Premsankar, Gopika; Ghaddar, Bissan; Di Francesco, Mario; Verago, Rudi

Efficient placement of edge computing devices for vehicular applications in smart cities

Published in:
IEEE/IFIP Network Operations and Management Symposium

DOI:
10.1109/NOMS.2018.8406256

Published: 01/01/2018

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Efficient Placement of Edge Computing Devices for Vehicular Applications in Smart Cities

Gopika Premsankar*, Bissan Ghaddar†, Mario Di Francesco* and Rudi Verago‡
*Department of Computer Science, Aalto University, Finland
†Department of Management Sciences, University of Waterloo, Canada
‡IBM Research – Ireland

***Abstract*—Vehicular applications in smart cities, including assisted and autonomous driving, require complex data processing and low-latency communication. An effective approach to address these demands is to leverage the edge computing paradigm, wherein processing and storage resources are placed at access points of the vehicular network, i.e., at roadside units (RSUs). Deploying edge computing devices for vehicular applications in urban scenarios presents two major challenges. First, it is difficult to ensure continuous wireless connectivity between vehicles and RSUs, especially in dense urban areas with many buildings. Second, edge computing devices have limited processing resources compared to the cloud, thereby requiring careful network planning to meet the computational and latency requirements of vehicular applications. This article specifically addresses these challenges. In particular, it targets efficient deployment of edge computing devices in an urban scenario, subject to application-specific quality of service constraints. To this end, this article introduces a mixed integer linear programming formulation to minimize the deployment cost of edge devices by jointly satisfying a target level of network coverage and computational demand. The proposed approach is able to accurately model complex urban environments with many buildings and a large number of vehicles. Furthermore, this article presents a simple yet effective heuristic to deploy edge computing devices based on the knowledge of road traffic in the target deployment area. The devised methods are evaluated by extensive simulations with data from the city of Dublin. The obtained results show that the proposed solutions can effectively guarantee a target application-specific quality of service in realistic conditions.**

***Index Terms*—edge computing, deployment, roadside units**

I. Introduction

There has been a tremendous growth in the number of smart, connected cars over the past few years. Indeed, many vehicles already available in the market are equipped with advanced sensing technologies, including ultrasound, infrared, radar and video sensors [1]. These sensors collect information from the surroundings of the vehicle and enable assisted or even autonomous driving. Moreover, connected vehicles can exchange sensor information with each other and to fixed access points, known as roadside units (RSUs), deployed along the road. Applications that leverage sensor information from multiple vehicles can be used to improve road safety [2], reduce road traffic congestion [3] or emissions [4], and enhance driving comfort [5].

Vehicular applications require complex processing of data and substantial storage space [5, 6]. Thus, sensor data generated by vehicles are usually transferred to the cloud for further processing. Indeed, the scalable and on-demand nature of the cloud is ideally suited for processing these data. However, cloud computing resources are usually centralized into large data centers, typically located far away from the majority of end-users. As a result, data exchanged between vehicles and the cloud may experience high latency. Moreover, the increasing amount of data demands high-bandwidth connections to the cloud. To address these issues, the new approach of *edge computing* has been proposed [7]. Accordingly, computing and storage resources are made available at the edge of the access network, usually one hop away from end-devices. In the context of vehicular networks, computing resources can be deployed at RSUs to realize an *edge computing device*. Thus, vehicular applications can be hosted at the edge of the network so as to process data with a low latency [5, 6, 8, 9]. For instance, safety applications at the edge can combine information from multiple vehicles and pedestrians to predict hazards. These applications can then send advisory messages (such as to change lanes or slow down) with very low latencies to relevant vehicles [10]. Another example of an edge application is a merging control algorithm for autonomous connected vehicles [11]. This application relies on vehicles communicating their mobility data every second to RSUs where new optimized trajectories are evaluated and communicated to the vehicles. Here again, the communication latency is very critical.

Deploying edge devices for vehicular applications in urban scenarios presents two major challenges. First, it is difficult to ensure that vehicles have continuous connectivity to the RSUs due to their mobility. This is especially true in urban areas, where direct line-of-sight connectivity is often impaired due to the presence of buildings. Second, edge computing devices have limited processing resources compared to the cloud. Consequently, it is critical to ensure that they meet the computational requirements of vehicular applications given a certain amount of traffic in an urban area. A large share of the existing works in vehicular networks have addressed cost-effective placement of RSUs purely from the communication perspective [12–15], i.e., without considering the computational demand of vehicular applications. Even though there are a few solutions explicitly targeting edge computing in vehicular networks [5, 8], they focus on resource allocation and scheduling rather than on provisioning an infrastructure suitable for vehicular applications. Indeed, the physical deployment of computation to meet user demands while balanc-

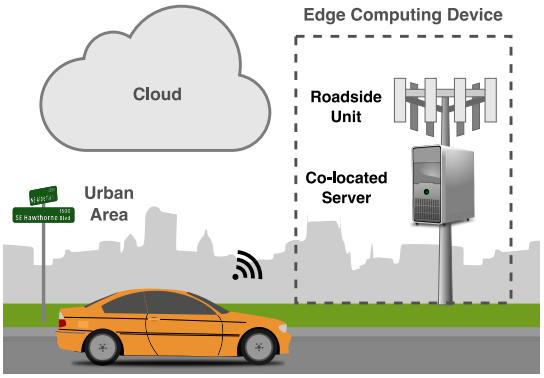


Fig. 1: Reference scenario.

Sym.	Description
$\mathcal{C}$	Set of grid cells
$\mathcal{P}$	Set of power transmit levels
i	Candidate grid cell for RSU placement
k	Power level of an RSU antenna
$A_{i,j,k}$	Adjacency matrix for cells i and j covered with power level k
m	Number of CPU cycles available at RSU (supply)
γ	Percentage of network coverage to be met
α	Percentage of computational demand to be met
d_i	Demand at grid cell i as number of CPU cycles
l_i	Length of roads present in grid cell i
$a_{i,j}$	Cost multiplier based on the distance between cell i and cell j
b_k	Cost multiplier based on the power level
c	Fixed cost for an RSU
$x_{i,k}$	Binary variable for RSU placed at cell i with power level k
$z_{i,j}$	Binary variable for cell i covered by the RSU placed in cell j
y_i	Binary variable for cell i selected to place at least one RSU
h_i	Binary variable for cell i being covered

TABLE I: Summary of notation in the optimization problem.

ing the associated costs has been identified as one of the open issues for edge computing [16].

The objective of this article is to develop an optimization problem and a methodology that can be used by infrastructure providers to deploy edge computing devices in a smart city. To this end, this article presents a mixed integer linear programming formulation that minimizes the deployment cost of edge computing devices by jointly satisfying a target level of network coverage and computational demand. Our model includes an accurate characterization of complex urban scenarios, including the effect of the built-up environment on communication between vehicles and edge devices. Additionally, this article proposes a simple yet effective heuristic to deploy edge computing devices based on the knowledge of road traffic in the target deployment area. The proposed solutions are evaluated through extensive simulations with data from the city of Dublin. The obtained results show that our solutions can effectively guarantee a target application-specific quality of service in realistic scenarios.

The rest of the article is organized as follows. Section II introduces the reference scenario and our proposed optimization problem. Section III describes the setup and the methodology used in our performance evaluation. Section IV discusses the obtained results. Section V reviews the related work. Finally, Section VI provides some concluding remarks.

II. Efficient Placement of Edge Computing Devices

Before proceeding further, let us introduce the reference scenario illustrated in Figure 1. We consider a smart city with vehicles (e.g., cars) moving in an urban area. Such vehicles run applications that collect data from the surrounding environment and report them to intermediate computing nodes for further processing and analysis. In doing so, they are assisted by a long-range wireless communication infrastructure consisting of *roadside units* (RSUs), i.e., access points specifically deployed to enable vehicular communications. Vehicles communicate directly[1] with RSUs.

Each RSU is equipped with a server; the combination of an RSU and a co-located server – simply referred to as RSUs in the rest of the article, for the sake of conciseness – constitutes an *edge computing device*. These edge computing devices are connected to the Internet, where additional storage and processing resources are available through the cloud computing paradigm. Figure 1 shows the considered reference architecture. The target scenario consists of a smart city that needs to be instrumented with an infrastructure consisting of edge computing devices to support vehicular applications. Within this context, we aim at minimizing deployment costs while jointly satisfying an application-specific quality of service in terms of network coverage and processing constraints.

This section first describes the system model for the considered scenario. It then presents a mixed integer linear programming model to minimize the deployment costs of the edge computing infrastructure, subject to application-specific constraints.

A. System Model

The target deployment area is modeled as a set $\mathcal{C}$ of grid cells, not necessarily consisting of the same size. Each cell $i \in \mathcal{C}$ has a candidate location for placing an RSU.

Each RSU can be deployed with a specific transmit power level, with the set $\mathcal{P}$ indicating the actual values allowed. An adjacency matrix $A_{i,j,k}$ describes if the RSU located in cell i covers the roads in cell j with power level k. The computation tasks are characterized in terms of CPU cycles[2], an approach that is widely used in the literature on mobile cloud computing [19–23]. All RSUs are assumed to be homogeneous in terms of processing power, i.e., their co-located servers all have the same hardware characteristics. Consequently, we define m as the available CPU cycles at an RSU and d_i as the demand for CPU cycles in cell i. Finally, application-specific quality of service is described in terms of both required coverage and processing power. Accordingly, γ represents the network coverage percentage of the deployment

[1]While direct message exchanges between individual vehicles could also be employed, this work focuses on vehicle-to-infrastructure communications.

[2]The CPU cycles required by a certain application can be estimated, for instance, by using the approaches described in [17, 18].

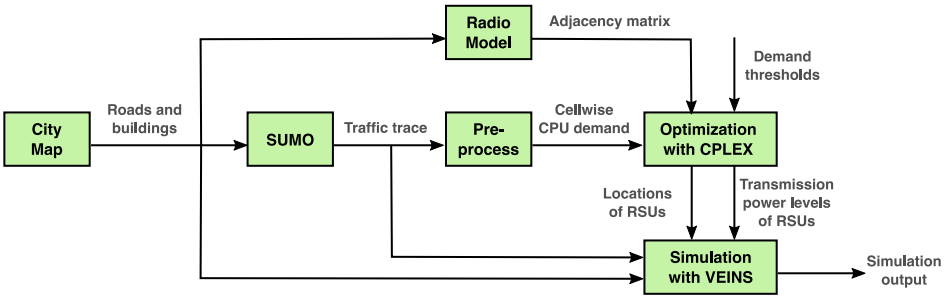


Fig. 2: Overview of the methodology used to evaluate the performance of RSU-OPT.

area, and α represents the percentage of total computational demand that is satisfied in the network.

B. Optimization Problem

Given the notation in Table I, our goal is to minimize the cost of deploying RSUs while ensuring a target network coverage and a given level of available computational resources. Accordingly, we formulate a mixed integer linear programming model that captures the distinctive features of the scenario under consideration. The problem of efficient deployment of RSUs is called RSU-OPT and is defined as:

$$\min \sum_{i=1}^{|\mathcal{C}|} c y_i + \sum_{i=1}^{|\mathcal{C}|}\sum_{j=1}^{|\mathcal{C}|} a_{i,j} z_{i,j} + \sum_{k=1}^{|\mathcal{P}|}\sum_{i=1}^{|\mathcal{C}|} b_k x_{i,k} \quad (1)$$

$$\text{s.t.} \sum_{k=1}^{|\mathcal{P}|} A_{j,i,k} x_{j,k} \geq z_{i,j} \quad \forall i,j \quad (2)$$

$$\sum_{j=1}^{|\mathcal{C}|} z_{i,j} \geq h_i \quad \forall i \quad (3)$$

$$\sum_{i=1}^{|\mathcal{C}|} h_i l_i \geq \gamma \sum_{i=1}^{|\mathcal{C}|} l_i \quad (4)$$

$$\sum_{k=1}^{|\mathcal{P}|} x_{i,k} = y_i \quad \forall i \quad (5)$$

$$\sum_{i=1}^{|\mathcal{C}|} h_i d_i \geq \alpha \sum_{i=1}^{|\mathcal{C}|} d_i \quad (6)$$

$$\sum_{i=1}^{|\mathcal{C}|} z_{i,j} d_i \leq m y_j \quad \forall j \quad (7)$$

$$y_i,\ x_{i,k},\ z_{i,j},\ h_i \in \{0,1\} \quad (8)$$

We now explain each expression in RSU-OPT, starting from the objective function in Equation (1). The first term therein denotes the fixed cost of all deployed RSUs. The second term signifies a cost $a_{i,j}$ that increases with the distance between an RSU and the cell it covers. As an RSU can cover one or more cells, the cost $a_{i,j}$ ensures that an RSU placed in cell i is preferred to cover adjacent cells j rather than more distant cells. Finally, the last term accounts for the cost of each RSU's assigned power level k (clearly, lower power levels have a lower cost b_k).

Next, we describe the constraints. On the one hand, Equations (2)–(5) refer to network coverage, while Equations (6)–(7) characterize the computational demand. Specifically, Equations (2) and (3) set h_i to 1 if cell i is covered by an RSU. Equation (4) ensures that the coverage requirement γ is met. As cells may have different sizes, the coverage is weighted by the length of roads l_i in the cell. Equation (5) sets y_i to 1 if an RSU is deployed in cell i. Equation (6) ensures that at least α percentage of the demand for CPU cycles is satisfied. Moreover, Equation (7) ensures that the available CPU cycles at an RSU are not exceeded. Finally, Equation (8) signifies that the decision variables are binary integer variables.

III. Experimental Setup and Methodology

This section first introduces some preliminary processing that is needed to provide the input data for our evaluation. Then, it describes the evaluation setup and methodology. Figure 2 overviews the methodology used to solve RSU-OPT and evaluate its performance.

A. Preliminaries

We consider the city of Dublin in Ireland as the target area for deploying RSUs. A map of Dublin city center, including information about roads and buildings, is obtained from OpenStreetMap[3]. The map encompasses an area of 3.2 by 3.1 square kilometers. We use the tools provided by Simulation of Urban Mobility (SUMO) [24], a road traffic simulator, to extract the road network as line segments and buildings as polygons. Residential and service roads are removed from the map as they are not intended for general traffic. We then use Python and its geospatial libraries to divide the map into a grid of cells with a maximum size of 200 m $\times$ 200 m. Only the cells that contain roads are considered as potential candidates for placing an RSU. There are 249 such cells in the considered map and the candidate location for an RSU is chosen in each cell based on the characteristics of roads and buildings. Moreover, some cells are divided into smaller ones to ensure that an RSU can cover all roads within a cell with one of the possible transmission power levels and despite the presence of buildings in the line of communication. Figure 3 shows the

[3]http://www.openstreetmap.org

(a) (b)

Fig. 3: (a) Map of the considered deployment area in Dublin, Ireland. (b) Grid cells for the area highlighted in (a).

Parameter	Value
RSU transmission power	21 or 24 dBm
RSU antenna height	3 m
Receiver sensitivity	-100 dBm
Propagation model	Free-space with obstacle shadowing [25]
Message size	160 bytes
Message frequency	1 Hz

TABLE II: Simulation parameters.

map of the considered deployment area and the grid cells over a subsection of the area. As shown in the figure, the cells may not all have the same size.

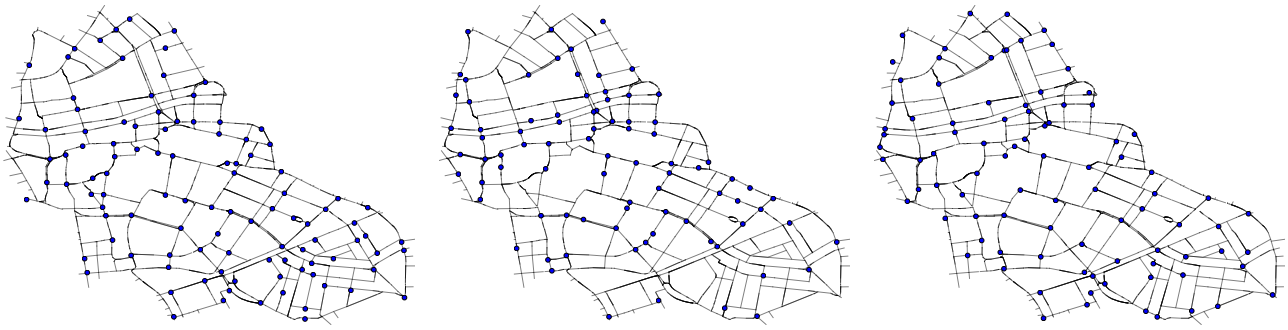
In addition to the target deployment area, RSU-OPT requires the adjacency matrix $A_{i,j,k}$ describing cell coverage. The adjacency matrix is derived by using the obstacle shadowing model proposed by [25], which accounts for communication impairments due to the presence of buildings. This model is ideally suited for the heavily built-up nature of the deployment area. Accordingly, a Python script is created to set $A_{i,j,k}$ to 1 if cell j can be covered by an RSU in cell i when transmitting with power level k. We consider a cell as covered if vehicles receive a signal above the receiver sensitivity in at least 80% of the roads in that cell. As a consequence, our approach accurately takes into account the shadowing effect of buildings using up-to-date information from OpenStreetMap. The specific values of transmission power and receiver sensitivity are described in Section III-B.

The optimization problem also requires the average demand d_i for CPU cycles in each cell. To this end, we first use SUMO tools to extract the average number of vehicles in each cell i from a traffic trace. The traffic trace is generated with the `randomTrips` tool, part of SUMO, due to the lack of relevant and publicly available datasets for Dublin. Realistic traces are generated by assigning higher weights to busier roads (with more lanes, roads in the city center) and by generating sufficiently long routes for each vehicle in the trace. The average number of vehicles in each cell is mapped to the CPU demand by specifying the number of CPU cycles required to process each message received from vehicles. Specifically, the computation requirement is set to 18,000 CPU cycles per input bit, similar to the value employed in [21].

B. Experimental Setup

We evaluate the efficiency of our proposed RSU placement scheme by simulation. In particular, we employ the Vehicles in Network Simulation (Veins) [26] framework, which integrates SUMO with the OMNeT++ network simulator. Veins enables the use of traffic traces from SUMO as input, supports the messaging protocol standards for vehicular networks, and includes the shadowing effects of buildings for realistic simulations. Furthermore, Veins simulates message losses due to collisions between overlapping transmissions and due to low signal strength.

Communication between vehicles and RSUs occurs over the IEEE 802.11p standard for vehicular communications [25]. The possible transmission power levels are set to 21 dBm and 24 dBm, which are within the limit of recommended parameters for typical RSUs [27]. In the simulations, vehicles send one message every second (similar to the application described by [11]) with a payload of 160 bytes. The RSU sends a response back to the vehicle for each incoming message. Table II summarizes the key parameters and their associated values as employed in the simulations. Furthermore, each RSU provides 600 megacycles per second to sequentially execute the tasks of a single application, according to [21, 22]. We assume that each RSU server is equipped with a 1.2 GHz processor, similar to currently available processors specifically developed for edge applications [28, 29]. Thus, the availability of computing resources for our considered application is limited to 50 % of the available CPU cycles at the RSU. Such a modeling approach captures the effect of deploying multiple applications as containers (a unit of virtualization) on



(a) (b) (c)

Fig. 4: Placement of (a) 105 RSUs with RSU-OPT (γ and α set to 100%), (b) 84 RSUs with traffic volume, and (c) 91 RSUs uniformly distributed every 500 m.

the server, each of which is limited to a certain percentage[4] of the server's CPU cycles.

C. Methodology

We solve RSU-OPT with IBM ILOG CPLEX (version 12.6.3) through its Python API. The solver provides solutions within 2 seconds (for all instances with different values for α and γ) on a laptop with an Intel Core i5-5300U CPU and 16 GB of RAM. The solution consists of the total number of deployed RSUs, their optimal locations (i.e., in terms of the corresponding cells), and their transmission power levels. Unless otherwise specified, the target percentage of network coverage and CPU demand (i.e., the QoS-related parameters γ and α, respectively) are set to 100%. This results in 105 RSUs placed in the considered area, out of which 3 operate at a higher transmission power.

For comparison purposes, we consider a baseline approach, wherein RSUs are placed at a fixed distance from each other, and a heuristic, wherein RSUs are placed depending on average traffic volume in a given area. These approaches are detailed next; Figure 4 shows the resulting placement of RSUs[5] in the target deployment area.

- **Uniform**. RSUs are placed at an approximately fixed distance apart at road intersections, which are preferred locations for deployment purposes in vehicular networks [14]. According to the characteristics of the source deployment area, 500 m and 1 km are used as reference distances in the evaluation. For a fair comparison, we keep the number and power levels of RSUs similar to that of the optimized solution. In particular, 91 RSUs are deployed (3 at higher power level) for a reference distance of 500 m and 42 RSUs (all at higher power level) for a reference distance of 1 km.
- **Traffic volume**. In this heuristic, RSUs are placed based on the volume of traffic in the target deployment area. For a fair comparison, the same grid cell overlay used in the optimization problem is considered. First the average traffic volume in each cell is obtained using aggregated statistics from SUMO. RSUs are then placed in the cells as follows. If the traffic volume in a cell exceeds a threshold of 100 vehicles/hour, an RSU is placed in the cell. The threshold is chosen in such a way that the number of RSUs from this scheme is close to that of RSU-OPT. In cells with a traffic volume lower than the threshold, RSUs are deployed further apart by only considering those cells which are not located adjacent to other cells already containing RSUs. This deployment consists of 84 RSUs, all at the lower power level of 21 dBm.

[4]https://docs.docker.com/engine/admin/resource_constraints/#cpu

[5]For illustration purposes, we report the deployments that result in a similar number of RSUs being deployed.

Parameter	Value
Maximum speed	14 m/s
Maximum acceleration	2.6 m/s^2
Maximum deceleration	4.5 m/s^2
Speed deviation	0.1
Driver imperfection	0.5

TABLE III: Vehicle mobility parameters.

We consider the following metrics (the lower the better) to evaluate the performance of the considered approaches: (i) the *message loss*, in terms of the number of sent messages that were not successfully received by the recipient, expressed as a percentage; (ii) the *exceeded CPU capacity*, in terms of the percentage of messages that all RSUs in the network were not able to process due to the unavailability of CPU resources; (iii) the *deployment cost*, in terms of the total number of RSUs placed in the target area.

We use 10 different traffic traces, each lasting for 500 s of mobility, as input for the simulations. The traces are sufficiently long so as to observe vehicular traffic on almost the entire road network (less than 10% of the roads are unused). The vehicle mobility parameters (listed in Table III) are chosen to be realistic in urban scenarios. We divide the input in two sets of five traces each: one corresponding to moderate traffic, with an average of 620 vehicles; another corresponding to heavy traffic, with an average of 1,003 vehicles. Specifically,

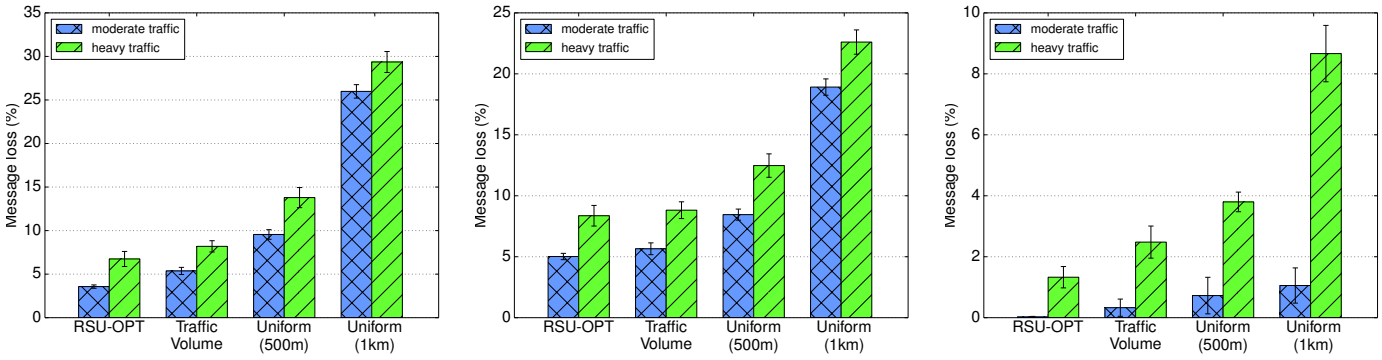


Fig. 5: Performance of the different schemes considered: message loss in (a) the uplink only (i.e., from vehicles to RSUs) and (b) both directions (i.e., uplink and downlink); (c) exceeded CPU capacity.

moderate traffic traces are obtained by drawing vehicle arrivals from a binomial distribution[6] with 2 simultaneous arrivals and a period of 0.3. The heavy traffic traces are obtained by drawing arrivals from a binomial distribution with 5 simultaneous arrivals and a period of 0.5. We carry out simulations according to the independent replication method. The results report the average values obtained from five runs as well as the corresponding standard deviations, shown as whiskers in the plots. The simulations include a warm-up period of 50 s to allow sufficient number of vehicles to enter the road network.

IV. Experimental Results

This section first presents a comparison of RSU-OPT with the other approaches discussed earlier. Then, it presents the impact of target QoS parameters on performance. Finally, the section concludes with a summary and discussion.

A. Impact of Traffic

We start by studying the performance of RSU-OPT in comparison with the other approaches for placement under two different traffic conditions: moderate and heavy (as described in Section III-C). The related results are shown in Figure 5.

In particular, Figure 5a shows the message loss in the uplink, i.e., the one affecting messages sent by vehicles to RSUs. We can clearly see that RSU-OPT performs the best under both traffic conditions. The traffic volume approach achieves a slightly worse delivery ratio, while the uniform placement of RSUs every 500 m incurs in a high percentage of message loss, i.e., 10% with moderate traffic and close to 14% with heavier traffic. Moreover, the uniform placement of RSUs every kilometer results in one message dropped out of every four sent by vehicles, even with moderate traffic.

For all the considered schemes, the message loss increases under heavy traffic as more messages are exchanged in the network and the probability of interference from neighboring vehicles increases as well. Under these conditions, RSU-OPT still achieves a satisfactory delivery ratio and outperforms the other schemes. It is important to highlight that the uniform deployment of RSUs every 500 m results in a higher message loss than the traffic volume scheme despite placing more RSUs. This demonstrates the need for smarter heuristics for placing RSUs, e.g., by taking into account the traffic volume in the considered area. On the other hand, the traffic volume scheme performs similar to RSU-OPT, despite the lower number of RSUs deployed. This is because RSUs are more densely placed in "busy" areas with the traffic volume scheme. Note that the same message can still reach the network even though it may be lost at a specific RSU (e.g., due to interference from nearby vehicles and simultaneous transmissions by multiple RSUs), as it might be successfully received by an RSU in a cell nearby.

Next, Figure 5b presents the message loss in both uplink and downlink. The trend is similar to that in Figure 5a. In particular, RSU-OPT achieves a slightly lower percentage of message loss than the traffic volume scheme. With an average of 400,000 packets and 250,000 messages generated in the high and moderate traffic scenarios, respectively, even a small difference in percentage of message loss is indeed non-negligible.

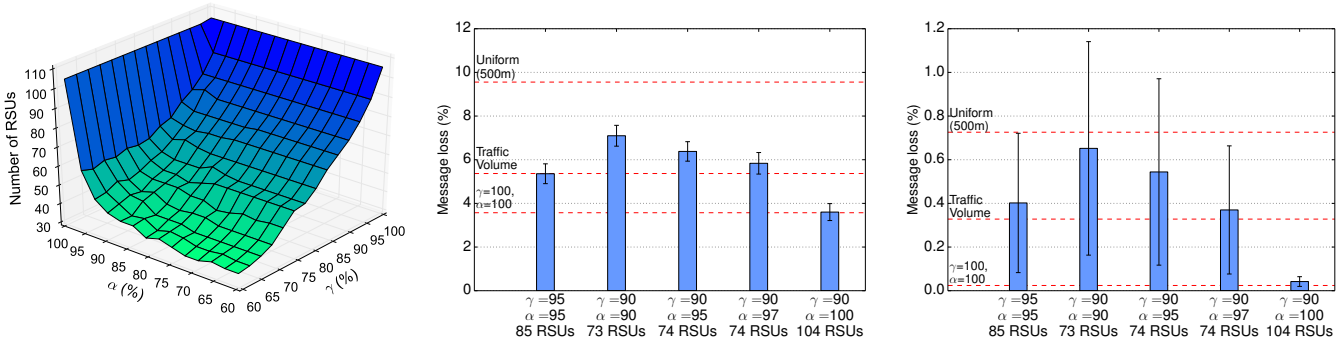
Finally, Figure 5c shows the exceeded CPU capacity, i.e., the percentage of incoming messages dropped at RSUs due to the unavailability of computing resources. The performance of RSU-OPT is the best among the considered approaches, also with the least spread across multiple iterations. Note that the traffic volume approach performs worse than RSU-OPT, even though it places more RSUs in areas with high traffic volume. This is because RSU-OPT jointly considers the requirements of coverage and processing in the entire deployment area.

B. Impact of Target Level of QoS

We now examine the impact of different requirements of network coverage (γ) and computational demand (α) in the optimization problem.

Figure 6a shows the number of RSUs required under these different conditions. In particular, RSU-OPT places 105 RSUs to achieve 100% network coverage and meet 100% computa-

[6] http://sumo.dlr.de/wiki/Tools/Trip



(a) (b) (c)

Fig. 6: Impact of the target network coverage (γ) and computational demand (α) on performance: (a) number of RSUs placed by RSU-OPT, (b) message loss in the uplink (from vehicles to RSUs), and (c) exceeded CPU capacity.

tional demand. The number of RSUs rapidly decreases as γ and α are reduced. For instance, only 85 RSUs are required to provide network coverage for 95% of the roads and to meet 95% of the computational demand. This indicates the presence of certain areas in the city (i.e., grid cells in RSU-OPT) that cannot be covered by neighboring RSUs, for instance, due to the presence of buildings and lack of line-of-sight connectivity. Reducing γ further to 90% results in only 73 or 74 RSUs depending on the value of α. Furthermore, decreasing either α or γ while keeping the other at 100% does not significantly affect the number of RSUs. For instance, setting α to 100% results in 104 RSUs independent from the value of γ. It is important to note the actual placement (in terms of grid cells) of RSUs varies even though their number may be the same for different values of γ and α.

Next, we evaluate how the network performs with RSU-OPT under selected values of target network coverage γ and computational demand α. As network coverage is the most stringent constraint, γ is set to either 90% or 95% while α is varied between 95% and 100%. In this scenario, the trends remain the same under both moderate and heavy traffic and thus, only the results for moderate traffic are presented here.

Figures 6b and 6c show the message loss with the different placements obtained with RSU-OPT. The figures also indicate (as horizontal lines) the message loss for the different approaches: RSU-OPT (with α and γ set to 100%), traffic volume and uniform placement every 500 m. The uniform placement every kilometer is no longer considered as the observed message loss percentage is very high.

Reducing both α and γ to 95% results in 85 RSUs being deployed. In this case, the network performance is similar to the traffic volume scheme, which places 84 RSUs. Reducing γ to 90% and varying α between 90% to 100% produces deployments with only 73 or 74 RSUs. The corresponding performance of RSU-OPT is significantly better than the uniform deployment every 500 m, which requires 91 RSUs. In Figure 6c we observe that increasing α from 90% to 100% reduces the exceeded CPU capacity. Although the number of RSUs are the same for α set to 95% or 97%, their actual locations vary, thus decreasing the exceeded CPU capacity.

Finally, the exceeded CPU capacity decreases for messages sent from vehicles to RSUs when α increases from 90% to 100%. Indeed, RSU-OPT balances the requirements of both network coverage and computational demand to find optimal locations of RSUs. The reduction in the exceeded CPU capacity from vehicles to RSUs occurs because placing RSUs in areas with higher computational demand (implying higher traffic volume) results in better network coverage too. With α set to 100%, the performance improves in terms of both overall percentage and reduced spread in the results.

C. Summary and Discussion

Our experiments show that the optimization problem RSU-OPT outperforms other approaches, with the traffic volume heuristic coming a close second. Although the traffic volume scheme performs very well, the number of deployed RSUs depends on selecting a specific threshold. Moreover, such a heuristic cannot obtain the number of RSUs required to cover a given percentage of roads or meet a certain computational demand. Besides, the performance of the heuristic is similar to that of RSU-OPT with α and γ set to 95%.

We observe that certain RSUs drop more messages under heavy traffic, due to the unavailability of CPU resources. There are two ways to address this issue. First, the input to the optimization problem can be changed from the currently used average demand. For instance, a different metric could be the maximum demand for CPU cycles in each cell. However, the solution to this problem would over-provision the network by increasing the computation resources at RSUs. This is not really necessary, especially when the current placement and capacity at each RSU is able to meet the requirements under moderate traffic. Besides, the chosen configuration is in the range of the processors available (up to 1.6 GHz) for intelligent edge devices. Also, edge computing platforms are expected to

have limited computing power as they are typically constrained by radio resources [22]. Thus, we consider a second approach, where the number of RSUs is increased in the areas with heavy traffic. This can be achieved by using feedback from the Veins simulations to identify the specific cells where an RSU is not able to meet the computational resource demand and an extra RSU is deployed in those cells. This approach was tested by deploying an extra RSU to the placement obtained by RSU-OPT, which requires 105 RSUs. Specifically, an extra RSU was placed in one of the grid cells with high demand for computation. We ran the simulations with 106 RSUs under heavy traffic and observe that the percentage of messages lost reduced from 1.33% to 0.76%.

V. Related Work

Optimal placement of RSUs for network coverage has been studied extensively in both urban and highway scenarios. We only discuss the optimization problems targeting RSUs deployments in urban areas as they are closer to our work. Among them, Liang et al. [12] formulate a linear programming model to minimize the cost of deploying RSUs. The model allows multi-hop communication and reduces delay by setting a maximum value for the number of hops. In contrast, we consider only single-hop communication to enable the extremely low latency required for edge applications. Under this scenario, vehicles always have to be under the coverage of an RSU. Aslam et al. [13] describe the optimal placement of a specified number of RSUs in an urban area, while minimizing the time taken for a vehicle to enter the coverage range of an RSU and report an accident. Unfortunately, the average reporting time with this approach is beyond the values required by latency-critical applications. Trullols et al. [14] consider the placement of RSUs for information dissemination to vehicles. The problem is modeled as a maximum coverage problem, which is NP-hard. In particular, the objective of the problem is to maximize the number of vehicles that are in contact with the RSUs as well as the time spent by vehicles under RSU coverage. On the other hand, our model attempts to provide coverage at all times to a target percentage of the deployment area. Balouchzahi et al. [15] propose a model based on integer programming for efficient placement of RSUs in urban and highway scenarios. They also consider the placement of servers for vehicular applications, even though the computational requirements of the tasks are not included in their model. To the best of our knowledge, none of the works on RSU placement incorporate computational requirements. In contrast, our model explicitly considers the computational requirements of vehicular applications and limited computational capability of edge devices.

Several articles propose vehicular applications and architectures that rely on edge servers co-located with RSUs [5, 8, 9, 30, 31]. Yu et al. [5] describe the allocation of cloud resources at edge servers deployed along with RSUs. The resources available at RSUs are limited and the authors propose a game-theoretic approach to allocate the required resources. Salahuddin et al. [8] introduce an integer linear programming model to manage computation resources at RSUs in such a way to minimize re-configuration costs as well as delay. In these two solutions, virtualized computational resources follow the location of vehicles and, thus, need to be moved from an RSU to another accordingly. In our solution, all RSUs run the same application(s), consequently, there is no need to move computational resources in the network (although intermediate data can be transferred, if required). Zhang et al. [9] propose a computational offloading framework for vehicular networks that relies on servers with limited computational resources co-located with RSUs. The authors study the optimal selection of edge servers and transmission of data between the edge devices. However, the considered articles do not describe the efficient deployment of the edge devices but the efficient movement of resources between them. Our work is complementary to the described articles as it focuses on the physical deployment of edge computing devices to enable the proposed offloading architectures. Our proposed deployment model also enables the applications described by [11, 30, 31], which rely on intelligent edge devices.

VI. Conclusion

This article addressed the efficient placement of edge computing devices for vehicular applications in an urban scenario. To this end, we have introduced an optimization problem that is able to capture application-specific quality of service parameters. The objective of the problem is to provide network coverage in a vehicular network while also meeting the computational demand in the considered area. Our solution was evaluated through extensive simulations with the road topology of Dublin and compared to other placement schemes. The obtained results showed that the proposed solution is effective, outperforms other placement schemes and satisfies the needs for edge computing in realistic conditions. As a future work, we plan on investigating the processing delay at the edge computing servers by means of probabilistic models. We can then examine the placement of RSUs based on different thresholds of response delays. While we have considered a static deployment of RSUs and edge computing resources, our model can also be extended to include dynamic adjustment of RSU parameters so as to meet the requirements of changing traffic patterns. Finally, we are interested in applying our optimization framework to scenarios involving the use of wireless communication technologies other than IEEE 802.11p.

Acknowledgments

Part of this work was carried out when Gopika Premsankar and Bissan Ghaddar were at IBM Research – Ireland. This work has been partially supported by the Academy of Finland under grant number 299222. We would like to thank Rodrigo Ordonez-Hurtado and Wynita Griggs for their help with the data describing the city of Dublin and its road network. Finally, we would like to thank the CSC – IT Center for Science and the Aalto Science-IT project for provisioning the computational resources used in the evaluation.

REFERENCES

[1] E. Uhlemann, "Introducing connected vehicles," *IEEE Vehicular Technology Magazine*, vol. 10, no. 1, pp. 23–31, 2015.

[2] T. Taleb, A. Benslimane, and K. B. Letaief, "Toward an effective risk-conscious and collaborative vehicular collision avoidance system," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 3, pp. 1474–1486, 2010.

[3] K. Katsaros, R. Kernchen, M. Dianati, and D. Rieck, "Performance study of a Green Light Optimized Speed Advisory (GLOSA) application using an integrated cooperative ITS simulation platform," in *2011 7th International Wireless Communications and Mobile Computing Conference*. IEEE, 2011, pp. 918–923.

[4] S. Widodo, T. Hasegawa, and S. Tsugawa, "Vehicle fuel consumption and emission estimation in environment-adaptive driving with or without inter-vehicle communications," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2000, pp. 382–386.

[5] R. Yu, Y. Zhang, S. Gjessing, W. Xia, and K. Yang, "Toward cloud-based vehicular networks with efficient resource management," *IEEE Network*, vol. 27, no. 5, pp. 48–55, 2013.

[6] S. K. Datta, R. P. F. Da Costa, J. Härri, and C. Bonnet, "Integrating connected vehicles in internet of things ecosystems: Challenges and solutions," in *WoWMoM 2016*. IEEE, 2016, pp. 1–6.

[7] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct.-Dec. 2009.

[8] M. A. Salahuddin, A. Al-Fuqaha, M. Guizani, and S. Cherkaoui, "RSU cloud and its resource management in support of enhanced vehicular applications," in *Globecom Workshops (GC Wkshps), 2014*. IEEE, 2014, pp. 127–132.

[9] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Predictive offloading in cloud-driven vehicles: Using mobile-edge computing for a promising network paradigm," *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 36–44, 2017.

[10] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen, "Data-driven intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1624–1639, 2011.

[11] C. Letter and L. Elefteriadou, "Efficient control of fully automated connected vehicles at freeway merge segments," *Transportation Research Part C: Emerging Technologies*, vol. 80, pp. 190–205, 2017.

[12] Y. Liang, H. Liu, and D. Rajan, "Optimal placement and configuration of roadside units in vehicular networks," *IEEE Vehicular Technology Conference*, 2012.

[13] B. Aslam, F. Amjad, and C. C. Zou, "Optimal roadside units placement in urban areas for vehicular networks," in *2012 ISCC*, July 2012, pp. 423–429.

[14] O. Trullols, M. Fiore, C. Casetti, C. F. Chiasserini, and J. M. Barcelo Ordinas, "Planning roadside infrastructure for information dissemination in intelligent transportation systems," *Computer Communications*, vol. 33, no. 4, pp. 432–442, 2010.

[15] N. M. Balouchzahi, M. Fathy, and A. Akbari, "Optimal road side units placement model based on binary integer programming for efficient traffic information advertisement and discovery in vehicular environment," *IET Intelligent Transport Systems*, vol. 9, no. 9, pp. 851–861, 2015.

[16] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, 2017.

[17] A. Iranpour, "Exploring processor and memory architectures for multimedia," Ph.D. dissertation, Lund University, 2012.

[18] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing." *HotCloud*, vol. 10, pp. 4–4, 2010.

[19] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Joint allocation of computation and communication resources in multiuser mobile cloud computing," in *SPAWC 2013*. IEEE, 2013, pp. 26–30.

[20] C. You and K. Huang, "Multiuser resource allocation for mobile-edge computation offloading," in *2016 GLOBECOM*, Dec 2016, pp. 1–6.

[21] Y. Yu, J. Zhang, and K. B. Letaief, "Joint subcarrier and cpu time allocation for mobile edge computing," in *2016 GLOBECOM*, Dec 2016, pp. 1–6.

[22] Y. Mao, J. Zhang, and K. B. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems," in *WCNC 2017 IEEE*, 2017, to appear.

[23] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, Oct 2016.

[24] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, December 2012.

[25] C. Sommer, D. Eckhoff, R. German, and F. Dressler, "A computationally inexpensive empirical model of IEEE 802.11 p radio shadowing in urban environments," in *WONS 2011*. IEEE, 2011, pp. 84–90.

[26] C. Sommer, R. German, and F. Dressler, "Bidirectionally coupled network and road simulation for improved IVC analysis," *IEEE Transactions on Mobile Computing*, vol. 10, no. 1, pp. 3–15, 2011.

[27] ETSI, "TR 102 654, V1.1.1 Electromagnetic compatibility and Radio spectrum Matters (ERM)," 2009.

[28] NXP, "ARM processors," http://www.nxp.com/products/microcontrollers-and-processors/arm-processors/qoriq-layerscape-arm-processors:QORIQ-ARM.

[29] Cohda Wireless, "RSU Hardware," http://cohdawireless.com/Products/Hardware.aspx.

[30] S. M. Khan, K. C. Dey, and M. Chowdhury, "Real-time traffic state estimation with connected vehicles,"

IEEE Transactions on Intelligent Transportation Systems, 2017.

[31] S. Basudan, X. Lin, and K. Sankaranarayanan, "A privacy-preserving vehicular crowdsensing based road surface condition monitoring system using fog computing," *IEEE Internet of Things Journal*, 2017.