2022-05-20

Scaling and Placing Distributed Services on Vehicle Clusters in Urban Environments

Kanika Sharma
South East Technological University

Bernard Butler
South East Technological University, bernard.butler@setu.ie

Brendan Jennings
Technological University Dublin, brendan.jennings@tudublin.ie

Recommended Citation

K. Sharma, B. Butler and B. Jennings, "Scaling and Placing Distributed Services on Vehicle Clusters in Urban Environments," in IEEE Transactions on Services Computing, doi: 10.1109/TSC.2022.3173917.

Funder: Science Foundation Ireland

Scaling and Placing Distributed Services on Vehicle Clusters in Urban Environments

Kanika Sharma *(Member, IEEE)*, Bernard Butler *(Member, IEEE)*, and Brendan Jennings *(Member, IEEE)*

Abstract—Many vehicles spend a significant amount of time in urban traffic congestion. Due to the evolution of autonomous vehicles, driver assistance systems, and in-vehicle entertainment, these vehicles have plentiful computational and communication capacity. How can we deploy data collection and processing tasks on these (slowly) moving vehicles to productively use any spare resources? To answer this question, we study the efficient placement of distributed services on a moving vehicle cluster. We present a macroscopic flow model for an intersection in Dublin, Ireland, using real vehicle density data. We show that such aggregate flows are highly predictable (even though the paths of individual vehicles are not known in advance), making it viable to deploy services harnessing vehicles' sensing capabilities. After studying the feasibility of using these vehicle clusters as infrastructure, we introduce a detailed mathematical specification for a task-based, distributed service placement model. The distributed service scales according to the resource requirements and is robust to the changes caused by the mobility of the cluster. We formulate this as a constrained optimization problem, with the objective of minimizing overall processing and communication costs. Our results show that jointly scaling tasks and finding a mobility-aware, optimal placement results in reduced processing and communication costs compared to the two schemes in the literature. We compare our approach to an autonomous vehicular edge computing-based naive solution and a clustering-based solution.

Index Terms—Vehicular Cloud Computing, Vehicular Fog Computing, Flexible service models, Service Placement, Resource Allocation.

1 Introduction

Vehicles will be one of the most important agents in the emerging Internet of Things (IoT) ecosystem, owing to their embedded sensors and built-in cameras. These moving vehicles can be used to capture contextual data for object detection and surveillance [1]. Since each vehicle generates an average of 30 Tb of data per day, it is infeasible to send all the generated data to the Cloud using the controlled and limited cellular bandwidth [2]. The increasing number of *Smart* vehicles and overall vehicular traffic has inspired the concept of Vehicular Fog Computing (VFC) [3], [4], where vehicles are utilised as Fog nodes and play the role of service providers. This new data generation and communication paradigms is motivated by Fog Computing [1] and Mobile Edge computing based models. These newer computing paradigms [5], [6] provide ubiquitous connectivity and location-aware network responses at the edge of the network, complemented with cloud computing in the network core.

The closely-spaced, moving vehicles, are proposed to collect video that can be used to estimate usage patterns of highways for urban planning, reducing the need for installing dedicated infrastructure for surveillance. Slowly moving vehicles can also be used to collect 3D roadmap data, to increase the perception range of intelligent vehicles, reducing the need for sending high definition data to the Cloud [7], [8]. The VFC-based data collection and processing applications can be swiftly deployed to monitor the compliance of both vehicles and pedestrians to lockdown restrictions introduced in response to the COVID-19 pandemic, by capturing data from essential service vehicles. All these use cases require rich computation and communication resources so that this data can be used for insights and decision-making. The otherwise under-utilized sensing and processing resources in VFC systems can meet both the data generation and processing requirements without the need for deploying additional infrastructure. Most of the existing work on VFC either consider buses [9] and taxis [10] as potential fog nodes which are not representative of the mobility patterns of all vehicles in an urban city center or consider very simplistic mobility models [11]. Many of the service placement/allocation schemes in VFC consider static services that are not adapted according to the dynamics of the network or consider stationary/parked vehicles as Fog nodes.

In this paper, we first study the predictability of vehicular flows using real vehicular density data and compare our model to competing schemes. We also study the overall feasibility of using vehicles as infrastructure by computing their aggregate communication and computing capacity. We propose that vehicles lease their otherwise unused processing, communication, and storage resources to collaboratively host data analytics services that can preprocess and filter the data they collect. Thus, a dense group (or cluster) of moving vehicles can cooperatively execute distributed services that comprise: (i) delay-sensitive tasks that have a short sense-actuate cycle and require real-time decision making, and/or (ii) data collection and analytics tasks that are location- and context-specific. These services can be flexibly deployed on the vehicle cluster based on the mobility pattern of the vehicles.

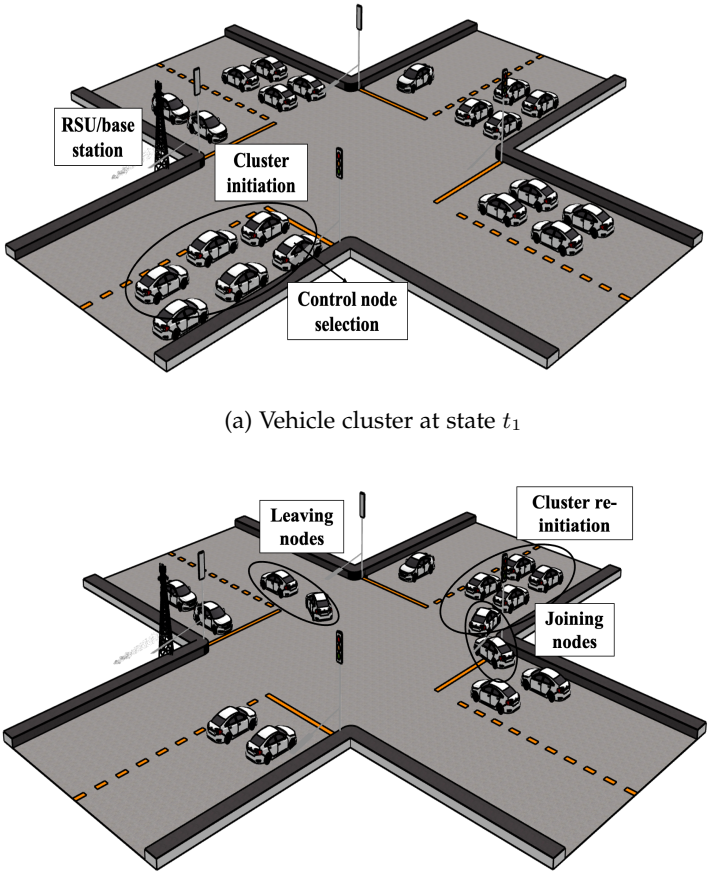
• *K. Sharma and B. Butler are with the Walton Institute, Waterford Institute of Technology, Ireland. B. Jennings is with TU Dublin. E-mail: kanika.sharma@waltoninstitute.ie,{bbutler,bjennings}@ieee.org*

Vehicles are distinguished through their *mobility* (in particular, vehicles join and leave a cluster in a stochastic manner), so the resource allocation task becomes time-dependent. Mobility affects both network connectivity and computation capacity, and hence the Quality-of-Service (QoS). Our work aims to utilize the aggregate mobility behavior of vehicles to select reliable vehicle nodes, i.e., those that have a higher probability of staying with a given cluster of vehicles, in order to avoid service failure and reduce the need for service reconfiguration. As depicted in Fig. 1, one vehicle, or Roadside Unit (RSU), acts as a managing entity to collect and update both the resource and mobility states of the cluster and enable flexible service scaling.

We take the specific case of using in-built cameras in vehicles that are willing to lease their resources, to provide streaming data on request. This data is processed by streaming it to linear chains of tasks, where each task has different processing functionality. One linear chain of tasks form a service that satisfies a service request. We employ a component-oriented distributed service model, where each task can be realised via a collection of multiple task instances (TIs). In this manner each task can be scaled out according to the demand and the available infrastructure resources. For example, using multiple camera instances increases the spatio-temporal coverage of the data collection, thereby increasing the scope for more accurate and efficient data analysis, especially in applications like building 3D road maps for self-driving vehicles. Moreover, having replicas of computing tasks enables the utilization of distributed resources and reduces the impact of nodes leaving the vehicle cluster. Node and link failure in this service model requires only the replication of a problematic TI onto a more suitable vehicle so the service chain still works, instead of re-configuring the entire service. As the tasks are data-dependent, even if the cluster is computationally rich, it needs to be split into smaller TIs if the link capacity between host nodes is not enough. The task placement also needs to avoid placing TIs on those resource-rich nodes that have a high probability of leaving the cluster. Thus, service deployments can be adapted at run-time according to both the known resource availability and the predicted mobility state of the cluster.

This paper builds upon our previous work [12], where we introduced the problem of the placement of distributed data collection services on a moving vehicle cluster. In this paper the following novel contributions are made:

- Instead of using a simple mobility model, the paper uses real vehicle density data to make a flow model for intersections. We use Multivariate Linear Regression to predict the traffic flows at the intersections and achieve an R^2 score of 0.93-0.99. The performance of the flow model is compared to competing models like random forest regression and ARIMA model for time-series forecasting.
- The feasibility of using clusters is analyzed by studying the aggregate communication and computation capacity of these moving vehicle clusters.
- We then formulate the service placement problem mathematically in two parts: 1) a flexible and distributed service model with data-dependent tasks instead of static service templates; and 2) a mobility-aware infrastructure model. This is an important contribution towards utilising the distributed and dynamic vehicular network for service provisioning.
- We compare our approach to the autonomous vehicular edge computing based-naïve solution, presented in [13] and a clustering-based solution introduced in [14]. The experimental results demonstrate that the proposed scheme outperforms baseline approaches in terms of efficient resource utilization.



(a) Vehicle cluster at state t_1

(b) Vehicle cluster at state t_2

Fig. 1: Vehicle Clusters form, but membership changes over time. Clusters accept service placement requests from RSUs and perform scaling and placement of the accepted service. Fig (a) and (b) depict the state of the cluster over time t_1 and t_2. The mobility of the vehicles requires cluster re-initiation.

The paper is organized as follows: §2 describes related research in the field of service placement and task offloading schemes in vehicular networks. In §3, we describe the motivation behind using vehicles as infrastructure, estimate vehicular cluster capacity, and validate vehicle traffic predictability with Multivariate Linear Regression. In §4, we define the system model and the network topology and in §5, we define the constraints and the mathematical formulation of the placement problem. In §6, we introduce our service scaling and placement plan. In §7, we describe application types and run an experiment for an application

scenario on a Fog simulator. We then discuss solving the optimization problem, simulation setup and our results. We also discuss the performance of our model in comparison to other schemes. Finally, in §7 we conclude the work and give an outline of our future work.

2 Related Work

Gerla et al. [15] was the first to introduce the term Vehicular Cloud Computing (VCC) as a distributed computing platform, wherein vehicles form a micro cloud in an ad hoc manner. They identified many important applications for VCC, including urban sensing by uploading videos of congestion and pavement conditions that other vehicles could access. Hou et al. [4] were the first to introduce the concept of Vehicular Fog Computing (VFC) as an architecture that can be used to enable multiple end-user or edge devices to collaborate to carry computation and communication tasks. They considered both slow-moving and parked vehicles and analyzed the quantitative capacity of such a Vehicular Fog. Ma et al. [16] introduced a Platoon-assisted Vehicular Edge Computing system based on the stability of the platoon in vehicular networks. They were the first to introduce a Reinforcement Learning (RL)-based optimization scheme to obtain optimal price strategy of task flows. Lee et al. [3] also modified an RL-based algorithm to make efficient resource allocation decisions leveraging vehicles' movement and parking status to minimise service latency.

Zhao et al. [17] jointly optimize the computation offloading decision and computation resource allocation in vehicular networks. They designed a collaborative optimization scheme where offloading decisions are made through a game-theoretic approach and resource allocation is achieved using the Lagrange multiplier method. The feasibility of using vehicles as Fog nodes for video crowd-sourcing and real-time analytics has been studied by Zhu et al. [18]. They evaluated the availability of client nodes that generate data in proportion to the vehicle Fog nodes that process the data, using processing capacity on on-board units. However, they focus solely on the data transmission problem in the model. Xiao et al. [19] also evaluated the achievable performance of a vehicular cloud and analyze the total computation capacity for the same. They also model vehicle mobility patterns using parameters like staying time and the incoming and outgoing flow rate of vehicles. This capacity analysis is a crucial requirement for enabling a VFC model but they do not focus on an adaptable service model and applications to be deployed. In Kong et al. [20], the traditional mobility models for vehicles are replaced by methods based on social patterns, community interest group check-ins on social media data etc.

The mobility of vehicle nodes makes the task allocation problem more challenging in VFC. Zhu et al. [21] introduced an event-driven dynamic task allocation framework designed to reduce average service latency and overall quality loss. Both multi-source data acquisition and distributed computing in Fog-computing-based intelligent vehicular network are studied by Zhang et al. [22]. They introduce a hierarchical, QoS-aware resource management architecture, but consider the Fog servers as static. Vehicular micro cloud has been studied as virtual edge servers for efficient connection between vehicles and back-end infrastructure in Hagenauer et al. [23]. They use map-based clustering at intersections, as intersections have line of sight in multiple directions which result in better connectivity between the Cluster Heads (CHs) and other cluster members. Even though they primarily focus on cluster creation and cluster head selection, they evaluate a data collection application, with varying data aggregation rates at the CH.

Goudarzi et al. [24] introduced an application placement technique for concurrent IoT applications in Edge and Fog computing environments. They propose a a batch application placement technique based on the Memetic Algorithm to efficiently place tasks of different workflows on appropriate IoT devices, fog servers, or cloud servers.Vehicles have also been studied for efficient content distribution to meet the challenges of limited communication resources. Luo et al. [25], proposed a content distribution scheme to maximize the system utility for content distribution.

A lot of work has been done in the literature for leveraging vehicles as infrastructure, however many of the existing works on VFC either consider a very simplistic mobility model or consider vehicles to be stationary. We also introduce the placement of distributed and flexible services that are adaptable according to the dynamics of the vehicular network. We then jointly optimize both link and processing costs for the efficient placement of the distributed services on the vehicle cluster. In addition, we study the feasibility of using opportunistic vehicle clusters as infrastructure.

3 Motivation

Our work is motivated by the increasing number of smart vehicles with embedded sensors that can connect to other vehicles, and the unresolved issue of vehicle congestion—especially in urban areas. Before introducing our service scaling and placement scheme we first provide justification that placing services on a vehicle cluster in order to provide time and/or location sensitive sensing functionality is a viable proposition. There are two important aspects: 1) whether traffic flows in an urban setting are likely to be predictable over the course of a day, and 2) whether a slow moving cluster will accommodate sufficient communications capacity between vehicles to facilitate service operation.

3.1 Predictability of vehicle flows

We find that vehicular flow in urban traffic zones is predictable throughout the day. We also show that the vehicular density pattern at an intersection follows a similar pattern of peak and off-peak flow through different weeks. We use macroscopic vehicle density data to create a generalised flow model for an intersection. This helps in classifying traffic flow into six different driving profiles. The vehicle clusters can then be initiated at the predicted peak traffic times, on any of the traffic flows with an assured density flow.

We first focus on a road network near Dublin Airport, using the vehicle flow data captured by the Transport Infrastructure Ireland Traffic Data website [1]. A vehicular flow

1. https://trafficdata.tii.ie/publicmultinodemap.asp, available: 6/02/22.

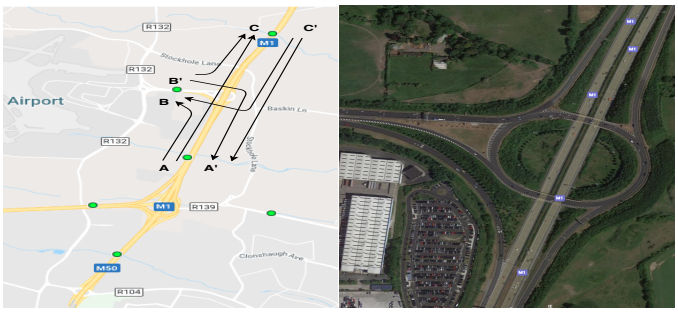


Fig. 2: Flow model at the selected intersection in Dublin with six different traffic flows from A to B, A to C, C′ to A′, C′ to B, B′ to C and B′ to A

is defined as the number of detected vehicles passing a point in a period of time. The idea is to use the stochastic traffic flows at an intersection to predict the trajectory of a vehicle cluster. As depicted in (Fig. 2), we consider northbound flow from A to B and A to C, southbound flow from C′ to A′ and C′ to B, eastbound traffic from B′ to C and B′ to A. We then employ a multivariate linear regression (MVLR) model to predict the traffic flow from one segment to the other, for all the six flows at the intersection. To understand the predictability of the traffic flows, we use the vehicle flow data, collected in the interval of 5, 10 and 15 minutes (based on the estimated travel time between any of the six points at peak and off-peak traffic time of the day) for a period of 24 hours. This data is used to model a generalized traffic flow model for an intersection. We predict the vehicle density at point B taking into consideration the vehicle density at point A, by first using a simple linear regression model, then consider traffic flow from consecutive days to use a MVLR model. To compare the performance of the MVLR model with competing schemes we use random forest-based regression model and an ARIMA model for traffic forecasting. We plot the actual and predicted incoming vehicle density at point B, for an interval of 5 minutes (Fig. 3a) and 10 minutes (Fig. 3b). The R^2 score for the Linear Regression model is 0.915 for a period of 5 minutes and 0.945 for 10 minutes respectively. This way, vehicles can be clustered in six different driving profiles for service execution, corresponding to the above-mentioned six flows. Table 1 depicts the r-value, p-value and the standard error for all the six flows.

We then use the vehicle flow data for the last 7 consecutive Mondays to predict a single flow, from A to B, using MVLR for data collected at an interval of 5 (Fig. 3c), 10 (Fig. 3d) and 15 (Fig. 3e) minutes. The same days in the week were studied to have similar patterns of mobility, within a range of a month to two, hence data for 7 consecutive Mondays was used. The predicted and actual vehicle flow at point B is depicted in Fig. 3c, 3d and 3e. The R^2 score of the prediction was 0.937, 0.948 and 0.992 for 5, 10 and 15 minutes respectively. We also considered the vehicle flow data during the period of COVID-19 lock-down, from 1st to 8th April 2020, to analyze the pattern of flow during the Coronavirus restrictions in Ireland. The restrictions resulted in much less traffic density at the intersection. Fig. 3g and Fig. 3f depict predicted vehicle flow using MVLR, considering seven consecutive days during the lock-down, with an

TABLE 1: r-value, p-value and standard error for predictability of the six flows at the intersection

	Slope	*Intercept*	*r value*	*p value*	*Standard error*
A ->B	0.28	75.14	0.81	4.45	0.02
B′ ->C	2.30	-86.36	0.85	4.00	0.12
C′ ->A′	1.03	100.14	0.97	2.45	0.02
B′ ->A′	0.01	64.38	0.75	2.83	0.15
C′ ->B	0.38	85.02	0.87	1.38	0.02
A ->C	1.49	-27.35	0.96	1.01	0.04

TABLE 2: Traffic prediction using three different comparative models using real vehicle density data at the intersection in Dublin

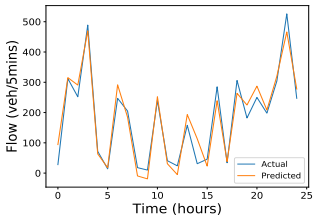
Technique used	**RMSE**	**R-squared**	**MAE**
Multivariate Linear Regression	11.70	0.97	8.1
Random Forrest	3.80	0.99	2.96
ARIMA time series forecasting	18.57	0.69	18.07

R^2 score of 0.98 and 0.987.

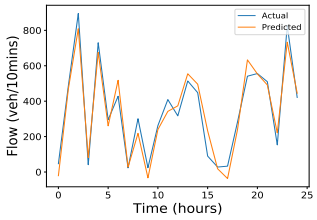
We also compare the prediction of MVLR with random forest regression, as depicted in Fig. 3g and Fig. 3f, which results in comparable prediction with an R^2 score of 0.979 and 0.997. We also compared the MVLR and random forest regression model to an ARIMA model for time-series forecasting. All the models are evaluated using the root mean squared error (RMSE), R-squared error, and mean absolute error (MAE) that are summarised for each model in Table 2. The random forest performs marginally better than MVLR. MVLR is a simple, linear model that predicts vehicular flows accurately whereas random forest is an ensemble learning model, which is more complex but generally a more accurate model. We use MVLR for traffic prediction, however, both models can be used interchangeably for traffic prediction. The ARIMA model, which is a standard model for time-series forecasting performs the worst out of the three schemes and has an R^2 score of 0.695. The logic of using the comparative schemes and other mobility models introduced in the literature is detailed in the supplemental pages. We also plot the overall vehicular flow data for four consecutive Mondays, recorded in an interval of 10 minutes (Fig. 3h) and 30 minutes (Fig. 3i). The figures depict the consistent and predictable vehicle density data for both northbound and southbound traffic for all four weeks.

3.2 Aggregate Communications and Computation Capacity Estimation

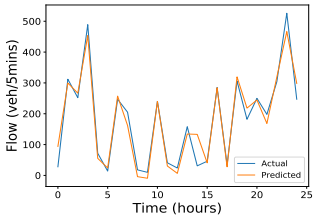
Due to the novelty of using moving vehicles as infrastructure, we estimate the communication capacity of a vehicular network. Estimating the capacity of a vehicular network is a challenging problem to solve as it depends on several factors including the average number of simultaneous transmissions, link capacities, the density of vehicles, mobility in the network, the distance between vehicles, and the transmission range of the vehicles. Our previous analysis shows that the problem of less vehicular density causing a delay in communication is not prevalent in urban centers, and even freeway traffic flow in some cases. We also demonstrated that most traffic flow prediction can be done effectively. The estimation of the capacity of the vehicular network has



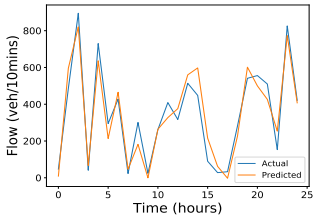
(a) LR model for traffic prediction every 5 minutes



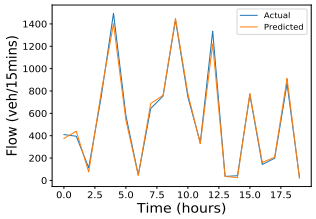
(b) Linear regression model for traffic prediction every 10 minutes



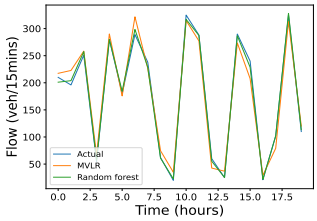
(c) Multivariate linear regression model for traffic prediction every 5 minutes



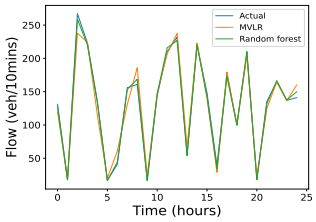
(d) Multivariate linear regression model for traffic prediction every 10 minutes



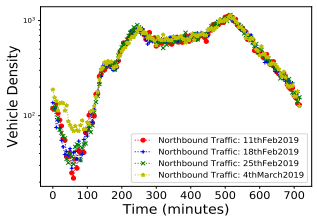
(e) Multivariate linear regression model for traffic prediction every 15 minutes



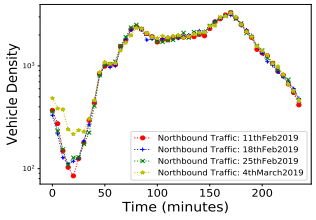
(f) Multivariate linear regression model for traffic prediction every 15 minutes, April 2020



(g) Comparison of MVLR with random forest for traffic prediction every 10 minutes, April 2020



(h) Vehicle density recorded every 10 minutes



(i) Vehicle density recorded every 30 minutes

Fig. 3: Traffic Prediction using real vehicle density data; these data depict the consistent and predictable vehicle densities at the intersection for all of the four weeks analyzed.

been done in great detail via customized theoretical studies [26]–[28]. We calculate the effective capacity of the vehicular network obtained using a cooperative scheme from Chen et al. [28].

Theoretical Capacity: We consider the closed-form expression of effective available capacity specified by Chen et al. [28], which uses a cooperative scheme to derive the communication capacity for a vehicular network. The cooperative strategy uses both Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication to increase the capacity of vehicular networks. They built an analytical framework to model the data dissemination process and derive a closed form expression of the achievable capacity, given as:

$$\begin{aligned} theoretical\_cap = \frac{L}{d} \min\{W_I(1-\exp^{-2\rho r_I}), \\ W_I(1-\exp^{-p\rho 2r_I}) \\ + \frac{W_V.c_2(d-2r_I)}{c_2.R_C + p - p\exp^{-2pr_o}} + \exp^{-p\rho 2r_o}\} \end{aligned} \quad (1)$$

where $c_2 = (1-p)p\rho(1-\exp^{-\rho 2r_o})$. In this expression L is the length of the highway segment, d is the distance between RSUs, W_I and W_V are the data rate for V2I and V2V communication respectively, ρ is the density of vehicles per meter, p is the proportion of vehicles with download requests in the range [0,1], r_I is the range of infrastructure points and r_o is the radio range of vehicles. R_C is the sensing range for the medium access control protocol. We calculate the available capacity for this case, taking the value for L as 100 km, d as 5, 10 or 15 km, W_I as 20 Mb/s, W_V as 2 MB/s, ρ as 0.03, 0.04, or 0.05. We take the radio ranges as typical values for Dedicated Short-Range Communication (DSRC) such that r_I is 400 m and r_o is 200 m. The value of R_C is taken as 300-400 m. For these values, the effective available capacity lies in the range of 5-20 Mb/s with different proportions of vehicles participating in the scheme. The density of vehicles, the use of cooperation schemes and the number of participating vehicles have a direct impact on this effective available capacity.

The potential computation capacity of a vehicle cluster is dependent on how dense the cluster is, in terms of the number of vehicles that are optimal for placement of a particular service request. The computation capacity is also based on how slow the vehicle cluster is, which can be predicted by the occupancy of a road segment, calculated as how much time vehicles take to pass over a detector. This time can also be derived as the sojourn time of vehicles with the RSU. According to the study conducted by Xiao et al. [10], predicted computation capacity is higher than 650 Gflops with a probability of 60% when the range of vehicle clusters is set to be 5m, and throughout the day the computation capacity is above this value. When the range is 10m, the predicted capacity is 1800 Gflops. With the increasing number of smart vehicles, the number of sensors, video cameras, and computation capacity should increase significantly in the next decade. This means that the infrastructure will exist to collect data, process it on the resource pool of a vehicular cluster and send it to the cloud for further processing. However, this infrastructure cannot be exploited unless services can be placed on it in such a way that the overall service objectives are met.

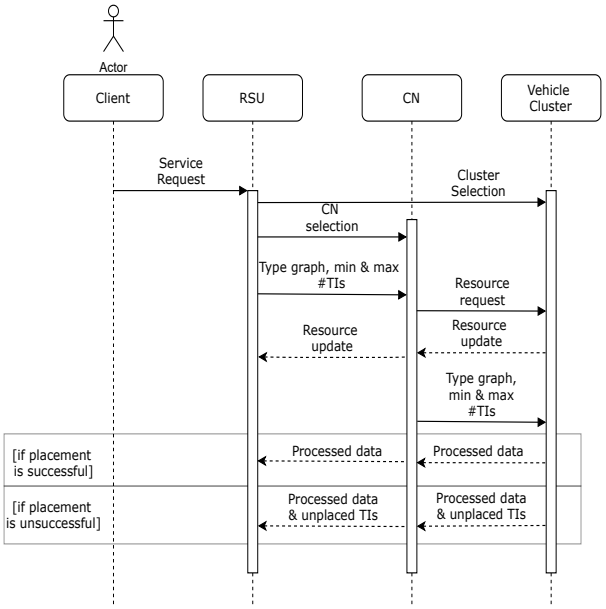


Fig. 4: System Model depicting the management between the RSU, CN and the vehicle cluster

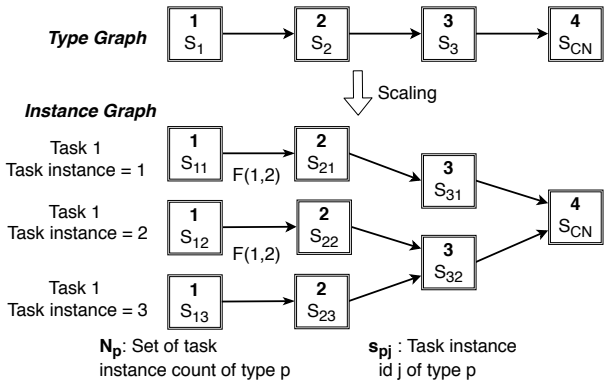


Fig. 5: Service model depicting tasks and their inter-dependencies. The Type graph is scaled to the Instance Graph based on the resource state of the vehicle cluster.

3.3 Summary

Through the motivation section, we have highlighted the predictability of vehicular flows using a linear regression-based model that predicts flows accurately and compared the model to other competing schemes. We also made aggregate communications and computation estimates for such vehicle clusters, initiated at urban intersections. We demonstrated the feasibility of using vehicles as infrastructure using real vehicular data. The next part of the paper focuses on how to utilize this moving pool of vehicular resources by introducing distributed service models and an optimization scheme to jointly optimize communication and computation resources to efficiently place services on the moving vehicles.

4 System Model

In this section we first describe the terminology of the system model; then we present the network topology and the distributed service model.

4.1 Terminology

- **Vehicle Clusters**: We consider vehicle clusters as micro cloud-like entities [29], whose members (vehicles) provide resources used to execute tasks that form a distributed service.
- **Control Node (CN)**: The CN is a vehicle in the cluster that acts as a gateway between the cluster and RSUs; is elected based on its connectivity to the RSU and other cluster nodes (this election process is outside the scope of this paper). It collects status information about the cluster, including available resources at nodes, link capacities and it also receives service placement requests from the RSU/client.
- **RSUs**: The vehicle nodes in a cluster are supported by resource-rich RSUs, which connect the cluster to the Internet. The management of services between the RSU, CN, and the vehicle cluster is depicted in Fig. 4. The RSU knows the system state of the cluster, which is communicated to it by the CN.
- **Task**: Tasks are data collection or processing functions that can be scaled out as multiple task instances (TIs) to realise a distributed service. For example, a distributed service to realise pedestrian counting may in its specification request as many vehicle cameras as possible for monitoring a given stretch of road. The TIs are the smallest unit that a task can be split into and that can be mapped to a vehicle node.
- **Service**: We consider distributed services with unidirectional, acyclic control, and data-flows. These services are specified as hierarchies of different task types, each with different functionality. Each task is typically deployed as several TIs, which can be dynamically and flexibly scaled (in terms of size per TI (*up*) and number of TIs per task (*out*)) according to resource availability and stability of the vehicle cluster at a given instant. We assume a linear chain of data-dependent tasks represented as a *Type graph*, in Fig. 5. This *Type graph* is sent as an input to the service placement function. Based on the *Type Graph*, an *Instance graph* is created, where each task of Type p (represented as s_p in Fig. 5) can have multiple TIs of Type p and count j (represented as s_{pj}). Other works that leverage parked vehicles (PVs) also deploy similar service models, where a task with a large workload is split into several sub-tasks and assigned to multiple PVs for cooperative execution [30].
- **Service Placement**: The process of placing the scaled Instance graph (in Fig. 5) on a vehicle cluster is called the service placement problem.

Our approach is to first find an optimal Instance graph, considering both service and infrastructure constraints, as this decision cannot be taken independently of the infrastructure state. This is optimized based on minimizing the total number of hops in the path between each *Type 1* TI and the CN. This step reduces the bandwidth usage and selects a dense service spread, which also reduces delay in service execution. We then map the optimized Instance graph onto the physical vehicle nodes. We jointly consider both TI mapping as well as the route/flow mapping, between the placed TIs.

We also take into account the predictable mobility pattern of these vehicles.

4.2 Network Topology

We assume that $\mathbb{I}$ nodes participate in the formation of the vehicle cluster. We represent the cluster as a directed, connected graph, $G = (V, E)$. The node $i \in V$ represents the vehicle nodes, each with K resource types, where $k \in \{1...K\}$ and $i \in \{1...\mathbb{I}\}$ denote resource type k on node i. The processing capacity of each vehicle node i in respect of resource type k is represented as $C_k(i)$. The directed edge, $(i_1, i_2) \in E$, of the graph represents the link between any two vehicle nodes i_1 and i_2. The link capacity limit is depicted as $\{B(i_1, i_2)\}$ Kb/s between any two nodes i_1 and i_2. If there is no direct connectivity, due to excessive range, line of sight difficulties and/or incompatible protocols then $B(i_1, i_2) \equiv 0$.

The mobility in the network is represented by the cluster cohesion probability (CCP) of each vehicle node ($P_{(t_1,t_2)}(i_1)$), which represents the probability of a vehicle to be in a certain segment of the road, in a particular period of time $[t_1, t_2]$. We also consider the joint probability ($P_{(t_1,t_2)}(i_1, i_2)$) of two nodes i_1 and i_2 to stay together on a given road segment over the time interval $[t_1, t_2]$, due to the inherent data dependency between two interacting task nodes, as specified in the service model. This makes it important to consider the combined probability of two nodes with data-dependent TIs to stay together until the completion of both TI tasks. We assume that this information regarding the mobility pattern, in terms of CCP of the nodes, is available at each road intersection.

4.3 Service Model: Task and Task Instances

The service model is composed of tasks, denoted as s_p, each with different functionality, to be deployed on different nodes of the cluster. The functions include video streaming, data compression/processing as well as application control, for the flexible management of infrastructure links and nodes. Each task can have any number of TIs, represented as s_{pj}, to be mapped in an optimal configuration onto vehicle nodes. The number of TIs for a task s_p is represented as $\mathbb{N}_{s_p}$. Each TI s_{pj} requires a minimum demanded amount of D_{pjk} units of each resource of type k. Furthermore, the flow *demand* between task s_{p_1} and task s_{p_2} is provided as $F(s_{p_1}, s_{p_2})$. Note that such flows might be point-to-point (between adjacent nodes) or might need to be routed via other nodes according to flow tables maintained by the CN. Both the per resource type k demand for TI s_{pj}, labeled by $\{D_{pjk}\}$, and the inter-task demand $\{F(s_{p_1}, s_{p_2}), s_{p_1} \neq s_{p_2}\}$ need to be specified as input to the model.

Each TI can support a maximum flow rate, which is derived from the processing requirement of the incoming flow, given as $C(F(s_{p_1 j}, s_{p_2 j}))$, i.e., the processing requirement for flow from TI $s_{p_1 j}$ to $s_{p_2 j}$. We check that the target TI has enough processing capacity to process an incoming flow and also ensure that this leaving flow, after being streamed or processed at an TI, is directed to a single corresponding TI. Recall that each processing TI can have multiple incoming flows. We follow this rule to promote the collocation of processing nodes, whenever vehicle nodes have available resource capacity. This promotes a balanced service placement rather than over-provisioning the available infrastructure.

We aim to minimize the cost of service execution, by favoring nodes with a higher probability of staying with the vehicle cluster and promoting a "narrower" service placement (just enough nodes for reliability in the presence of node mobility) to reduce resource bandwidth usage. The model can be used to optimize other resources like the increasing number of accepted requests on vehicle clusters, the number of nodes used or other performance metrics like latency or service bandwidth demand, based on the requirements of the application.

5 Constraints and Problem Formulation

The placement problem first scales the service type graph to an instance graph and then *maps* the service onto the vehicle cluster.

5.1 Infrastructure Constraints

5.1.1 *Node Resource Constraints*

Each resource type of a vehicle node is denoted as k, where $k = 1$ is CPU capacity, $k = 2$ is Memory capacity and $k = 3$ is sensing resource capacity. The minimum resource *requirement* (of type k) to host TI s_{pj} on node i is given as D_{pjk}. This constraint 2 checks if the TI is mapped to node i which is denoted by the binary mapping variable $M(p, j, i)$. The resource required by the placed TI must not exceed the availability of resource type k on the selected node. Since we are interested in using only spare resources for placing services on vehicular clusters, a minimum set of system resources must also be reserved for the operations required for the vehicles. Thus, the net available capacity for hosting collaborative services at every node i is represented as $C_k(i)$, which is the *capacity* of the node for such additional services. The resource constraint is formally presented as: .

$$\forall i \in \{1, \ldots, \mathbb{I}\}, k \in \{1, \ldots, \mathbb{K}\}, \sum_{\forall p,j} M(p, j, i).D_{pjk} \leq C_k(i) \quad (2)$$

where the decision variable $M(p, j, i) \in \{0, 1\}$ is set to 1 to indicate that TI s_{pj} is placed on node i or 0 otherwise. The use of this indicator variable ensures that TI s_{pj} requires resources from node i, *if and only if* it is placed on that node.

5.1.2 *Bandwidth Constraint*

The bandwidth requirement between two task s_{p_1} and s_{p_2}, where the latter requires data from the former, is represented by $F(s_{p_1}, s_{p_2})$ Kb/s. We consider only one-directional traffic, from task s_{p_1} *to* s_{p_2}. However, the model can easily be extended to consider duplex communication needs by adding extra constraints of the form 3.

$$\forall i_1 \in \{1, \ldots, \mathbb{I}\}; i_2 \in \{1, \ldots, \mathbb{I}\}; i_1 \neq i_2$$
$$\sum_{\forall p_1, j_1; p_2, j_2; p_1 \neq p_2} M(p_1, j_1, i_1) F(s_{p_1}, s_{p_2}) M(p_2, j_2, i_2) \leq B(i_1, i_2) \quad (3)$$

where $i_1 \neq i_2$ and $B(i_1, i_2) \neq 0$. Constraint 3 ensures that, for each node pair labeled by i_1 and i_2, the total bandwidth requirement, for all TI pairs $s_{p_1 j}$ and $s_{p_2 j}$ placed on nodes

i_1 and i_2 respectively, is $F(s_{p_1}, s_{p_2})$, which does not exceed the bandwidth limit $B(i_1, i_2)$ between the two nodes.

In our model, two tasks that are mapped to two different vehicle nodes i_1 and i_2, where $i_1 \neq i_2$ might not be linked directly to each other, but are connected over multiple hops. In the following bandwidth constraint, we consider the resource capacity of each link over the full path between tasks s_{p_1} and s_{p_2}. We consider another binary valued mapping variable $m(p_1, p_2, j, i)$ which takes the value 1 for each node i that is mapped to forward the flow between TIs of type $s_{p_1 j}$ and $s_{p_2 j}$ and is part of the path between the two data dependent TIs. Thus, nodes can act as both processing nodes or forwarding nodes. The constraint 4 ensures that the bandwidth used for forwarding the flow between any connected pair of forwarding nodes should be less than the available bandwidth capacity between those two nodes. This constraint is formally presented as:

$$\begin{aligned} &\forall i_1 \in \{1, \ldots, \mathbb{I}'\}; i_2 \in \{1, \ldots, \mathbb{I}'\}; i_1 \neq i_2 \\ &\sum_{\forall p_1, j_1; p_2, j_2; p_1 \neq p_2} m(p_1, p_2, j_1, i_1) F(s_{p_1}, s_{p_2}) m(p_1, p_2, j_1, i_2) \\ &\leq B(i_1, i_2) \end{aligned} \tag{4}$$

where i_1 and i_2 belong to $\mathbb{I}'_{(p_1 j)(p_2 j)}$, which is the set of all nodes on the path between TIs $s_{p_1 j}$ and $s_{p_2 j}$.

5.2 Service Model Constraints

We now formulate the constraints for placing distributed TIs and the corresponding service data flow between these TIs. We ensure that the data flow is processed before reaching the CN and the order of TIs is maintained according to the service chain or service description.

5.2.1 *Flow Rate Constraint*

As we propose a distributed service model, it is crucial to ensure that the TIs have enough processing capacity for the incoming flow. The constraint 5 ensures that the flow rate entering a TI should not exceed the processing capacity of that TI. The processing capacity of TI $s_{p_2 j}$ is represented as $C(F(s_{p_1 j}, s_{p_2 j}))$, which is the function of incoming flow from $s_{p_1 j}$ to $s_{p_2 j}$. This constraint is given as:

$$\begin{aligned} &\forall i_1 \in \{1, \ldots, \mathbb{I}\}; i_2 \in \{1, \ldots, \mathbb{I}\}; i_1 \neq i_2 \\ &\sum_{\forall p_1, j_1; p_2, j_2; p_1 \neq p_2} m(p_1, p_2, j_1, i_1) C(F'(s_{p_1 j}, s_{p_2 j})) \leq C(s_{p_2 j}) \end{aligned} \tag{5}$$

where $C(s_{p_2 j})$ represents the processing capacity of TI $s_{p_2 j}$. This TI is placed on the node that receives the incoming flow to be processed, from TI $s_{p_1 j}$. Here $F'(s_{p_1 j}, s_{p_2, j})$ represents the flow that has been processed at TI $s_{p_1 j}$ or is forwarded from $s_{p_1 j}$ specifically for processing (not forwarding).

5.2.2 *Flow Conservation Constraint*

Constraint 6 ensures that the incoming to outgoing flow rate ratio, at a node, is governed by the data processing factor of the TI. α_{pj} represents the data reduction/processing factor for a task with *Type p* resource. The constraint is presented as:

$$\begin{aligned} &\forall i_1 \in \{1, \ldots, \mathbb{I}\}; i_2 \in \{1, \ldots, \mathbb{I}\}; i_1 \neq i_2 \\ &\sum_{\forall p_1, j_1; p_2, j_2; p_1 \neq p_2} F(s_{p_1 j}, s_{p_2 j}) \alpha_{p_2 j} \leq F'(s_{p_1 j}, s_{p_2 j}) \end{aligned} \tag{6}$$

where $0 \leq \alpha_{pj_1} \leq 1$ and $F(s_{p_1 j}, s_{p_2 j})$ represents the incoming flow to be processed at TI $s_{p_2 j}$. $F'(s_{p_1 j}, s_{p_2 j})$ represents the outgoing flow, that has been processed at the TI $s_{p_2 j}$. This constraint ensures that all the necessary pre-processing is performed on the flow, at each TI before the flow reaches the CN. Since nodes in our model can be forwarding nodes, or processing nodes, or have both processing and forwarding role, the data processing factor can lie in the range from [0,1].

5.2.3 *Task Order Constraints*

Constraint 7 ensure that the flow traverses the task instance graph in the order specified by the service model, we require that once the flow is processed at one node, it is directed to the "next" node with at least one "subsequent" TI (according to the Type Graph), i.e.,

$$\begin{aligned} &\forall i_1 \in \{1, \ldots, \mathbb{I}\}; i_2 \in \{1, \ldots, \mathbb{I}\}; i_1 \neq i_2 \\ &\sum_{\forall p, j; p+1, j_2; p \neq p+1} M(p, j, i_1) F'(s_{p_1 j}, s_{p_2 j}) \geq M(p+1, j, i_2) \end{aligned} \tag{7}$$

where the decision variable $M(p, j, i_1)$ represents the TI mapped to node i_1, $F'(s_{p_1 j}, s_{p_2 j})$ is the flow processed at the node i_1. The right hand side of the equation employs mapping instance $M(p+1, j, i_2)$ to show that a subsequent TI of type $s_{(p+1)j}$ is mapped on the node i_2, which has enough resource capacity.

As forwarding nodes are introduced in constraint 3 to facilitate these multi-hop flows, it is crucial to preserve the order of tasks at the service level. To ensure that the flow is directed towards a subsequent TI, in case there is no direct path between two placed TIs, we also ensure that the forwarding node is on the path joining nodes (i_1, i_2) with TIs s_{pj} and $s_{(p+1)j}$ mapped on them. This is represented as constraint 8:

$$\begin{aligned} &\forall i_1 \in \{1, \ldots, \mathbb{I}\}; i_2 \in \{1, \ldots, \mathbb{I}'\}; i_1 \neq i_2 \\ &\sum_{\forall p, j; p+1, j_2; p \neq p+1} m(p, p+1, j, i_1) F'(s_{p_1 j}, s_{p_2 j}) \\ &\geq m(p, p+1, j, i_2) \end{aligned} \tag{8}$$

where $m(p, p+1, j, i_1)$ and $m(p, p+1, j, i_2)$ are mapping variables with 0 or 1 value. Here $m(p, p+1, j, i_1)$ represents node i_1 as a forwarding node for processed data flow $F'(s_{p_1 j}, s_{p_2 j})$, between TIs s_{pj} and $s_{(p+1)j}$, and $m(p, p+1, j, i_2)$ represents the next forwarding node for the same flow.

5.3 Cluster cohesion probability

In order to use the mobility of slow moving vehicles in our favor, it is crucial to incorporate mobility awareness in the infrastructure model. There are many ways to predict the mobility patterns of a group of vehicles. Here we consider all nodes that have a higher probability to chose a similar road segment (S_i), based on their historical mobility patterns, to be candidates for the cluster. We assume that each RSU maintains a table of known vehicle nodes, with their probability of taking a particular road segment (say S_1) at the next intersection. Vehicles that do not have entry in the table, but are willing to offer their resources, can be added to the table. However, they would be assigned the average road exit probabilities of known vehicles, with a low

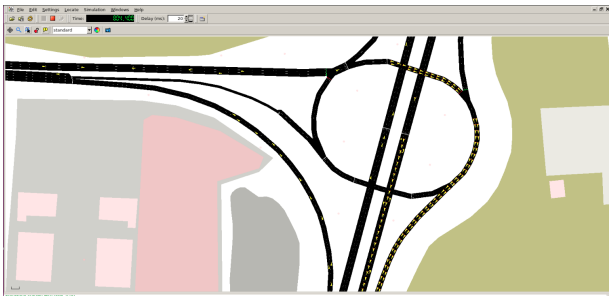


Fig. 6: The SUMO simulation for the intersection in Dublin, Ireland

confidence score. As the history of a given vehicle builds up with time, its road exit probabilities are updated and their confidence score increases.

We have calibrated the microscopic car-following model, using the macroscopic vehicle flow data from the Dublin intersection based on the flow model in §3.1. For simulations, we extract the Dublin intersection road network, as depicted in Fig. 6, using Open Street Map (OSM) and calibrate the simulation using the real-world Dublin traffic dataset. We generate the calibrated traffic in the Simulation for Urban Mobility (SUMO) simulator.

$$A = \begin{array}{c} \begin{matrix} S_1 & S_2 & S_3 & S_4 \end{matrix} \\ \begin{pmatrix} P_{(t_1,t_2)}(car1) & \dots & \dots & \dots \\ 0.5 & 0.4 & 0.1 & 0 \\ 0.4 & 0.6 & 0 & 0.1 \\ 0 & 0 & 0.9 & 0.1 \\ 0.5 & 0.5 & 0 & 0 \\ 0.6 & 0.4 & 0.2 & 0 \end{pmatrix} \end{array} \begin{matrix} \\ car\ 1 \\ car\ 2 \\ car\ 3 \\ car\ 4 \\ car\ 5 \\ car\ 6 \end{matrix}$$

The transition matrix stores the mobility behavior of every candidate vehicle, for a particular time period. This table can be updated over time to increase the accuracy of mobility awareness. Each RSU thus has many tables stored for different time stamps during the day. We model the mobility of vehicles as a Markov Model, where each road segment is a state. As mentioned in [31], the vehicle node that moves from one road segment to the other represents a transition in the Markov process. But instead of considering the detailed trajectory of a single vehicle, the matrix stores all possible probabilities for a vehicle to stay at the segment or take another road segment with a certain probability. Thus, every intersection in the service zone maintains the probability of a vehicle that follows Markov memory-less property, wherein the node transitions from state i to i+1 and is independent of state i-1. Based on the mobility patterns, different vehicle clusters can be formed for the service execution. In this paper, we only consider the nodes with a high probability of going from road segment A to C (in Fig. 2), as continuing to belong to the cluster. Therefore, the CCP of a given vehicle node is the probability of that node going straight ahead at the next intersection.

5.4 Service Placement Cost

To incorporate the mobility of hosting nodes, we scale the resource capacity of each node in the vehicle cluster with a weighting factor, i.e., the probability of a node to stay with the cluster for the duration of service execution, i.e., from time t_1 to t_2, which is given as $P_{(t_1,t_2)}(i)$. This is because a node with enough resource capacity might not have a high probability of staying with the vehicle cluster, so this needs to be considered when placing TIs on that node. Placing TIs on such nodes can waste computation and bandwidth resources if the node leaves the cluster prematurely, and can also cause the service to fail. Thus, we scale the vehicle node capacity with its CCP, such that the *higher* the CCP (probability of staying with the cluster), the *lower* the costs of TI execution on that node. The Node Cost is given as:

$$\text{NodeCost}(i_1) = \sum_{\forall i_1,i_2;i_1\neq i_2} (1 - P_{(t_1,t_2)}(i_1)).(D_{pj,k}/C_k(i)).M(p,j,i) \tag{9}$$

where $M(p,j,i)$ is the mapping function of TI s_{pj} to node i, with node resource capacity of $C_k(i)$. To add the costs, we consider the ratio of required node capacity (D_{pjk}) with the available node capacity $(C_i(k))$.

Similarly we scale the link capacity of any two nodes with data-dependent TIs, with the joint probability of the two nodes to stay together for the duration of service execution (t_1 to t_2), given as $P_{(t_1,t_2)}(i_1,i_2)$. The total link cost for service execution is given as:

$$\begin{aligned} \text{LinkCost}(i_1,i_2) = \sum_{\forall i_1,i_2;i_1\neq i_2} (1 - P_{(t_1,t_2)}(i_1,i_2)).\\ (F(p_1,p_2)/B(i_1,i_2))\Big(m(p,p+1,j_1,i_1).m(p,p+1,j_2,i_2)\\ +M(p,j_1,i_1).M(p,j_2,i_2)\Big) \end{aligned} \tag{10}$$

where $m(p,p+1,j_1,i_1).m(p,p+1,j_2,i_2) \in \{0,1\}$ is an indicator that two nodes that form part of the path joining two TIs of task s_{p_1j} and $s_{(p+1)j}$ type. Similarly $M(p_1,j,i_1).M(p_2,j,i_2)$ indicates that two nodes, one hosting TI s_{p_1j} at node i_1 and the other TI s_{p_2j} at node i_2, have a direct link between them. For adding up the link cost and the operating cost on each node, we use the ratio of required bandwidth resource $(F(s_{p_1j},s_{p_2j}))$ with the available bandwidth $(B(i_1,i_2))$ at each link that forms part of the service placement.

5.5 Objective Function

The problem is formulated as a bi-objective optimization. We hierarchically solve the optimization with the first objective as:

5.5.1 *Adjacency TI placement*

When placing tasks on nodes, it is more efficient to ensure that the placement plan takes account of both task dependencies and of inter-node network distances. For example, if s_{p_2} depends on s_{p_1}, it is advisable to ensure that each is placed either on the same node or on nodes that are one hop away from each other. However, this requirement could make it difficult to find a feasible placement. Hence, we seek to ensure that the network distance between any two selected nodes with data dependency is minimized for efficient service placement. The hop count between two placed TIs is minimized when:

$$\begin{aligned} \forall i_1 \in \{1,\dots,\mathbb{I}'_{(p_1j)(p_2j)}\}; i_2 \in \{1,\dots,\mathbb{I}'_{(p_1j)(p_2j)}\}; i_1 \neq i_2\\ H(i_1,i_2) = \sum_{\forall p_1,p_2} m(p_1,p_2,j,i), \min\ H(i_1,i_2)), \end{aligned} \tag{11}$$

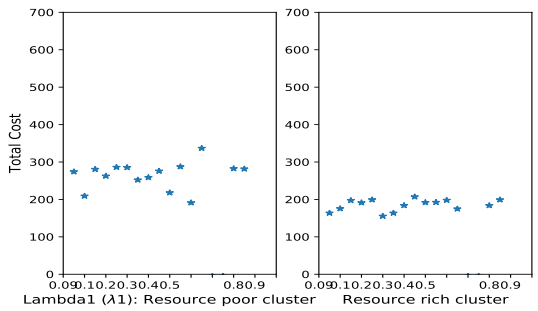


Fig. 7: Sensitivity analysis for resource-rich and resource poor cluster, we get total cost values in a similar range for almost every weight. We chose both λ_1 and $\lambda_2 = 0.5$, as it gives lower cost in both cases and other values do not significantly affect cost.

where $H(i_1, i_2)$ is the hop count between two nodes i_1 and i_2 for the flow $F(s_{p_1 j}, s_{p_2 j})$ between tasks s_{p_1} and s_{p_2}. In the model, mapping variable $m(p_1, p_2, j, i)$ applies to a forwarding node i along the path between TIs j of type p_1 and p_2, in cases where there is no direct link between the nodes hosting these TIs.

5.5.2 *Total Cost of Service Placement*

We then solve the model for the next objective function, which minimizes the Total Cost spent on service execution:

$$\min \sum_{\forall i_1, i_2; i_1 \neq i_2} \lambda_1 \text{LinkCost}(i_1, i_2) + \lambda_2 \text{NodeCost}(i_1) \qquad (12)$$

where λ_i are non-negative and sum to 1. When evaluating our model, we set $\lambda_1 = \lambda_2 = 0.5$, i.e., we give equal weight to node and link cost for simplicity. To come to this decision, we carry out a sensitivity analysis and generate random values for λ_1 (λ_2 = 1 - λ_1), and plot total cost for the same resource-poor and resource-rich clusters. In the event of low node capacity or low link capacity, the optimization takes care of the number of TIs deployed, with the objective of minimizing total cost. Thus, as depicted in Fig. 7, we get total cost values in a similar range for almost every weight. We chose both λ_1 and $\lambda_2 = 0.5$, as it gives lower cost in both cases and other values do not significantly affect cost. Also, in hierarchical optimization, the first objective function effectively gets a higher priority than the next. We give an explanation of this choice in Section 7.2. Thus, minimizing the hop counts is the first priority of the optimization and then equal priorities are given to both node and link cost.

6 Service Scaling and Placement Plan

In this section, we explain our procedure for service scaling and placement. As presented in Algorithm 1, the placement procedure first take the mobility parameter of vehicles as the CCP ($P_i(t_1, t_2)$), the available capacity for hosting service at every node i ($C_k(i)$)and bandwidth limit between the two nodes ($B(i_1, i_2)$). The linear service chain or the *Type graph* is also given as an input to the procedure. The *Type graph* is composed of several tasks that are scaled to TIs according to the demand of the service. The minimum resource requirement to host a TI of D_{pjk} units and the flow demand between two tasks $F(s_{p_1}, s_{p_2})$ is also given as an input to the service placement procedure. The service is placed by placing each TI in the order specified by the *Type graph*.

Algorithm 1 Service Scaling and Placement

Input: Mobility and resource state of vehicles: $P_i(t_1, t_2)$, $C_k(i)$, $B(i_1, i_2)$, Linear service chain: D_{pjk}, $C(F(s_{p_1 j}, s_{p_2 j}))$, $F(s_{p_1}, s_{p_2})$, $C(s_{p_2 j})$
Output: Service placement plan with minimized hop count and service placement cost

1: **procedure** SERVICE SCALING
2: Place Type 1 TIs on vehicles with data collection capability
3: **for** each vehicle pair i_1,i_2 in cluster **do**
4: **for** each TI Type n to Type m in Task order **do**
5: **if** Type n is placed **then** ▷ Call service placement procedure to check capacity and service-level constraints
6: $H(i_1, i_2)$, Cost = Service Placement()
7: **else if** Unplaced **then**
8: Scale new TI of Type n
9: $H(i_1, i_2)$, Cost = Service Placement()
10: **if** Type n is placed **then**
11: total_hop += $H(i_1, i_2)$
12: total_cost += Cost
13: **if** Type n to Type m placed **then**
14: **if** $H(i_1, i_2) \leq$ min_hop && Total Cost $\leq$ min_cost **then**
15: min_cost = total_cost
16: min_hop = total_hop
17: **return** New service placement plan
18: **else**
19: Continue to find placement until all nodes of the cluster are explored
20: **procedure** SERVICE PLACEMENT($P_i(t_1, t_2)$, $C_k(i)$, $B(i_1, i_2)$, D_{pjk}, $C(F(s_{p_1 j}, s_{p_2 j}))$, $F(s_{p_1}, s_{p_2})$, $C(s_{p_2 j})$)
21: **if** node and link capacity constraints are met **then**
22: **if** service level constraints are met **then**
23: Place Type n on node i_2
24: Calculate $H(i_1, i_2)$
25: Calculate Cost = λ_1 Link Cost(i_1, i_2) +
26: λ_2 NodeCost(i_1)
27: **return** Cost,$H(i_1, i_2)$
28: **else**
29: **return** Unplaced TI

The service scaling procedure first checks if a TI of a certain task type is already placed on the vehicle cluster (line 5). If a TI is placed then the service placement procedure is called to check the infrastructure and service-level constraints (in lines 20-29) on the TI and the node hosting it. If a TI is not already placed or the constraints are not met on the placed TI, a new TI is scaled on line 8 and the service placement procedure is called on line 9. As each TI is placed, the total hop count and the cost are added in lines 10-12. If all TIs of Type n to m is placed, the total hop count and the total cost are compared to the minimum hop count and

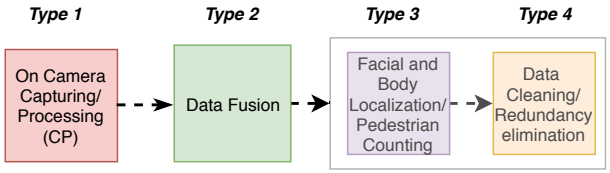


Fig. 8: Sample application for Pedestrian Detection

minimum total cost of an existing service placement plan to find the placement with the lowest objective value in lines 13 to 19.

7 Results

7.1 Application Types

We highlight two different application types that are suitable for the model described in this paper. The type-based service for distributed video analytics is given as an input to the service placement problem. The service is described as a linear chain, with one task of *Type 1* type, that is mapped to a vehicle with a dash camera or smart camera installed on it and the user is willing to lease their vehicle resources in exchange for some incentive. This data is streamed to a nearby vehicle that hosts a task of *Type 2*, followed by another vehicle hosting a task of *Type 3*. Such tasks execute lightweight video pre-processing like data compression or sub-sampling that reduces the size of the video data, based on the application requirement. Some examples include:

- Modality based pre-processing [32]: multimedia data may have more than one modality, e.g., video data with image and speech. This requires data separation.
- Data cleaning: only frames that have the required data can be separated from other redundant frames, especially in the case of more than one source of video data. This is relevant for a Fog computing scenario, where the computation and storage capacity is limited.
- Data Reliability: Other application like detecting video from unreliable data sources which are not subscribed to the service can also be detected and filtered at this stage.

Once the processing is complete at the cluster, this data is then sent to the CN which forwards the data to the edge/cloud for further high computational processing, like vision-based processing for video crowd-sourcing applications and traffic density estimation using convolutional neural networks, etc. We specify two different applications, with different resource requirements, that we place together on the vehicle cluster:

7.1.1 Application I: High-processing video streaming applications

The first application is a *pedestrian detection application* that can be used to study the popularity of a coffee shop or a gas station, based on the number of pedestrians detected in the stream of video data. This data is collected by vehicles standing at a traffic light or an intersection, close to the coffee shop, say. This data has local relevance/scope and hence, most of this data should be processed locally, based on the available resources on the vehicle cluster. For this

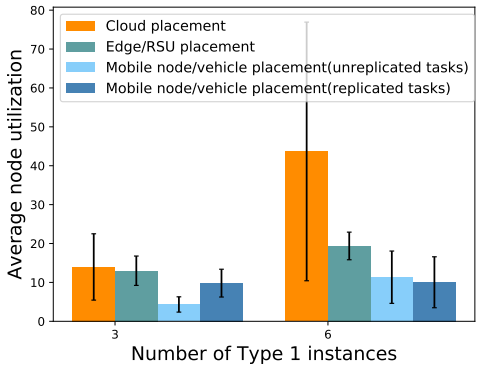


Fig. 9: Comparing different placement techniques using the average node utilization in the case of: cloud placement, edge/RSU placement, mobile node/vehicle placement (without replicating task instances), mobile node/vehicle placement (with replicated task instances: **our case**) for Application I

application, 1 to 6 camera or *Type 1* TIs are used in the Evaluation section (7.2), because more camera instances increase the richness of contextual data. The *Type 2* TIs aggregate and process this video stream from different *Type 1* TIs as depicted in the Concept Diagram of the application in Fig. 8. This TI can aggregate the data from different sources based on content or location similarity. The functionality of *Type 3* and *Type 4* TIs is application-specific. In the compute-intensive application, lightweight video processing is performed on the video stream to transform it into other forms, e.g., capturing specific frames with license plates, or highlighting pedestrians or other objects of interest in each scene. We assume that the data is reduced to 40-50% of its size, by *Type 2* instances and to 20% of its original size after processing by *Type 3* TIs. This pre-processed data is then sent to the CN, which forwards it to the RSU.

To validate the service model, we implement this applications on an existing simulator called Yet Another Fog Simulator (YAFS) [33]. It is a python-based discrete-event simulator that supports resource allocation and network design in Cloud, Fog or Edge Computing systems. We chose the simulator because it supports mobility of entities, which can act as both sources of data, called workloads or processing nodes. The simulator also provides a Distributed Data Flow based application model that allows task replicas and dynamic placement of tasks. The applications are represented as directed acyclic graphs (DAGs), where nodes represent service modules and links represent data dependency between modules. The simulator also incorporates strategies for dynamic service selection, placement and routing.

In Fig. 9, we consider the average node utilization in the placement of service described as Application I. As suggested by the authors in [33], we calculate the node utilization as the sum of the service times at each node divided by the total simulation time. We compare the average node utilization between:

- Cloud placement: all task placed in Cloud
- Edge/RSU placement: all tasks placed on edge/RSU
- Mobile node/vehicle placement (unreplicated tasks):

all tasks placed on mobile nodes/vehicles (without replicated TIs)
- Mobile node/vehicle placement (replicated tasks) (our approach): all tasks placed as multiple TIs on mobile nodes/vehicle

In Fig. 9, for variable workloads that are generated using custom temporal distributions, we compare placement for three and six video collection TIs of Type 1. For three Type 1 TIs, only mobile node placement with unreplicated tasks result in better node utilization, compared to our approach. For six Type 1 TIs, our approach of replicating processing TIs of Type 2 and Type 3 on different mobile nodes, results in lesser average node utilization compared to all the other approaches. This validates that our service model of replicating tasks is efficient from node utilization point-of-view, as compared to other placement approaches.

7.1.2 Application II: Low-processing video streaming application

Application II uses vehicles as moving sensors for video collection. Applications of this category includes measuring the traffic density at an intersection in real-time, or surveying road conditions for road traffic mapping. Generally, the focus is on passive video collection; most processing does not happen in the cluster. Such applications perform minor pre-processing tasks on data in the vehicle cluster. Such pre-processing includes data sampling, segmentation or encoding and is carried out on *Type 2* and *Type 3* instances. Thus, the data is reduced to 80% of its original size before being sent to the cloud for executing compute-intensive tasks, possibly applying complex machine learning to the data.

7.2 Evaluation

We solve the constrained optimization problem using the Gurobi Optimizer, which is a powerful mathematical sover, on an Intel i7-6500U dual-core processor running at 2.50 GHz. The solver uses a Linear Programming (LP) based branch and bound algorithm to solve the Mixed Integer Programming (MIP) problem.

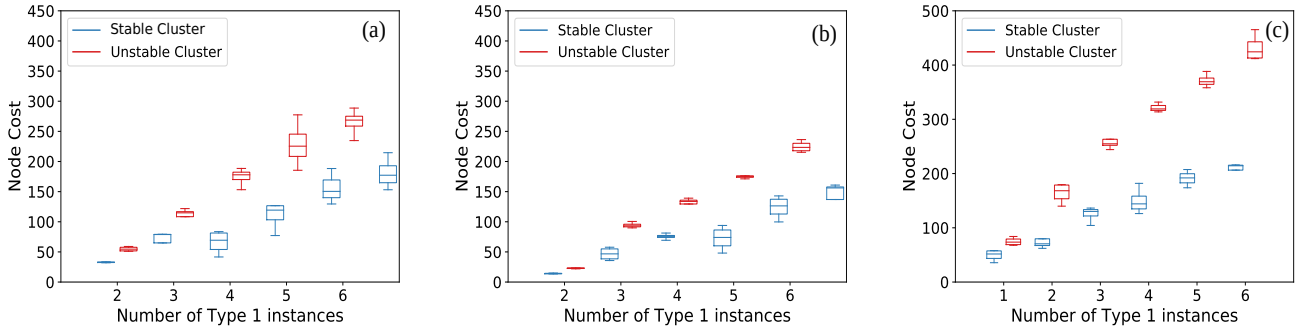
We place Applications I and II together on a vehicle cluster with 10 nodes, since more nodes in a cluster increases the time and space complexity of the problem. The cluster is a directed, connected graph, where each node has either video capturing or data processing functionality. We consider two types of resource states of the cluster, based on the mix of vehicles with one of three resource profiles: 1) Large node type: 5 CPUs, 500Mb disk, 6MB/s bandwidth; 2) Medium node type: 3 CPUs, 250Mb disk, 4MB/s bandwidth; and 3) Small node type: 2 CPUs, 100Mb disk, 2MB/s bandwidth. A *resource-rich cluster* has 50% large, 25% medium and 25% small resource vehicle nodes. A *resource-poor cluster* has 25% large, 50% medium and 25% small vehicle nodes. We consider a service chain with 2 processing instances, which makes the chain length = 3, including Type 1 instances and the CN. We ran the optimization for the longer chain length, which takes a much longer time to find a solution, specially for a higher number of video generating instances, with higher data rate. The worst case scenario was for a service chain of length 6 with 5 Type 1 instances, which took more than 5 hours to find a solution.

The 'type graph' is scaled as an 'instance graph', with data dependency and resource requirements. We use a service chain description similar to [34], without making it bidirectional. We impose multi-tenancy in the model, as it is beneficial to share TIs between applications, especially when more than one task replica is placed on the vehicle cluster.

For this paper, we consider that all nodes stop at an intersection and the RSU first selects a CN, which is one hop away from the RSU and is well connected to more than 70-80% of the nodes in the cluster. This CN needs to have ample communication and computation resources to manage the resource and cluster state. We also assume that the mobility behavior, in terms of the CCP is based on the mobility pattern of each vehicle, collected over its previous trips in this area. We derive the CCP by running the calibrated SUMO simulator, using the real vehicle density data from Dublin traffic, as explained in §5.3. We have broadly classified cluster states as stable and unstable. The stable clusters are formed when many vehicles follow a single trajectory, along with the CN. We consider two cluster states: *stable* with a CCP in the range [0.4,0.8] and *unstable* with a probability distribution between [0.2,0.6].

We consider three use cases for solving the optimization. For Case A, we take a resource-constrained cluster with low data rates of streaming video, and compare the node processing cost for stable and unstable cluster probabilities. We vary the number of Type 1 instances from 2 to 6, to study the effect of the amount of data on service placement and resource usage. When we use lower video data rates, we see that the stable cluster uses less resources than the unstable cluster. For Case B, resource-rich case (Fig. 10b) with lower data rates, the node cost is significantly less, compared to Case A (Fig. 10a), as it is easier to place more than one TI on nodes having more processing resources, for both stable and unstable cluster, resulting in better resource utilization. But in this case, the stable cluster still used less resource than the unstable cluster. The solution time is also significantly less for a resource-rich cluster: to find the optimal placement for Case B takes an average of 86s, versus a resource-constrained cluster (Case A: 300.7s). We also observed that weighting both objectives (adjacency TI placement and total cost of service placement) equally solves the problem faster than hierarchical solving, but the resulting placement uses more network resources.

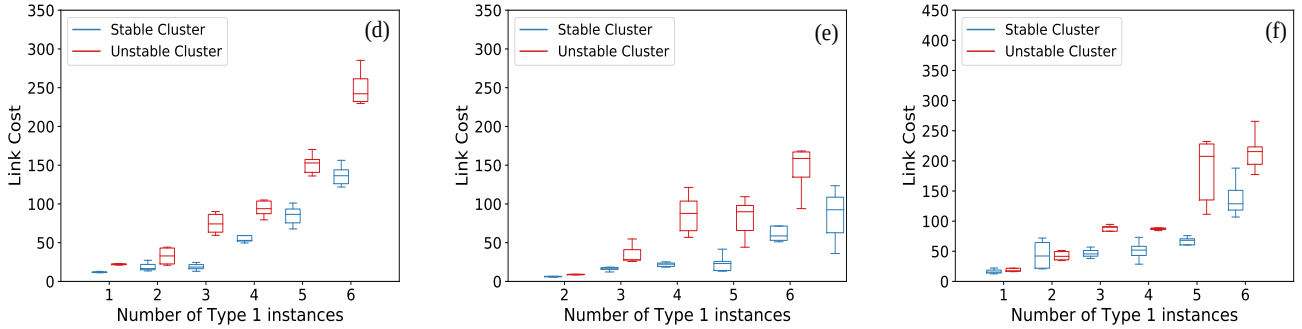
For link cost, the resource-constrained cluster (Case A) has significantly higher resource usage (Fig. 10d). The nearby nodes might not have enough processing capacity, so dependent TIs need to be placed on farther nodes, leading to more link utilization. The link capacity is also less in the resource-constrained cluster which adds to the cost. The Link Cost in the resource-rich cluster (Case B) (Fig. 10e) is significantly less and, in both cases, stable clusters outperform unstable clusters. The variability in link cost is more in this case, as the amount of video data processing in both applications is significantly different. Application I reduces the data to approx. 20% whereas Application II reduces the data to 80%. Hence, the link cost varies based on the number of *Type 1* TIs in each application. But as we double the data



(a) **Case A:Node cost for resource-constrained cluster and low data rate**

(b) **Case B:Node cost for resource-rich cluster and low data rate**

(c) **Case C:Node cost for resource-rich cluster and high data rate**



(d) **Case A:Link cost for resource-constrained cluster and low data rate**

(e) **Case B:Link cost for resource-rich cluster and low data rate**

(f) **Case C:Link cost for resource-rich cluster and high data rate**

Fig. 10: Node and Link Cost for Case A,B & C

rate of the video data in a resource-rich cluster (Case C), the unstable cluster utilizes much more computation resource (Fig. 10c). The difference between stable and unstable cluster node cost increases significantly as the Type 1 instances increase from 1 to 6. The unstable cluster uses slightly more resources, compared to the stable cluster in the low data rate case (Case B: Fig. 10b). For the link cost in this case (Fig. 10f), for fewer Type 1 TIs, stable and unstable cluster incur almost the same cost. The variability increases as the number of video instances increases.

7.3 Comparison of MIP with baseline approaches

We compare the optimal solution of our model to an autonomous vehicular edge computing-based naive solution introduced in [13] and a clustering-based solution introduced in [14]. The work in [13] is very similar to our concept of using the highly dynamic vehicular environment for deploying services. The solution selects combinations of nodes with the intention of minimizing latency, i.e., they have the lowest processing time and transmission time. As we focus on data collection services, we do not focus on the time needed to send results to the requesting node (typically the CN). As described in [21], they preferentially select nodes with the highest available link and node capacities.

The clustering-based approach in [14] uses graph-based services, similar to our service model, and generates *Herds* or clusters using the K-means clustering algorithm. To make the approach comparable, we use parameters like available resources ($C_i(k)$ in our model), CCP, and vehicular speed to form clusters. The mobility and resource state of the cluster are collected from all participating vehicle nodes. The first centroid is selected randomly, the remaining k-1 centroids are selected based on the maximum squared distance from the nearest centroid. The classical k-means clustering is used to form k clusters. The intra-cluster squared distance (ICD) is calculated for each cluster and the cluster with the smallest ICD is selected for service placement. As a baseline, the nodes are randomly selected from the generated cluster to offload and process the tasks. However, this approach introduced in [14] only considers the processing cost, and hence, we only use the node cost to compare their clustering approach to other approaches.

As seen in Fig. 11, the node cost for the naive approach increases significantly as the number of Type 1 instances increases. Similarly, the node cost for the clustering approach increases linearly as the number of TIs of Type 1 increases. We get a similar result for link cost in Fig. 12 where the cost doubles for the naive approach for 5 and 6 Type 1 TIs, in comparison to the optimal solution. The naive approach results in less latency compared to the optimal approach but do not take account of node mobility, so is more likely to fail. By contrast, our objective reduces the cost of service execution *and* selects more reliable nodes that reduce the need for service reconfiguration. Of course, the delay is a crucial parameter for safety-related services like lane changing, accident prevention, and autonomous driving. However, delay can also be reduced by adding more resources and using them judiciously: by shortening service chains and placing many processing TIs of the same type in parallel.

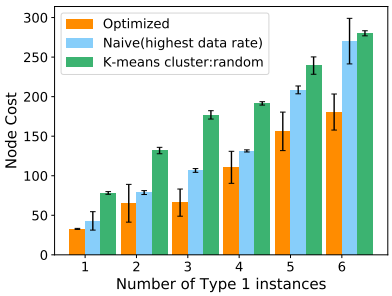


Fig. 11: Comparison of node cost in optimal, naive scheme and a clustering scheme for Case A:resource constrained cluster and low data rate

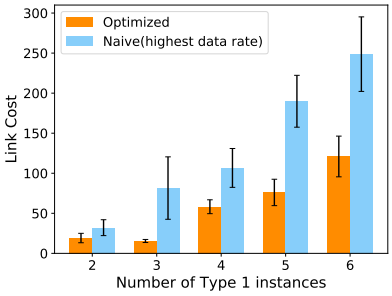


Fig. 12: Comparison of link cost in optimal v/s naive scheme for Case A:resource constrained cluster and low data rate

8 Conclusion and Future Work

This paper focuses on the concept of scaling and placement of distributed services on vehicle clusters, harnessing the knowledge of mobility patterns. The novelty of the work is in considering the mobility pattern of urban road traffic and utilizing moving vehicles as a potential site for deploying services. The services are made adaptable for the dynamic vehicular environment and can be scaled dynamically, based on the resource and mobility state of the multi-hop cluster. We have introduced a flow model for the traffic, depicted predictability in vehicular flow, and estimated communication capacity using real vehicular traffic data. We also introduced a detailed mathematical model for the mobility-aware scaling of distributed services based on resource-rich and resource-poor as well as stable and unstable cluster states. We solved the constrained bi-objective optimization problem, introduced data collection and data pre-processing applications, and validated our model for different resource and mobility states. Our approach outperforms the naive solution introduced in [13] and the clustering-based approach introduced in [14] significantly.

As part of the future work, a decentralized, mobility-aware task offloading algorithm will be introduced that solves the optimization problem in real-time. To make the service model more practical, we will introduce a distributed service reconfiguration scheme to send collected data or service states back to the vehicle cluster. We aim to use hyper-parameter optimization techniques to decide the number of TIs to be deployed in real-time, for satisfying the service placement requirements. We will focus on the replacement of concurrent, data-dependent tasks as part of the failure recovery scheme.

References

[1] Z. Ning, J. Huang, and X. Wang, "Vehicular fog computing: Enabling real-time traffic management for smart cities," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 87–93, February 2019.

[2] C. Huang, R. Lu, and K. R. Choo, "Vehicular fog computing: Architecture, use case, and security and forensic challenges," *IEEE Communications Magazine*, vol. 55, no. 11, pp. 105–111, 2017.

[3] S.-S. Lee and S. Lee, "Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 450–10 464, 2020.

[4] X. Hou *et al.*, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3860–3873, 2016.

[5] X. Li, X. Huang, C. Li, R. Yu, and L. Shu, "Edgecare: Leveraging edge computing for collaborative data management in mobile healthcare systems," *IEEE Access*, vol. 7, pp. 22 011–22 025, 2019.

[6] Z. Ning, J. Huang, X. Wang, J. J. P. C. Rodrigues, and L. Guo, "Mobile edge computing-enabled internet of vehicles: Toward energy-efficient scheduling," *IEEE Network*, pp. 1–8, 2019.

[7] I. W. Ho, S. C. Chau, E. R. Magsino, and K. Jia, "Efficient 3d road map data exchange for intelligent vehicles in vehicular fog networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 3151–3165, 2020.

[8] H. Du, S. Leng, F. Wu, X. Chen, and S. Mao, "A new vehicular fog computing architecture for cooperative sensing of autonomous driving," *IEEE Access*, vol. 8, pp. 10 997–11 006, 2020.

[9] D. Ye, M. Wu, S. Tang, and R. Yu, "Scalable fog computing with service offloading in bus networks," in *2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*, 2016, pp. 247–251.

[10] X. Xiao, X. Hou, X. Chen, C. Liu, and Y. Li, "Quantitative analysis for capabilities of vehicular fog computing," *Information Sciences*, vol. 501, pp. 742 – 760, 2019.

[11] Z. Zhou, H. Liao, X. Wang, S. Mumtaz, and J. Rodriguez, "When vehicular fog computing meets autonomous driving: Computational resource management and task offloading," *IEEE Network*, vol. 34, no. 6, pp. 70–76, 2020.

[12] K. Sharma, B. Butler, B. Jennings, J. Kennedy, and R. Loomba, "Optimizing the placement of data collection services on vehicle clusters," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2018, pp. 1800–1806.

[13] J. Feng, Z. Liu, C. Wu, and Y. Ji, "Ave: Autonomous vehicular edge computing framework with aco-based scheduling," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10 660–10 675, Dec 2017.

[14] S. Hu, G. Li, and W. Shi, "Lars: A latency-aware and real-time scheduling framework for edge-enabled internet of vehicles," *IEEE Transactions on Services Computing*, pp. 1–1, 2021.

[15] M. Gerla, "Vehicular cloud computing," in *2012 The 11th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, June 2012, pp. 152–155.

[16] X. Ma, J. Zhao, Q. Li, and Y. Gong, "Reinforcement learning based task offloading and take-back in vehicle platoon networks," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, May 2019, pp. 1–6.

[17] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7944–7956, Aug 2019.

[18] C. Zhu, G. Pastor, Y. Xiao, and A. Ylajaaski, "Vehicular fog computing for video crowdsourcing: Applications, feasibility, and challenges," *IEEE Communications Magazine*, vol. 56, no. 10, pp. 58–63, 2018.

[19] X. Xiao, X. Hou, C. Wang, Y. Li, P. Hui, and S. Chen, "Jamcloud: Turning traffic jams into computation opportunities – whose time has come," *IEEE Access*, pp. 1–1, 2019.

[20] X. Kong, F. Xia, Z. Ning, A. Rahim, Y. Cai, Z. Gao, and J. Ma, "Mobility dataset generation for vehicular social networks based on floating car data," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 3874–3886, May 2018.

[21] C. Zhu *et al.*, "Folo: Latency and quality optimized task allocation in vehicular fog computing," *IEEE Internet of Things Journal*, pp. 1–1, 2019.

[22] W. Zhang, Z. Zhang, and H. Chao, "Cooperative fog computing for dealing with big data in the internet of vehicles: Architecture and hierarchical resource management," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 60–67, 2017.

[23] F. Hagenauer *et al.*, "Vehicular micro clouds as virtual edge servers for efficient data collection," in *Proceedings of the 2Nd ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services*, ser. CarSys '17. New York, NY, USA: ACM, 2017, pp. 31–35. [Online]. Available: http://doi.acm.org/10.1145/3131944.3133937

[24] M. Goudarzi, H. Wu, M. S. Palaniswami, and R. Buyya, "An application placement technique for concurrent iot applications in edge and fog computing environments," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2020.

[25] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Edgevcd: Intelligent algorithm-inspired content distribution in vehicular edge computing network," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5562–5579, 2020.

[26] M. Grossglauser and D. Tse, "Mobility increases the capacity of ad-hoc wireless networks," in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, vol. 3, 2001, pp. 1360–1369 vol.3.

[27] G. Mao, Z. Lin, X. Ge, and Y. Yang, "Towards a simple relationship to estimate the capacity of static and mobile wireless networks," *IEEE Transactions on Wireless Communications*, vol. 12, 06 2013.

[28] J. Chen, G. Mao, C. Li, W. Liang, and D. Zhang, "Capacity of cooperative vehicular networks with infrastructure support: Multiuser case," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 2, pp. 1546–1560, 2018.

[29] T. Higuchi, F. Dressler, and O. Altintas, "How to keep a vehicular micro cloud intact," in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, 2018, pp. 1–5.

[30] J. Zhang, X. Huang, and R. Yu, "Optimal task assignment with delay constraint for parked vehicle assisted edge computing: A stackelberg game approach," *IEEE Communications Letters*, pp. 1–1, 2019.

[31] L. A. Maglaras and D. Katsaros, "Social clustering of vehicles based on semi-markov processes," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 1, pp. 318–332, Jan 2016.

[32] L. Ang, K. P. Seng, G. K. Ijemaru, and A. M. Zungeru, "Deployment of iov for smart cities: Applications, architecture, and challenges," *IEEE Access*, vol. 7, pp. 6473–6492, 2019.

[33] I. Lera, C. Guerrero, and C. Juiz, "Yafs: A simulator for iot scenarios in fog computing," *IEEE Access*, vol. 7, pp. 91 745–91 758, 2019.

[34] S. Dräxler, S. Schneider, and H. Karl, "Scaling and placing bidirectional services with stateful virtual and physical network functions," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, June 2018, pp. 123–131.

KANIKA SHARMA [S'16] is a Ph.D. researcher at the Walton Institute, Waterford Institute of Technology, Ireland. Her research is funded by Science Foundation Ireland (SFI) funded CONNECT Research Center. She was selected as an NGI Explorer to extend her work with the University of Tennessee at Chattanooga. Her research is focused on building Fog Computing models for Intelligent Transport Systems.

BERNARD BUTLER [S'10, M'16] received his PhD degree from Waterford Institute of Technology (WIT), Ireland. He lectures in machine learning and security and is a postdoctoral researcher in the Walton Institute, WIT, where his research interests include the management of distributed computing and sensing systems, applied to future networking, smart cities and agriculture.

BRENDAN JENNINGS [(M'05)] received the BEng. and PhD degrees from Dublin City University, Dublin, Ireland, in 1993 and 2001, respectively. He is currently Vice President for Research and Innovation at TU Dublin, Ireland's first Technological University. He is a Principal Investigator in CONNECT, the Science Foundation Ireland (SFI) Research Centre for Future Networks and Communications. Brendan served on the SFI Expert Advisory Board for Contact Tracing and has provided expert commentary to RTÉ (the Irish national broadcaster), the BBC, and the Irish Times, on contact tracing apps. He was Executive Chair for IEEE ICC 2020, a flagship international conference in communications networking. Previously, Brendan has worked as Dean of Graduate Studies for Waterford Institute of Technology, and has spent periods as a Visiting Research with KTH Royal Institute of Technology in Sweden, and with EMC2 Research Europe in Ireland. His research interests include network management, vehicular networks, and molecular communications.