations reported acceptance ratios around 40%, 39.2%, and 37.4%, giving that they started degrading around demands 488, 451, and 92 for the three of them respectively.

Overall, the simulation results of VECSlic-LB were much closer to the optimal solution than VECSlic without load-balancing, this is most likely due to the systematic behavior of VECSlic-LB on selecting the best nodes and identifying the slice among each node that has availability to host more demands. Results of VECSlic without load-balancing for 6 and 8 slices were much lower than VECSlic-LB, mainly due to limited capacities assigned to multiple smaller slices. Therefore, while attempting to offload the demands requesting specific PLMN, VECSlic without load-balancing may deny some offloading attempt due to congested slices, accordingly, its acceptance ratios will be less than VECSlic-LB.

5.4.2. CPU utilizations

Behavior of optimal solution and VECSlic-LB was generally near to each other when utilizing the different slices as shown in Fig. 6E-H. More specifically, the final utilization rates for 2 slices configuration resulted on 100% for each of optimal, VECSlic-LB, and VECSlic without load-balancing. For 4 slices the final utilizations were around 93%, in 6 slices configuration it was around 70%, but for 8 slices, optimal and VECSlic-LB scored 56, while VECSlic without load-balancing had CPU utilizations around 54%.

These results clearly show that, in terms of utilizations ratio the proposed algorithm VECSlic-LB performed on very similar trends near the optimal solution results, reflecting its solid and stable performance when handling large number of slices for multiple nodes.

5.5. VECSlic-LB comparison to the reference PaNFV

In this subsection, the overall performance of the proposed algorithm VECSlic-LB with load-balancing, and VECSlic without load-balancing feature will be compared to the reference algorithm PaNFV. The presented results are for 2, 4, 6, and 8 slices configurations, while PaNFV results are for 1 slice. Table 3 provides a high-level comparison between VECSlic-LB and PaNFV algorithms.

5.5.1. Acceptance ratio

The final accumulated acceptance ratios for VECSlic without load-balancing compared to PaNFV are shown in Fig. 7A. For 2 slices VECSlic resulted on the same typical ratios as those of PaNFV scoring 48%, but provided less ratios for 4 slices by 3.6%, for 6 slices by 7.6%, and 10.6% for 8 slices. However, Fig. 7B shows the results for VECSlic-LB which provided better results than VECSlic without load-balancing compared to the reference PaNFV, giving that for 2 slices configuration both had 48% acceptance ratio, but for 4 slices VECSlic-LB was less than PaNFV by 3.2%, for 6 slices was less by 6.2%, and 9.3% for 8 slices.

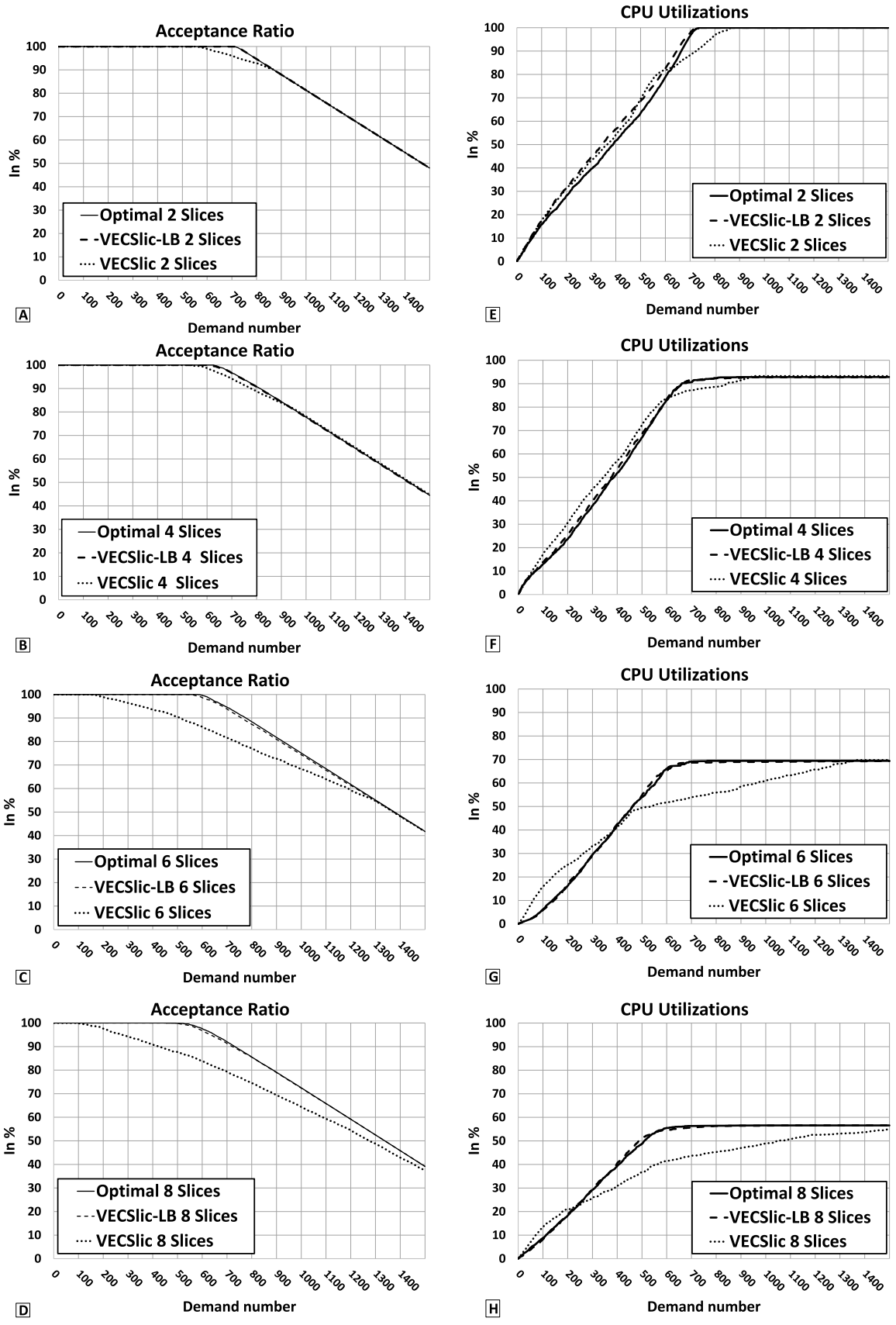

Fig. 6. Results of the optimal solution compared to the proposed algorithm VECSlic-LB, and VECSlic without load-balancing. A-D show the acceptance ratios for 2, 4, 6, and 8 slicing configurations, and E-H present the CPU utilization for the same slicing configurations.

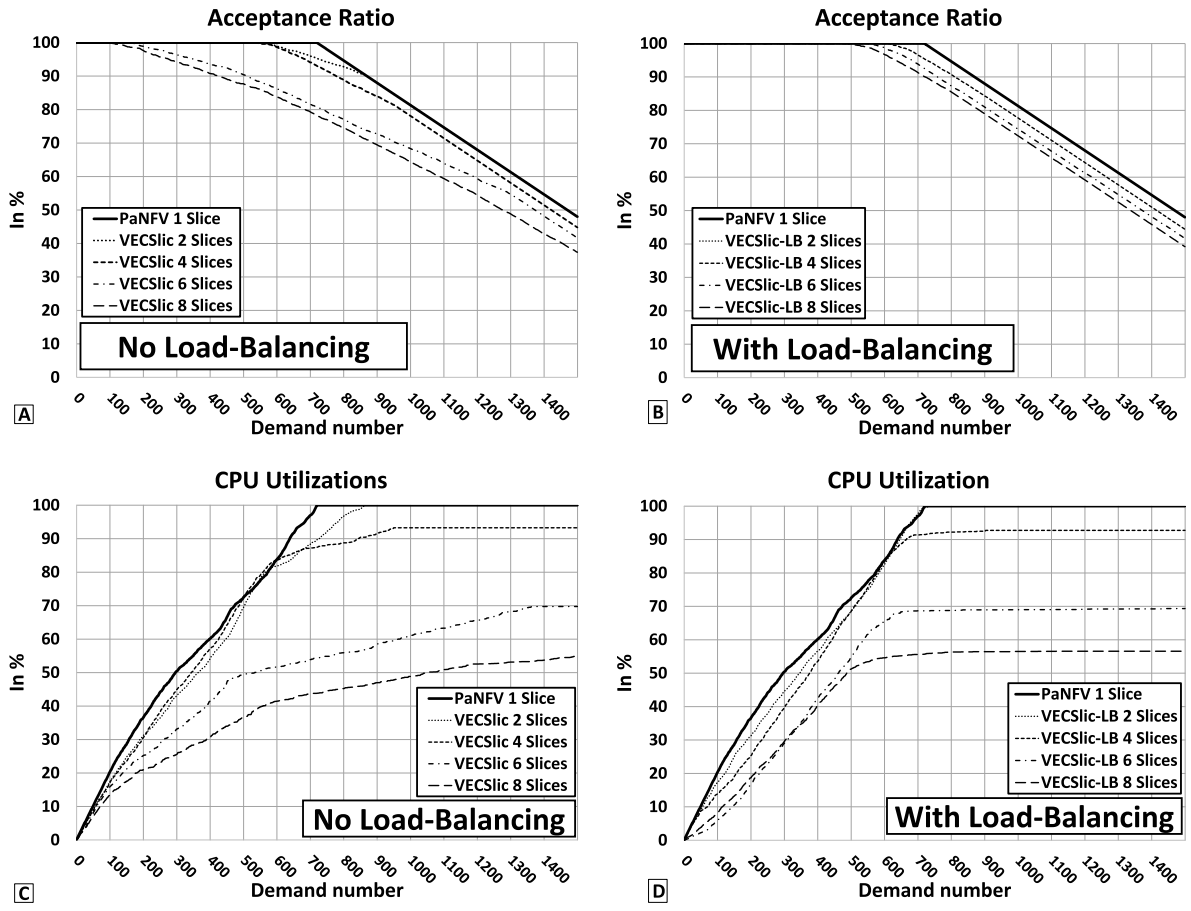

Fig. 7. Comparing VECSlic with and without load balancing feature against the reference algorithm PaNFV.

Compared to PaNFV, the results of VECSlic-LB and VECSlic without load-balancing were mostly near those of PaNFV, but with slight difference regarding the point when the acceptance ratio started to decay. However, as a general observation from the simulations, the results of VECSlic-LB which has slicing and load-balancing features showed great reliability in balancing the loads between the various slices in the edge nodes, and that should be reflected positively on managing the edge node's processing and memory resources for real edge computing applications.

5.5.2. CPU utilizations

The great advantage of VECSlic-LB over the reference algorithm PaNFV is demonstrated in Fig. 7D, showing how efficient is VECSlic-LB in utilizing the CPU resources. For example, compared to PaNFV which supports single slice configuration only and had 100% CPU utilization, VECSlic-LB for 8 slice configuration utilized around 52% of the nodes' CPU resources, leaving space for free resources around 48% to handle more demands.

This means that if more vehicular demands than the 1500, request offloading service, PaNFV will immediately reject them, since the CPU resources under PaNFV were full, but VECSlic-LB will have much better chances to accept the offloading request since it has more free resources. The same analysis applies to 6 and 4 slice configurations, and to VECSlic without load-balancing in Fig. 7C.

The rational for why VECSlic-LB had such great lead over the reference algorithm in utilizing the CPU resources (the same for VECSlic without load-balancing), is because of the network slicing feature mainly. Referring to acceptance ratio results, it was obvious that PaNFV managed to accept more demands than VECSlic-LB for 8 slices configuration by around 10%, but when evaluating the future capabilities of both algorithms based on the efficiency of resources utilizations, VECSlic-LB support for network slicing clearly leads, due to its superior advantage in handling more demands than PaNFV, which will be translated into better acceptance ratio as well.

5.6. Offloading distributions per gNBs and RSUs

The rational of this experiment is to gather some statistical analysis about the performance of VECSlic-LB when it offloads various tasks on the gNBs and RSUs. VECSlic-LB algorithm strategy in selecting candidate gNB or RSU to host the offloaded traffic from the vehicles, relies on selecting the node that has the shortest delay towards the vehicle, and has free resources within its slices, otherwise it moves to the next shortest node in terms of delay if any.

Fig. 8A,C show the results of VECSlic without load-balancing for 2 and 8 slices configuration, clarifying the distribution of the successful offloading attempts on gNBs or RSUs that have free hosting resources. The figures indicate that in the initial attempts, VECSlic mostly leaned towards RSUs, then if some of the RSUs became congested, it also started to offload on the gNBs since they have more capacities. Notice that once the network becomes more loaded, VECSlic spread the offloading attempts towards the remaining RSUs and gNBs that still have free capacities, even if they were lower ranked in terms of delay, but still comply with the delay and data rate constraints.

VECSlic-LB results on the other hand highlight that the offloading distributions are more condensed than VECSlic without load-balancing as shown in Fig. 8B,D. Overall, the figures show

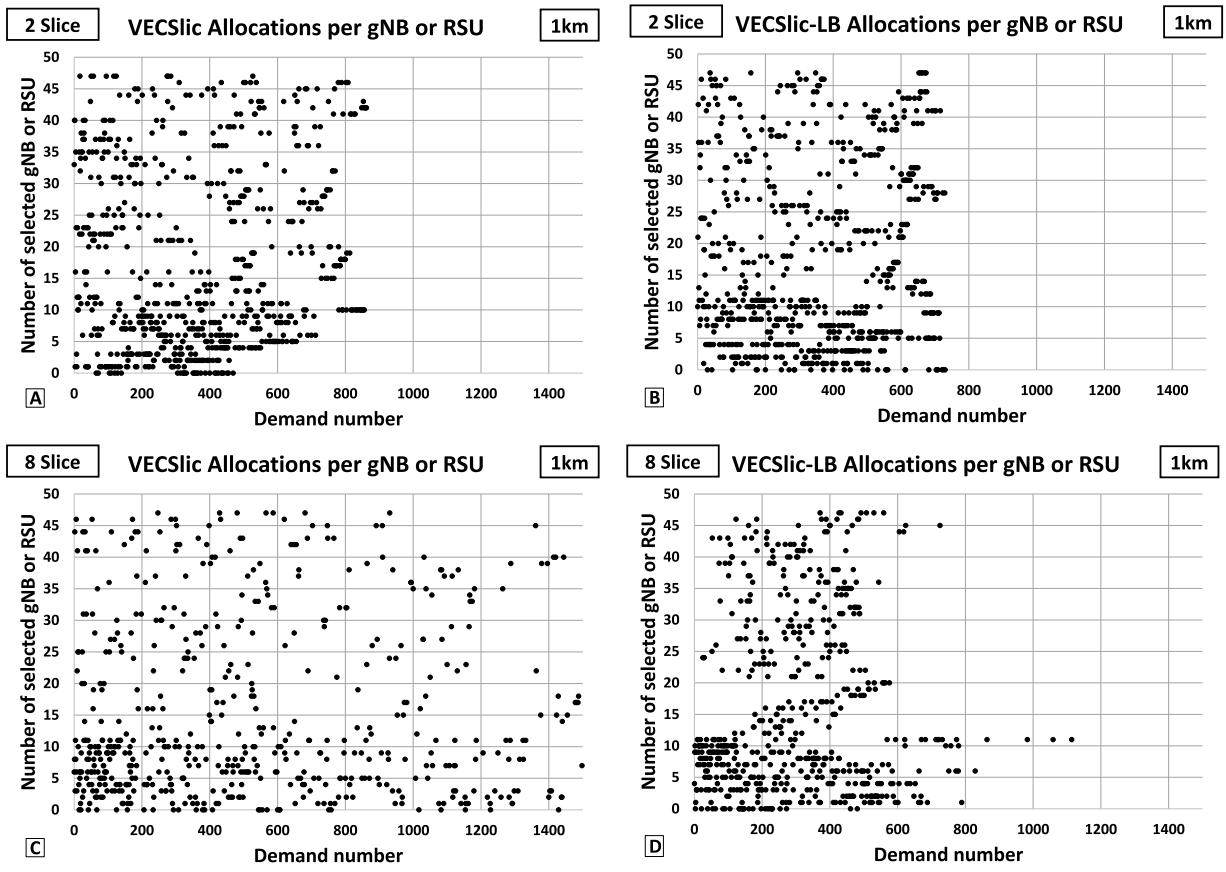

Fig. 8. Distribution of successful offloading attempts on gNBs and RSUs. In (A,C) the results of VECSlic without load-balancing for 2 and 8 slices configurations, and in (B,D) are the results for VECSlic-LB for 2 and 8 slices configurations.

that from the beginning VECSlic-LB offloading attempts leaned a bit more towards the gNBs than RSUs for the 2 and 8 slices configurations, giving the activated load-balancing feature from the start.

The rational of these results offer direct explanation for the offloading phase of VECSlic-LB algorithm, which clarify that VECSlic-LB will first keep selecting the nodes that have least delay towards the vehicles and has free resources, therefore, most nodes will start filling-in early on. Afterwards, VECSlic-LB will start offloading to the other few remaining nodes, which could be further away from the vehicle, yet may have free CPU capacities to offload vehicles' traffic to them. Note that VECSlic-LB does not force any distance thresholds, but only ranks these nodes based on their CPU utilizations and delays from the vehicles, and checks if their delays comply with the demanded delay. If no more nodes could fulfill the demands, VECSlic-LB will drop the demand.

5.7. Average offloading times

The average offloading times For the 2 and 8 slicing configurations are shown in Fig. 9. Overall, VECSlic without load-balancing was much faster in performing the whole offloading attempts compared to VECSlic-LB, mainly because with load-balancing, VECSlic-LB needed to rank the slices per node based on their CPU utilizations, in addition to ranking the nodes themselves based on their transmission delay from the vehicles, reflecting the impacts of activating the load-balancing feature.

In summary, these results suggest that the benefits of load-balancing in accepting more demands and saving more resources could be comprised due to the longer offloading times required to handle the demands from the vehicles.

Moreover, the proposed algorithm provides better results than the optimal solution and this holds for both schemes, namely with and without load balancing. Therefore, the optimality loss incurred by the proposed algorithm, in terms of acceptance ratio, is well compensated for by the offloading times.

6. Conclusions

This paper presented a vehicular edge computing algorithm for offloading tasks from vehicles towards wireless nodes hosting edge computing servers. The algorithm is denoted as VECSlic-LB, supports network slicing and load-balancing features, and adopts centralized control plane and virtualized data plane based on network function virtualization architecture. The performance of the proposed algorithm compared to the optimal solution, resulted on close results, confirming VECSlic-LB reliable and stable performance. The algorithm performed the offloading tasks in fraction of a second for different slicing configurations, and compared to state-of-art algorithm, VECSlic-LB provided more efficient results in terms of resources utilization by 48%, thanks to the integration of slicing and load-balancing features which allowed the proposed algorithm to handle large number of slices and manage their resources in a reliable manner.

In future work we will develop another version of the algorithm for online scenarios including network slicing for offloading and caching services at the same time, while adding comparison with other VEC offloading algorithms with and without slicing. We will also investigate other channel and mobility models.

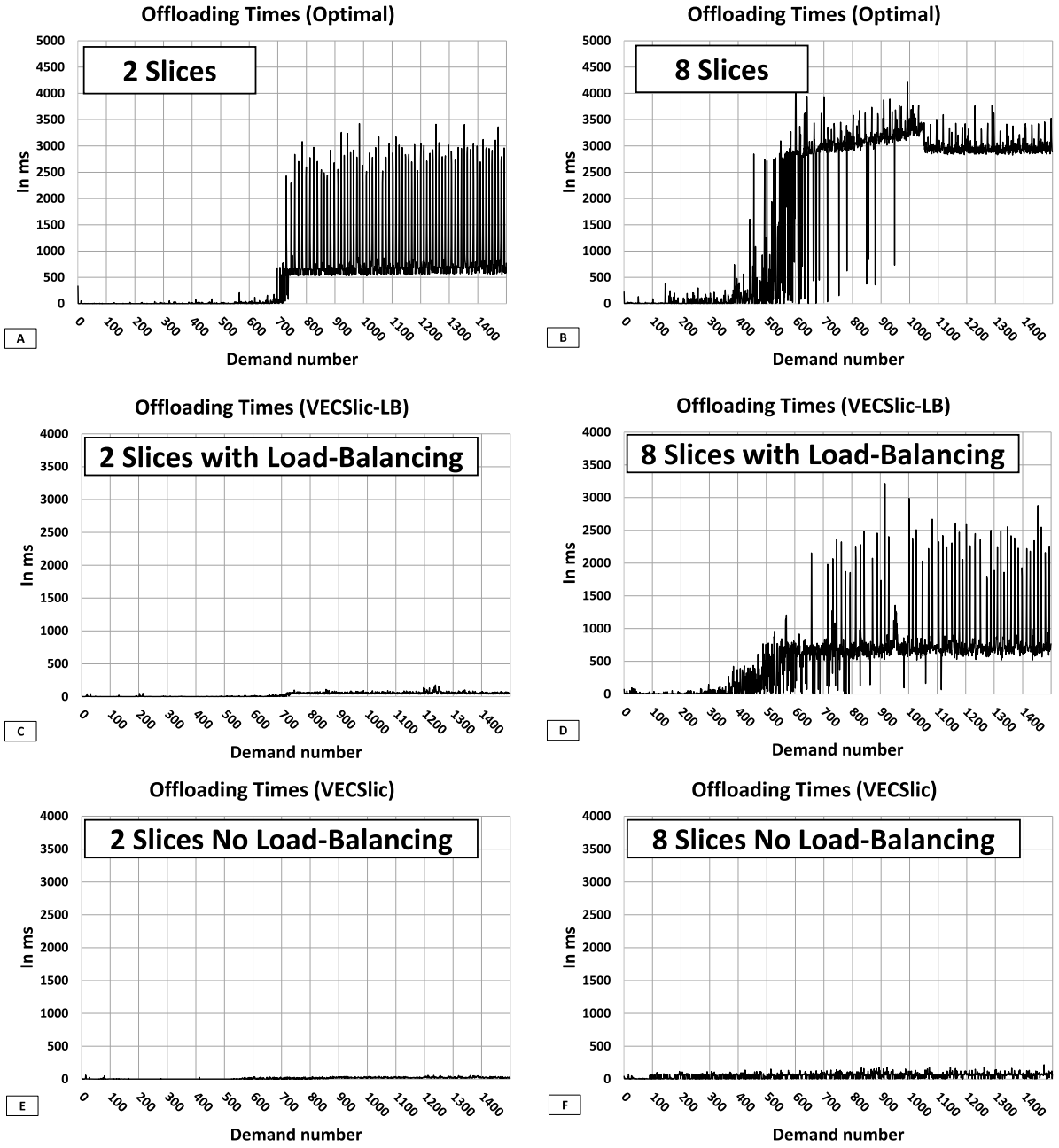

Fig. 9. VECSlic's average offloading times for the demands from vehicles. Optimal solution and the proposed schemes with load-balancing and without load-balancing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work is supported by the EU project InDiD (Infrastructure Digitale de Demain) co-financed by the connecting Europe facility of the European Union.

References

[1] R16-3GPP TS 23.285 V16.2.0 (2019-12), Architecture enhancements for V2X services, R16-3GPP TS 23.285.

[2] 3GPP TS 22.186 V16.2.0 (2019-06), Enhancement of 3GPP support for V2X scenarios, Stage-1, R16, 3GPP TS 22.186.

[3] 5G Automotive Association (2017-12), Toward fully connected vehicles: Edge computing for advanced automotive communications, Version 1.0, 5GAA T-170219-White Paper Edge Computing.

[4] 3GPP TS 23.501 V16.3.0 (2019-12), 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; System architecture for the 5G System (5GS); Stage 2 (Release 16), 3GPP TS 23.501.

[5] R17-3GPP TR 23.764 V0.3.0 (2019-12), Study on enhancements to application layer support for V2X services, 3GPP TR 23.764.

[6] 5G Automotive Association (2019-06), White Paper, C-V2X Use Cases: Methodology, Examples, and Service Level Requirements, Version 1.0, White paper on C-V2X Use Cases v.1.1.

[7] Fifth Generation Communication Automotive Research and innovation (2019-03), Deliverable D4.2 Final Design and Evaluation of the 5G V2X System Level Architecture and Security Framework, Version 1.0, 5GCAR.

[8] U. Bulkan, T. Dagiuklas, M. Iqbal, K.M.S. Huq, A. Al-Dulaimi, J. Rodriguez, On the load balancing of edge computing resources for on-line video delivery, IEEE Access 6 (2018) 73916–73927, https://doi.org/10.1109/ACCESS.2018.2883319.

[9] S. Razzaghzadeh, A. Habibizad, A. Masoud, M. Hosseinzadeh, Probabilistic modeling to achieve load balancing in Expert Clouds, Ad Hoc Netw. 59 (2017) 12–23, https://doi.org/10.1016/j.adhoc.2017.01.001, ISSN 1570-8705.

[10] D. Kesavaraja, A. Shenbagavalli, QoE enhancement in cloud virtual machine allocation using Eagle strategy of hybrid krill herd optimization, J. Parallel Distrib. Comput. 118 (Part 2) (2018) 267–279, ISSN 0743-7315, https://doi.org/10.1016/j.jpdc.2017.08.015.

[11] S. Kassir, G.d. Veciana, N. Wang, X. Wang, P. Palacharla, Service placement for real-time applications: rate-adaptation and load-balancing at the network edge, in: 2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom), New York, NY, USA, 2020, pp. 207–215.

[12] 3GPP TR 28.801 (V15.0.0), 2017, Study on management and orchestration of network slicing for next generation network, 3GPP TR 28.801.

[13] Khaled Hejja, Xavier Hesselbach, Offline and online power aware resource allocation algorithms with migration and delay constraints, Comput. Netw. 158 (2019) 17–34, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2019.04.030.

[14] Khaled Hejja, Xavier Hesselbach, Evaluating impacts of traffic migration and virtual network functions consolidation on power aware resource allocation algorithms, Future Gener. Comput. Syst. 101 (2019) 83–98, ISSN 0167-739X, https://doi.org/10.1016/j.future.2019.06.015.

[15] H. Zhou, H. Wang, X. Chen, X. Li, S. Xu, Data offloading techniques through vehicular ad hoc networks: a survey, IEEE Access 6 (2018) 65250–65259, https://doi.org/10.1109/ACCESS.2018.2878552.

[16] Salman Raza, Shangguang Wang, Manzoor Ahmed, Muhammad Rizwan Anwar, A survey on vehicular edge computing: architecture, applications, technical issues, and future directions, Wirel. Commun. Mob. Comput. 2019 (2019) 3159762, https://doi.org/10.1155/2019/3159762.

[17] L.U. Khan, I. Yaqoob, N.H. Tran, S.M.A. Kazmi, T.N. Dang, C.S. Hong, Edge computing enabled smart cities: a comprehensive survey, IEEE Int. Things J. 7 (10) (2020) 10200–10232, https://doi.org/10.1109/JIOT.2020.2987070.

[18] Chung-Ming Huang, Shih-Yang Lin, Zhong-You Wu, The k-hop-limited V2V2I VANET data offloading using the Mobile Edge Computing (MEC) mechanism, Veh. Commun. 26 (2020), https://doi.org/10.1016/j.vehcom.2020.100268.

[19] W. Zhan, et al., Deep reinforcement learning-based offloading scheduling for vehicular edge computing, IEEE Int. Things J. 7 (6) (2020) 5449–5465, https://doi.org/10.1109/JIOT.2020.2978830.

[20] S. Buda, S. Guleng, C. Wu, J. Zhang, K.A. Yau, Y. Ji, Collaborative vehicular edge computing towards greener ITS, IEEE Access 8 (2020) 63935–63944, https://doi.org/10.1109/ACCESS.2020.2985731.

[21] S. Li, S. Lin, L. Cai, W. Li, G. Zhu, Joint resource allocation and computation offloading with time-varying fading channel in vehicular edge computing, IEEE Trans. Veh. Technol. 69 (3) (March 2020) 3384–3398, https://doi.org/10.1109/TVT.2020.2967882.

[22] R. Su, et al., Resource allocation for network slicing in 5G telecommunication networks: a survey of principles and models, IEEE Netw. 33 (6) (Nov.-Dec. 2019) 172–179, https://doi.org/10.1109/MNET.2019.1900024.

[23] Alex Barakabitze, Arslan Ahmad, Rashid Mijumbi, Andrew Hines, 5G network slicing using SDN and NFV: a survey of taxonomy, architectures and future challenges, Comput. Netw. 167 (2020), https://doi.org/10.1016/j.comnet.2019.106984.

[24] H. Pydi, G.N. Iyer, Analytical review and study on load balancing in edge computing platform, in: 2020 Fourth International Conference on Computing Methodologies and Communication, ICCMC, Erode, India, 2020, pp. 180–187.

[25] J. Zhang, H. Guo, J. Liu, Y. Zhang, Task offloading in vehicular edge computing networks: a load-balancing solution, IEEE Trans. Veh. Technol. 69 (2) (Feb. 2020) 2092–2104, https://doi.org/10.1109/TVT.2019.2959410.

[26] Y. Dai, D. Xu, S. Maharjan, Y. Zhang, Joint load balancing and offloading in vehicular edge computing and networks, IEEE Int. Things J. 6 (3) (June 2019) 4377–4387, https://doi.org/10.1109/JIOT.2018.2876298.

[27] S. Agarwal, A. Das, N. Das, An efficient approach for load balancing in vehicular ad-hoc networks, in: 2016 IEEE International Conference on Advanced Networks and Telecommunications Systems, ANTS, Bangalore, 2016, pp. 1–6.

[28] V. Eramo, E. Miucci, M. Ammar, F.G. Lavacca, An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures, IEEE/ACM Trans. Netw. 25 (4) (2017) 2008–2025, https://doi.org/10.1109/TNET.2017.2668470.

[29] M. Chowdhury, M. Rahman, R. Boutaba, ViNEYard: virtual network embedding algorithms with coordinated node and link mapping, IEEE/ACM Trans. Netw. 20 (1) (2012) 206–219, https://doi.org/10.1109/TNET.2011.2159308.

[30] Z. Zhang, S. Su, J. Zhang, K. Shuang, P. Xu, Energy aware virtual network embedding with dynamic demands: online and offline, Comput. Netw. 93 (2015) 448–459, https://doi.org/10.1016/j.comnet.2015.09.036.

[31] R16-3GPP TS 23.287 V16.3.0 (2020-07), Architecture enhancements for 5G System (5GS) to support Vehicle-to-Everything (V2X) services, R16-3GPP TS 23.287.

[32] J. Kleinberg, E. Tardos, Algorithms Design, Addison-Wesley, 2009.

[33] B.M. Waxman, Routing of multipoint connections, IEEE J. Sel. Areas Commun. 6 (9) (Dec. 1988) 1617–1622, https://doi.org/10.1109/49.12889.

[34] W.C.Y. Lee, Estimate of channel capacity in Rayleigh fading environment, IEEE Trans. Veh. Technol. 39 (3) (Aug. 1990) 187–189, https://doi.org/10.1109/25.130999.

[35] P. Kyosti, et al., IST-4-027756 WINNER II D1.1.2 v. 1.1: WINNER II Channel Models, Tech. Rep., 2007, IST-4-027756 WINNER II D1.1.2 v.1.1: WINNER II Channel Models.