Placement of Distributed Services on Vehicle Clusters for Sensing Applications

Kanika Sharma, B.Tech

Supervisors: Dr Bernard Butler
Dr Brendan Jennings

School of Science and Computing
South East Technological University

Thesis submitted in partial fulfilment of the requirements for the award of
Doctor of Philosophy

January 2023

This thesis is dedicated to my loving *parents* - *Arun Sharma* and *Dr Rajni Sharma*, and the rest of my wonderful family.

tū shāhīñ hai parvāz hai kaam terā
tire sāmne āsmāñ aur bhī haiñ
Translated as: You are a falcon, you have to fly!
You have a vast sky before you. Soar!
—ALLAMA IQBAL

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Signed

Kanika Sharma, B.Tech
January 2023

Acknowledgements

This work would not have been possible without the support and efforts of many people who have helped me throughout my study.

The two people who have been my biggest supporter in the last few years are my supervisors, Dr Bernard Butler and Dr Brendan Jennings. Both were a constant source of encouragement and inspiration from the first day of my PhD degree. Dr Bernard Butler had the patience to guide me through the proposal writing process, gave insightful tips on formulating research problems and walked me through the process of carrying out valuable research work. He would gently nudge me to improve my work's shortcomings and had the patience to read my drafts multiple times without ever getting frustrated at my mistakes. Dr Brendan Jennings guided me to build up my research work from the ground up, focusing on smaller problems first. His vision for my PhD helped me to keep going on days when I felt like my work was not heading in the right direction. His scientific guidance, attention to detail and ideas in communicating research ideas to the community are things I will take away with me at the end of the study. Brendan and Bernard always encouraged me to take up research projects, and internships provided me with opportunities to work on international conference committees and research committees at SETU while ensuring my PhD research was on track. It would be impossible to find a supervisory team like them. They were always a call away to check on my posters, presentations, and papers and always approached me with great compassion. I will forever be grateful to them for trusting in me and helping me push through this journey.

I would also like to thank Dr Alan Davy for selecting me for the PhD study and for always encouraging me as the head of the Emerging Networks Lab (ENL). I am also very grateful to Dr Deirdre Kilbane for her immense support as the head of ENL and the Director of Walton Institute. I have been fortunate to have had the opportunity to work under her on projects. She always took time for me, motivating me on days when I felt my research study was stagnant. She encouraged me to write proposals, and think out of the box and was always available to answer all my queries even with her busy schedule. She has helped me to become a better researcher and team player, and I appreciate her support very much.

I also want to thank Science Foundation Ireland's Connect Research Centre and Intel Research Labs, Dublin, for funding my study. I appreciate the support they have provided me over the years. I am very thankful to our colleagues at Intel, John Kennedy and Dr Radhika Loomba, for their discussions and collaboration. Radhika has become a very close friend over the years. I would also like to thank the NGI Explorer project for supporting my visit to the University of Tennessee at Chattanooga, where I worked with the Center of Urban Informatics and Progress (CUIP). I am also very grateful to the head of CUIP, Dr Mina Sartipi and her team for welcoming me to the group, supporting my research and stay in Chattanooga and providing me with data from their testbed. I would also like to thank my managers in the SDN Lab at the Huawei Research Centre, Dublin, Dr Olga Havel, Dr Vincenzo Riccobene and Dr Adriana Hava, for giving me the opportunity to work on an exciting project with them as Research Intern.

I am very grateful to other colleagues at Walton, including Dr Daniel Martins, Dr Michael Taynnan Barros and Dr Sasitharan Balasubramaniam, for the research collaborations and fruitful discussions that helped me immensely. This research would have been impossible without the support of my wonderful colleagues at the RITS division, John Ronan, Derek O'Keeffe, Michael Kugel and Tibor Molnar, who made sure that the infrastructure was always running smoothly. Thank you for providing support through outages and the many transitions. I would also like to thank Jane Mahony, Nichola Courtney and Jonathan Mullowney for helping with all the logistics.

I would especially like to thank my wonderful colleagues who became my closest friends over the years. Dr Kriti Bhargava has been my number one supporter and cheerleader throughout my time at Walton Institute. I also owe her for all the time she spent discussing my research and her helpful advice at ungodly hours. I would also like to thank Sidhant Hasija, Dr Dixon Vimalajeewa, Dr Genaro Longoria and Dr Ruisong Han for ensuring I had downtime and felt loved and supported in Waterford. I would also like to thank Dr Mandy Hmar and Curtis Hall, who took great care of me in Waterford and Tennessee, where I was visiting for a project. I wish to also thank the newer batch of students who became my close friends including Samitha Somathilaka, Thakshila Wedage, and Naga Lakshmi Anipeddi. I am grateful to them for the enormous support and love they have offered me through my PhD research's many ups and downs.

I would especially like to thank my wonderful friends in Ireland and back home in India including Jean, Pavel, Neha, Naina, Paiker, Megha, Kanika and Medha, who would always ensure that I was taking care of myself, stayed up to date with my progress and always supported me when I needed them. I would especially like to thank my wonderful parents, Arun Sharma and Dr Rajni Sharma, who are the reason I could think beyond my limits and

aim for the sky. They have been there for me like a rock, taking care of everything else during my stint here in Ireland and making sure I was fully focused on my PhD work. I would also like to thank my sister Radhika and my brothers Aditya and Akash for all the fun times that I have had on my trips back home. Those times we spent together were a huge help in keeping me afloat through the challenging phases of my degree. I would also like to thank my wonderful uncles Ajay and Anil and aunts Lata and Kalpana, whom I am immensely lucky to have in my life. They have taught me to be patient and kind, no matter what life offers. I would also like to thank my wonderful cousins, who were my first set of friends and have always been a call away no matter how far we live. I would also like to thank my nephew and nieces for their selfless love. I will forever be grateful to my late grandparents, who taught me valuable life lessons that I keep very close to me. I have the most profound debt of gratitude to all four of them.

Last but certainly not least, I would like to thank my wonderful partner Chandan, who has been a pillar of strength throughout my course of study. I am so grateful to him for staying with me on zoom calls through the very last minute of my submission deadlines.

List of Publications

- "Graph-based Heuristic Solution for Placing Distributed Video Processing Applications on Moving Vehicle Clusters", K. Sharma, B. Butler and B. Jennings, IEEE Transactions on Network and Service Management, (2022).
- "Scaling and Placing Distributed Services on Vehicle Clusters in Urban Environments", K. Sharma, B. Butler and B. Jennings, IEEE Transactions on Services Computing, (2022).
- "CogITS: cognition-enabled network management for 5G V2X communication", M. T. Barros, G. Velez, H. Arregui, E. Loyo, K. Sharma, A. Mujika, and B. Jennings, IET Intelligent Transport Systems, 14, pp 182-189, (2020).
- "Optimizing the Placement of Data Collection Services on Vehicle Clusters", K. Sharma, B. Butler, B. Jennings, J. Kennedy and R. Loomba, 2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pp. 1800-1806, (2018)
- "Decentralised federated learning-based object detection scheme for vehicular fog computing applications", *In preparation*

Abstract

Vehicular fog computing (VFC) is an extension of cloud and mobile edge computing aimed to utilise the rich sensing and processing resources available in vehicles. The emergence of VFC was motivated by the need to reduce latency in delay-sensitive applications, which makes it infeasible to deploy applications on the cloud. Thus, there emerged a need to deploy data processing services close to the source of data generation to reduce response time and bandwidth usage in uploading collected data to the cloud. Most modern vehicles in the near future will be equipped with plentiful sensing and processing capacity to make sophisticated decisions related to autonomous driving. The aim of this study is to utilise the under-utilised resources on these vehicles to deploy data-intensive, over-the-top services on a group of closely-moving vehicles. With the emergence of data-driven applications developed in the process of urban informatization, there is a need to collect and process data in a resource-efficient manner.

In this work, we consider applications where vehicles gather and process data for surveillance purposes such as studying the interaction of users with roadside cafes and gas stations etc. or detecting vulnerable pedestrians to increase commuter safety. Our work introduces methods and techniques to use the historic mobility pattern of vehicles to address the challenge of dynamism and instability in the vehicular network. To determine the availability of these resources, a stochastic mobility model is utilised, to select nodes with similar mobility patterns. Then a distributed and flexible service model and a mobility-aware infrastructure model are designed that are compatible with an unstable, non-static network. These distributed services are scaled in real-time and placed as multiple instances on the selected vehicular cluster to make the services more robust. The service scaling and placement problem is modelled as a bi-objective, constrained optimisation problem with the objective of efficient resource utilisation. To place the service chains efficiently on the vehicular clusters, a community detection-based cluster selection scheme and graph-based service placement heuristics are introduced. The feasibility of the study is presented by demonstrating results from extensive simulations using different resource and mobility profiles of vehicle clusters. The results of the performance of our service scaling and placement scheme indicate that it performs better than competing schemes in terms of resource utilisation.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Fog Computing is an emerging paradigm to deploy existing and futuristic Internet of Things (IoT) applications closer to the edge of the network [Bonomi et al., 2012]. The fog computing platform introduces the distributed deployment of applications, as opposed to centralized cloud services, and processes most of the data close to the source of data generation instead of sending raw data to the backend servers. This reduces the usage of the expensive and limited network bandwidth at the core network and also promotes real-time interaction between end devices. Fog computing, as an extension of Cloud computing, is based on deciding where to deploy applications that have local context and have lower delay tolerance. These applications are orchestrated in an automated manner based on the demand and the availability of the heterogeneous Fog devices, to ingest the large amounts of data generated from end devices. A large amount of data that is relevant for historical analysis and requires more complex processing and persistent storage is sent to the cloud for further processing.

The need for distributed computing paradigms like mobile edge computing (MEC) [Li et al., 2019; Ning et al., 2019] and fog computing is generated from the end devices getting smarter and generating massive amounts of data that needs to be fused with other data sources. Most of this generated data from smartphones, static cameras, and wearables, is collected in isolation and is not fused with other sources of data to make inferences. For utilizing this generated data and adapting to the distributed computing paradigms, there is a need to introduce novel services and service deployment methodologies, that can be used to leverage the heterogeneous and distributed devices for data generation and for hosting services. In this work, we focus on vehicles as fog computing devices as they are rich in embedded sensors like cameras, and radars and have rich processing capacity with advanced onboard units. Vehicles spend a significant amount of time in urban traffic and their sensing and processing capability is largely under-utilized. There is a projection of growth for both semi-autonomous and autonomous vehicles to over 12–14 million over the next decade, which will lead to

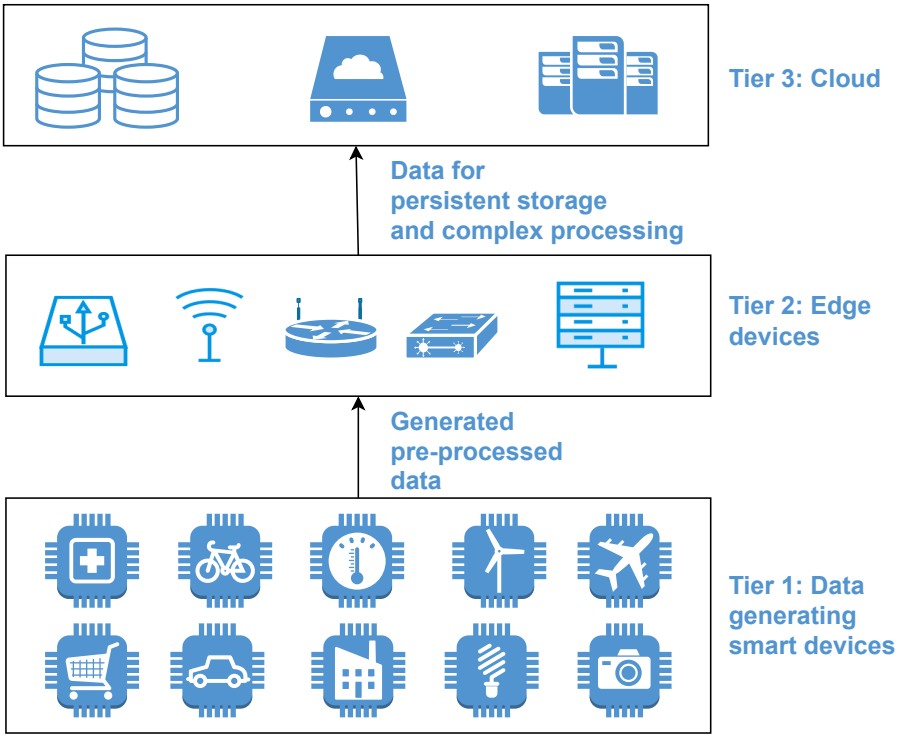


Fig. 1.1 A graphical representation of the three-tier Fog computing architecture.

the realization of connected vehicles (CVs) [Lienert, 2015; MEOLA, 2020]. The increasing deployment of applications for CV, related to safety, infotainment, or traffic management can be realized by exploiting the distributed fog computing resources.

This placement of services on either the end devices, the fog, or the cloud is a crucial problem to solve in fog computing research. There have been many existing works focusing on introducing the three-tier fog computing architecture [Aazam et al., 2018] and introducing service placement schemes aiming to efficiently utilise fog computing nodes for service deployment [Skarlat et al., 2017]. The three-tier vertical Fog computing architecture comprises multiple low-power sensing and computing IoT devices that collect raw sensor data [Sarkar and Misra, 2016] as the first layer. The second layer, also called the Fog computing layer comprises edge devices like edge routers, base stations, and home gateways. The devices in this layer have more processing capacity compared to the "thin" clients in the first layer and are also connected to the Cloud framework. The third layer of the architecture comprises powerful servers and data centres that can handle an enormous amount of data. A graphical representation of the three-tier fog computing architecture has been presented in Fig 1.1.

The architecture for the collaborative utilization of individual vehicular resources to carry out communication and computation is called vehicular fog computing (VFC) [Hou et al., 2016]. The computing capacity of each vehicle may not be enough to deploy compute-

intensive tasks, however, a group of vehicles can be used to collectively deploy sensing and processing applications. The initial work on vehicular cloud computing (VCC) considered vehicles as entities that generated demands for services or were used as sensing entities where the collected data was offloaded to the remote server [Abuelela and Olariu, 2010]. Eventually, the potential of vehicles themselves providing computational and networking resources for service deployment was explored. The use of vehicles as infrastructure was identified for three supporting reasons, firstly, vehicles are getting smarter with the emergence of autonomous vehicles that are rich in processing and sensing capacity. Secondly, the study of vehicular flow in busy segments of the city suggests that vehicular congestion is a major problem. Vehicles spend a significant amount of time either completely at a halt or moving slowly in high-density traffic. Thirdly, urban traffic has predictable macroscopic flow densities and peak traffic times. For example, a lot of vehicles commute towards the city centre or towards business parks on a weekday morning and travel back towards residential areas during evenings. Similarly, many vehicles commute towards shopping malls and recreational areas during weekends. These predictable vehicular flows can be used to initiate service-based clusters at busy intersections for data collection and processing, during peak traffic times. Thus, instead of focusing on parked vehicles or taxis and buses that have a very fixed trajectory, we focus on building flow models for private vehicles, whose mobility patterns are more challenging to identify.

In this dissertation, the focus is on horizontal scaling of services to not only utilize the ever-increasing computation and communication capacity in end devices but also to use the dash cameras on vehicles to collect data. Typically, service placement in data centre networks requires service placement plans for static and composite services. Such services are not suitable for a dynamic vehicular environment where vehicles will continue to join or leave the cluster leading to node and link failures. Individual vehicles might not have enough resources to deploy data collection and processing services, as they are computationally intensive. Such smart vehicles might also have their own computational requirements so they can run services like self-driving and navigation systems. These vehicles will lease their spare resources for over-the-top services only in return for an incentive. Thus, it is crucial to break down the service into smaller service components called ‘tasks’ in our work. Each task has different functionality and is dependent on data from another task to carry out its functionality. Thus, a service is modelled as a linear graph where nodes represent tasks and the edges represent the data dependency between these tasks. These tasks are further scaled to multiple task instances/replicas so that the processing in the service can be broken down further. This model increases the resilience of the service in case of node and link failures.

This service model also supports data collection from multiple vehicles. The data is sent to the data processing tasks on the vehicle clusters. Collecting data from multiple sources can improve the quality of the collected data. Our service model can be extended to collect multi-modal data from different sensors on vehicles. The processing tasks in our service model then clean the collected data such that redundant frames, especially with video data collected from multiple sources, can be separated from frames of interest. The processed information is then sent to the client as dedicated information or the roadside unit (RSU) for more complex processing and/or persistent storage.

A lot of existing work in VFC focuses on using vehicles for latency-sensitive applications. We focus on a novel use case of initiating an opportunistic vehicular cluster as a group of 'moving sensors' that can collect and process data for applications with different local contexts and scopes. Such applications include surveillance/monitoring, like the collection of data to study the interaction of commuters with gas stations, and cafes and collecting road traffic information to decrease traffic congestion. The data collected for such applications are pre-processed and then sent to edge or cloud services for more complex learning-based algorithm training. Other applications like detecting vulnerable cyclists and pedestrians require drivers to be alerted in real-time. Such applications have local and situational contexts and do not require the data to be collected at a back-end server. We specifically focus on video data in our service model as video streaming and object detection are the building blocks of many vehicular applications, including scene understanding, driving assistance etc. [Zhu et al., 2019]. Since video data requires more network bandwidth and higher processing capacity, it is more useful to process the video data close to the source of generation, in a distributed manner.

Hence, our service model can significantly reduce the need for installing additional infrastructure for data collection, which directly increases the CAPEX and OPEX for the data collection process. It will also reduce the need to have dedicated links that transfer data collected from static video cameras to servers using expensive network bandwidth. Our services reduce network usage by collecting most of the data close to the source of data generation. Such services will also help in ad-hoc data collection that can be initiated at any road segment which has enough vehicular density. The vehicles that agree to lease their data collection and processing capacity can be incentivised by road transport authorities by giving them free parking and discounts on toll booths. If such services are initiated by telecom companies, vehicle owners can be incentivised by giving extra credit in return for added infrastructure capacity provided to the network operators. Thus, a whole VFC marketplace can be created where these moving vehicles can provide additional capacity to the existing fog/edge and cloud infrastructure.

We try to resolve the complexity in vehicular networks due to the inherent mobility of vehicles that govern the availability of these vehicles as fog nodes. We study the traffic patterns of vehicles to identify the spatial and temporal distribution of slowly moving traffic. Furthermore, we focus on how these moving vehicles can be used as infrastructure, by intrinsically considering the mobility of vehicles as part of the service placement model. We also introduce a novel service model in which services can be scaled flexibly according to the availability of vehicular fog nodes and the demands of the application. We then focus on how to efficiently place these novel services on the distributed and dynamic vehicular network. The main goal of this thesis is to introduce this novel use case of using vehicles as an opportunistic sensing and computing infrastructure and introduce service placement schemes to tackle the complexity of the dynamic infrastructure. The scope of the dissertation, the research hypothesis and questions, and the contributions are elaborated below:

1.1 Research Hypothesis

We consider realistic mobility patterns of vehicles and build efficient node selection and service placement schemes to successfully implement distributed data collection and processing services on moving vehicle clusters. To make a robust system of IoV that can be used for data collection and service deployment, we formalise the following research hypothesis:

The coordination and effective management of computation and communication, in a group of closely moving vehicles, can result in efficient service deployment at the network edge, resulting in a systematic method to collect and process data without deploying additional infrastructure.

To find a solution to efficiently utilize resources in moving vehicles, we divide the research broadly into four research questions:

1.1.1 RQ 1. How feasible is it to use vehicle clusters for service execution? How to estimate the aggregate computation and communication capacity of these vehicle clusters?

Unlike traditional infrastructural networks that are stationary, in distributed and moving networks, the resources are available only for a certain amount of time. There has been very limited work undertaken on the total time when vehicle clusters stay together, especially as a potential resource pool for service deployment. We start by introducing a generalised flow model for intersections, to study the predictability of vehicular flow using multivariate linear regression, random forest and ARIMA models etc. We then study the aggregate

communication and computation capacity of these vehicle clusters. We use real highway data to understand the speed versus occupancy/density of vehicles on different days of the week at different road segments. The performance or stability of such clusters with respect to the number of well-connected nodes that make it to the end of service execution time has not been studied in the literature. Similarly, the relative mobility of vehicles, w.r.t. the vehicles moving around them has also not been studied. We thus introduce centrality score-based measures to evaluate the performance of selected clusters. To understand the feasibility of using vehicle clusters by leveraging their mobility data, we break RQ1. into the following issues:

- Are vehicle flows predictable? Can a generalised intersection-based flow model be used for predicting vehicular flows and initiating vehicular flows?
- What is the aggregate communication and computation capacity of such vehicle clusters?
- What is the relationship between vehicular speeds versus occupancy?

1.1.2 RQ 2. How can closely moving vehicles be used for service provisioning?

The second research question focuses on how to utilize the mobile, virtualized resource pool of vehicle clusters to deploy applications. Such a model requires the efficient management of resources, especially if those resources are to be leased to third-party service providers. The network states in a dynamic and virtualized network constantly change over time, requiring flexible service provisioning and orchestration. With virtualization and loose coupling, we can map services to resources in a shared infrastructure. However, it is not always feasible to place static and monolithic services as an individual vehicle might not have enough resources to perform complex analytics. For example, a scene understanding application that can help grasp the information of the target environment to make a reasonable judgement can be deployed on a vehicular cluster. The application will be broken down into several service components or tasks. A perception service would be required for detecting and tracking objects. Then data from multiple sensors will be fused at the fusion service to construct a more accurate representation of the environment in comparison to what a single sensor can achieve. The final task would be a decision layer that will include alerting based on the application requirement or behaviour planning based on the perception of the application. Such a task would require a lot of data and high computation capacity for processing the data. Thus, instead of deploying it as a monolithic service, it is broken down into several tasks to

leverage the distributed resources available in nearby vehicles. The dynamic network would also lead to link and node failures, thus, more resilience needs to be added to the service for successful deployment. This is achieved in our model by replicating a task to multiple task instances.

In the case of mobile nodes, the availability of resources is dependent on the location of each vehicle in space and time and its connectivity to a managing entity. In the special case of vehicular networks, the mobility is restrained to the topology and the traffic conditions of a particular road. The mobility of the vehicles makes the problem of service placement in the vehicular network more complex. Thus, mobility needs to be an intrinsic part of the model to find a resource-efficient service placement plan. We consider video collection and processing services as video data provides more context and information to the services. The analytics of video data requires more processing capacity in comparison to other sensor data, thus making it beneficial to break down services to multiple task instances. To address these dynamism issues in the vehicular network and leverage the distributed, heterogeneous and limited resources on each vehicle, we break down RQ2. into the following issues:

- What constraints are necessary for service cluster selection and effective service placement with the objective of efficient utilization of bandwidth resources and increasing the probability of successful service execution?
- What kind of service model is required for effective and adaptive service provisioning based on the resource and mobility state of the service cluster?
- What is the performance of the network, derived from a realistic emulation of the vehicular network and video collection services?

1.1.3 RQ 3. How to resolve the vehicle cluster selection, service scaling, and service placement problem using community detection and graph-based algorithms?

The second question covers the mathematical modelling of the IoV infrastructure and the distributed service model and studies the feasibility of service execution on moving vehicle clusters. This question aims to find solutions for the cluster nodes and control node (CN) selection techniques and introduces graph-based heuristics for service placement. As we deploy distributed applications on moving vehicles, we aim to select those vehicles that are more probable to stay together for the duration of data collection and processing. The cluster selection ensures that a sufficient number of vehicles stay as part of the cluster to

collect and process data over a period of time. The selection of CN and vehicular clusters is based on the historic mobility patterns of the vehicles. This mobility-based CN and vehicular cluster selection procedure will reduce the need for service reconfiguration as those vehicle nodes are selected that are more probable to stay together for a defined period of time. We introduce two specific applications related to vehicular crowdsensing and object detection for increasing commuter safety.

We aim to select stable opportunistic clusters managed by a local controller (the RSU or CN). The service placement model uses vehicular mobility models built by leveraging the predictability of vehicular flows at intersections and the historic mobility patterns of vehicles in urban regions. The model aims to select those nodes that are more likely to stay within the cluster till the service is executed fully. After CN and vehicle cluster selection, we need to introduce service scaling and placement methods such that the competing objectives of distributing the service widely and of minimizing bandwidth usage in placing these services are achieved. To solve the cluster selection and the service placement problem, RQ3. needs to address the following issues:

- How to select nodes for the service cluster such that the selected nodes stay together for a longer period of time?
- How to scale and place the services on the selected vehicular cluster effectively? How does the introduced scheme perform compared to existing service placement and node selection solutions?
- What examples of distributed applications can be deployed on moving vehicles?

1.1.4 RQ 4. Can a federated-learning-based scheme be introduced to deploy a distributed object detection service using video data from multiple video sources?

To implement the object-detection application introduced in our work, we introduce a distributed federated learning (FL) scheme. FL is a collaboratively decentralised technology, designed to overcome data silos and sensibility [Li et al., 2020a]. It is also a privacy-preserving mechanism where multiple clients coordinate with one or more centralised servers to share local updates with local data for the model to achieve the desired performance. The distributed FL model is a direct match to the service and infrastructure models introduced in our work. We use data from an intersection-based test-bed in Chattanooga, Tennessee, to implement a distributed and hierarchical federated learning scheme to implement an object

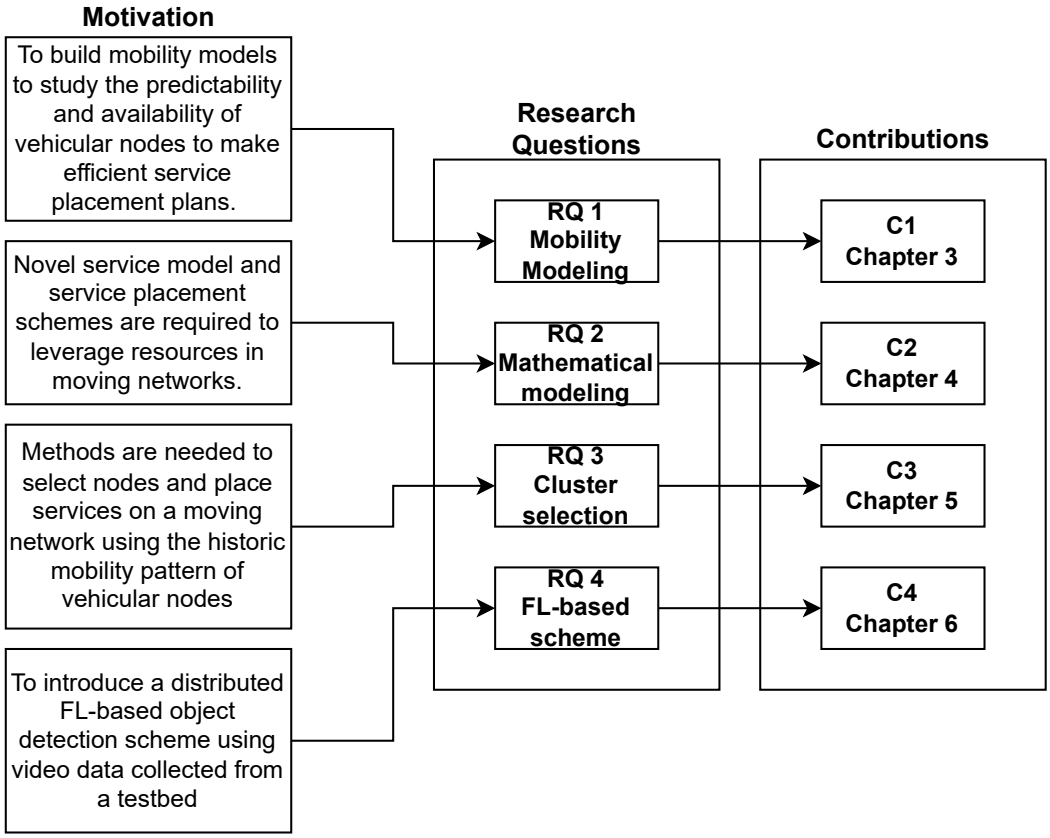


Fig. 1.2 An overview of the research plan and the mapping between the motivation, the research questions and the contributions.

detection application to detect pedestrians and cyclists and alert drivers in the region. To implement such a scheme and application and to address RQ4., we aim to focus on the following issues:

- How to use video data from a real road intersection-based testbed for implementing an object detection application?
- How does the centralized FL-scheme perform against a hierarchical and distributed scheme?

1.2 Research Contributions

The main contribution of this dissertation is the management and utilization of dynamic networks, leveraging the mobility patterns of vehicles, for deploying novel crowdsensing and commuter safety-related applications. We aim to find service placement plans that are

Table 1.1 Research contributions from the dissertation.

Research questions	*Research contributions*	*Chapter*	*Publication*
RQ1.	Introduced methods for mobility management for slow-moving vehicles in terms of: - predictability of vehicular flow. - resource capacity estimation for vehicle clusters. - density estimation for initiating vehicle clusters.	3	IEEE Trans. on Services Computing [Sharma et al., 2022b]
RQ2.	Introduced novel service models, an efficient service placement scheme for the dynamic networks.	3, 4	IEEE PIMRC IEEE Trans. on Services Computing [Sharma et al., 2018]
RQ3.	Introduced heuristics for node selection and service placement. Introduced two novel and distributed services related to video crowd-sourcing and pedestrian detection.	5	IEEE Trans. on Network and Service Management [Sharma et al., 2022a]
RQ4.	Introduced a federated learning scheme for object detection Compared the federated learning approach to existing schemes like YOLO and Faster-R-CNN.	6	Manuscript in preparation

probabilistically optimal, given the absence of information about individual vehicles. This is an example of decision-making under uncertainty. In cloud service placement, the service requests are uncertain (in scale, frequency, etc.) but the infrastructure is static with prior knowledge of available resources. In this work, the opposite applies, such that the availability of computational and communication resources is dependent on the mobility of vehicles. An overview of the motivation, the research questions and the contribution is illustrated in the Fig. 1.2. The contribution, listed as C1-C5, with respect to the RQs are also summarised in the research contribution table (1.1), and are detailed below:

- **RQ1:C1** To understand the feasibility of using vehicle clusters as infrastructure for data collection and data processing, we use real vehicle density data to introduce a

generalised vehicular flow model for intersections. We use the macroscopic vehicular density data to study the predictability of vehicular flow using techniques like multivariate linear regression, random forest, and ARIMA model for time series forecasting to study the predictability of vehicular flow. We show that vehicular flows can be predicted accurately using historical mobility data. We also study the speed versus density data of the available vehicles to identify service zones where the vehicular cluster can be initiated. We study the speed versus occupancy data and Level of Service (L0S) parameters to study the traffic flow description using vehicular density. We also compute the aggregate communication and computation capacity of these vehicle clusters.

- **RQ2:C2**: The main contribution of this work is to develop an infrastructure and service model to implement the novel use case of initiating opportunistic clusters and deploying distributed services on these clusters. We formulated the service placement problem as a detailed integer linear program (ILP), which gives an optimal service placement plan for the opportunistic vehicular cluster. The ILP solves the competing objectives of distributing the service widely, increasing the robustness of the service, and also minimising the bandwidth usage, which makes the problem challenging to solve. We also validated the service placement on a dynamic network by emulating the service using road mobility simulators like SUMO and network simulators like Mininet-WiFi. We evaluate the optimal service placement solution which results in a higher service completion rate and more efficient network utilisation. We also run extensive simulations to study the effect of stable and unstable vehicle clusters on processing and communication costs. We compare the introduced scheme to other placement schemes including cloud placement, edge/RSU placement, and mobile node placement without scaling the task to multiple instances.
- **RQ3:C3** We first leverage community detection-based algorithms to select vehicle clusters. The group of participating vehicles is modelled as undirected graphs, where the edge between any two vehicular nodes is weighted by the joint cluster cohesion probability (CCP). The CCP is the probability of two nodes staying together for a time period. To find an optimal placement plan for the services, we introduced a graph-based service placement plan that finds an optimal placement of service chains in-network, on the path from the video collection task instances (TIs) to the CN. The graph-based heuristic promotes that most of the TIs are placed on the vehicular cluster, to promote the horizontal scaling of resources. We compare the introduced heuristics to the ILP and other competing schemes in the literature.

- **RQ4:C4** We use an FL scheme to deploy an object detection application for detecting pedestrians and cyclists. We use real video data from static cameras installed at an intersection-based testbed in Chattanooga, Tennessee. We study the application's performance, based on the detection accuracy.

1.3 Dissertation Organisation

The thesis is organized as follows. This chapter introduces the research and highlights the research hypothesis and research questions. The contributions with respect to each research question have also been highlighted.

Chapter 2, presents a detailed background of the evolution of vehicular networks from VANETs to IoV systems today. The chapter also presents the relevant literature review of VFC architecture, different VFC use cases, and task scheduling and offloading schemes used in fog computing systems. We then summarise the literature review by identifying gaps in VFC research and map each gap to the research questions that address the gap.

The following four chapters address the four research questions described in this chapter in Section 1.1. In chapter 3, the first research question (RQ1) is addressed. The chapter first introduces the intersection model for predicting vehicular flows at intersections. We study flow predictability using MVLR, random forest and ARIMA models. We introduce the aggregate communication and computation capacity estimates for the vehicle clusters. We then study the correlation between vehicular density/occupancy versus the speed of vehicular flow, to see how closely spaced slow-moving vehicle clusters are. The chapter then introduces the system model, network topology and service model for the service placement problem introduced in this work.

Chapter 4, addresses RQ2 where we provide the detailed mathematical formulation for the mobility-aware infrastructure model, the distributed service model, and the service placement problem modelled as an ILP. The chapter also provides detailed simulation results for validation of the model. In chapter 5, the third research question (RQ3) is addressed. It introduces the community-detection-based vehicle cluster and CN selection schemes. The chapter also introduces the graph-based service placement scheme to promote in-network processing of the collected video data on an opportunistic cluster. The introduced schemes are compared to the competing schemes in the literature.

In chapter 6, we introduce an FL-based scheme to deploy an object detection application for detecting pedestrians and cyclists using data from an intersection-based testbed. This chapter addresses RQ4. Chapter 7, presents the summary of the research work carried out

in this dissertation. The chapter also highlights the short-term and long-term future works planned for this research.

Chapter 2

Background

This chapter presents an in-depth background of technologies like vehicular computing and fog computing that are crucial to understand the need for using vehicles as infrastructure. The technological advancement in distributed computing has driven the concept of using moving networks with distributed resources for service deployment. Section 2.1 explores the transition of vehicular networks from being used for safety-related information dissemination networks to connected vehicles today that are envisioned to use their huge computational power by aggregating the resources of individual vehicles and RSUs. Section 2.2 presents the VFC architecture and also gives details of applications with different update cycles, classifying them into (i) crowdsensing, (ii) processing-intensive and (iii) autonomous driving type applications.

We then present the suitability of placing distributed services on the VFC ecosystem by first highlighting the urban mobility scenarios and then introduce the key challenges in utilising moving vehicular nodes for application deployment in Section 2.3. We then present different mobility models considered for mobility prediction in Section 2.3.3, flexible programmability in Section 2.3.4 and the communication management for the distributed service deployment in Section 2.3.6.

We then give a detailed outlook on the existing literature in the field of fog computing and vehicular computing paradigms. We have identified gaps in the literature, such as limited study in utilising moving vehicles as a means of collecting and processing data and the lack of novel service models and service placement plans to leverage the resources in moving networks.

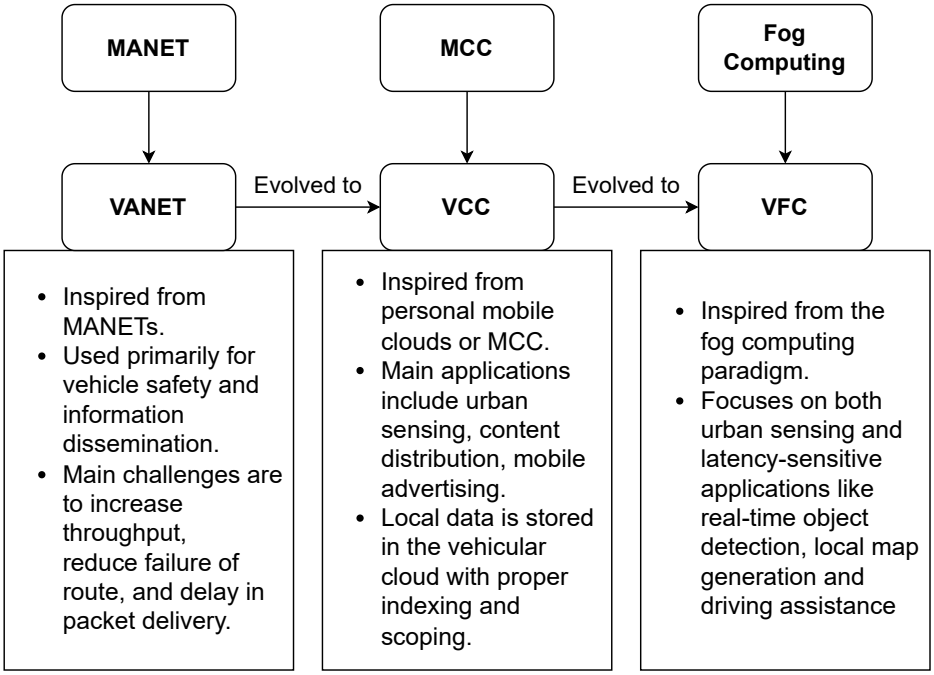


Fig. 2.1 The evolution of VANETs to VFC. The figure also highlights the mobile networks that inspired the respective vehicular networks.

2.1 Evolution of VANET to the Internet of Vehicle Ecosystem

Vehicular adhoc networks (VANETs) have been studied extensively over the past decade from a communication point of view, to effectively disseminate safety-related information to neighbouring vehicles and transport authorities [Ucar et al., 2016]. These vehicles focused on timely dissemination of safety messages to update nearby drivers about accidents, traffic jams, and the condition of roads. More specifically the applications included safety recall notice, wrong way driver warning and alerting the approach of emergency vehicles etc. [Report, 2009]. The main focus of the research undertaken in VANETs was achieving mobility and connectivity management while ensuring high data packet delivery ratio and low latency. The research also focused on efficiently utilising fourth-generation and Long Term Evolution (LTE) along with IEEE 802.11p which is the standard for wireless access for vehicular networks. There has also been active research in the application of Software-Defined Networks (SDN) to manage VANETs and to control the heterogeneous demands of different applications in a scalable and flexible manner [Al-Heety et al., 2020].

VANETs were also visualised for value-added services like parking navigation, automated toll payment, location-based services such as finding the closest washrooms, restaurants,

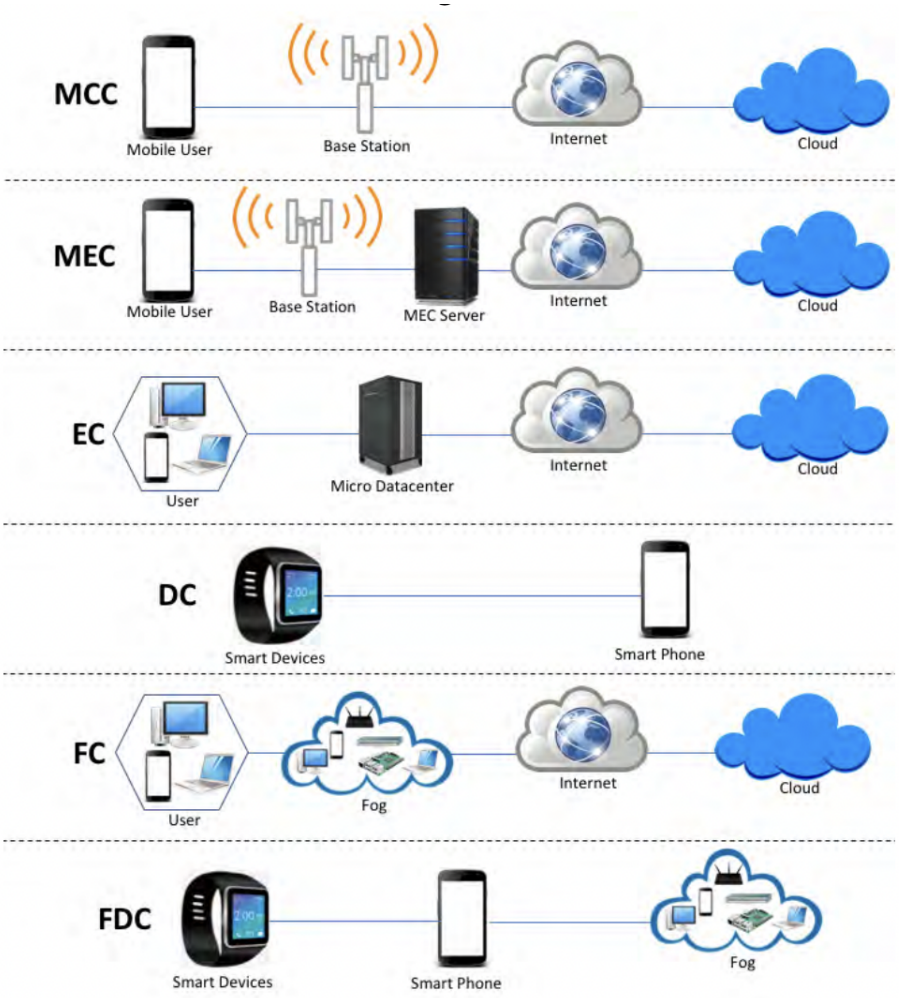


Fig. 2.2 High level architecture of Mobile Cloud Computing (MCC), Mobile Edge Computing (MEC), Edge Computing (EC), Dew Computing (DC), Fog Computing (FC) and Fog Dew Computing (FDC) from Naha et al. [2018].

gas stations etc. Many of these services are now provided as part of navigation maps like google maps. VANETs were also aimed to utilise for infotainment type applications, for example, to provide internet to nearby vehicles. However the role of each vehicle was to be either the sender, the receiver or the router of messages to nearby vehicles. Vehicles were not yet visualised as a means of hosting processing services. There was a lot of relevant research undertaken for configuring inter-vehicle or V2V and vehicle-to-everything (V2X) communication using multi-hop broadcast/multicast or unicast to transmit traffic-related information over the network [Benkerdagh and Duvallet, 2019],[Lin et al., 2017]. This research on adequate communication protocols was crucial to build the foundation for utilising vehicular networks as a means to deploy applications that can benefit from wireless communication between vehicles.

The advances in computation and communication technologies have led to the advent of automated vehicles with embedded sensors and onboard computers that can be used to fuse large amounts of sensor data to make decisions in real-time. This has further led to vehicles being visualised as a potential source of data generation as well as data processing for analysing and disseminating information. The benefits of the close coupling between the data source and data processing agents are (i) reducing the latency between users and processing device, specifically for real-time interactive applications and decision making, (ii) reducing network utilisation by not offloading all the generated data to the remote cloud and (iii) improving Quality of Service (QoS) and Quality of Everything (QoE) by flexibly scaling services in real-time, based on the resource availability, mobility patterns of vehicles, and the application requirements.

The IoT trend started the movement of application deployment away from clouds to *things* on the edge of the network. In the newer computational paradigms like Edge and Fog computing, the application logic and analytics are placed on devices close to the source of data generation. The study of such technological paradigms that deal with heterogeneous devices being used to deploy services coupled with the evolution of vehicles as intelligent devices, has led to the application of fog computing on vehicle-based settings.

Even after decades of research on VANETs that focused on improving the intermittent connectivity and decreasing the delay for safety-related applications, there have not been many implementations of such networks in real-life settings. The next technological advancement in vehicular network research was the concept of VCC which was inspired by the use of Mobile Cloud Computing (MCC). MCC is a platform where individual mobile devices can be both cloud users or cloud service providers [Khan et al., 2014]. The research on MCC included adapting applications to cloud computing application models and introducing an MCC architecture for application offloading. Most of these devices were seen as hosts

for communication whereas all the processing was offloaded to the cloud. VCC included the use of a group of vehicles for enabling safe driving, urban sensing, content distribution and parking navigation. For the first time, [Gerla, 2012] suggested that VCC can carry out significant computing on the vehicular cloud. The vehicles in a VCC keep local and contextual information instead of directly uploading it all to the cloud. The VCC also has search features for the content stored in the vehicular cloud. In the different VCC architectures introduced in the literature, vehicular clouds have some processing autonomy however, they had to communicate with remote data centres even if intermittently [Gu et al., 2013].

MCC had some drawbacks including high latency, low coverage and lagged data transmission. MCC also falls short in scenarios that require real-time decision-making. The implementation of MCC, along with the exponential growth in connected devices on the internet further makes the usage of MCC in highly dynamic environments infeasible. The newer era of computing paradigms included Mobile Edge Computing (MEC) [Abbas et al., 2018] and Fog computing tries to address the challenges faced by MCC by deploying the cloud resources, e.g., processing and storage capacity at the edge of the network. This extends the capacity of cloud computing to the edge of the network by providing high processing capacity, high bandwidth and ultralow latency.

VFC is the concept of using vehicles as both computation and communication resources and are considered as Fog nodes themselves [Hou et al., 2016]. VFC is motivated by MEC and fog computing paradigms[Keshari et al., 2022]. Even though there is no one widely accepted definition of VFC, the role of vehicles in VFC system is of data generation due to the availability of computing and sensing devices like cameras, radars and GPS and of data processing due to the availability of on-board units with computational and communication capacity. Some of this collected data can be processed on the vehicle itself, in order to make real-time decisions for delay intolerant applications [Huang et al., 2017]. The evolution of VFC from VANETs has been described in Fig. 2.1. The figure also reprsents the mobile networks that inspired to respective vehicular networks.

Fig. 2.2 represents similar computing paradigms besides Fog computing including MCC, MEC, Edge Computing (EC), Dew Computing (DC) and Fog Dew Computing (FDC). EC is defined as a computing paradigm where edge devices provide computing facilities without spontaneously associating with any cloud-based services. DC is a microservice-based concept that does not require any management or coordination from centralised server. The dew server is accessible to devices in the same network without the need for Internet. The FDC paradigm provides offline services to IoT devices by connecting to a community server. The community server interacts with the Cloud, thus providing services in offline conditions.

2.2 Vehicular Fog Computing Architecture

A high-level VFC architecture is composed of three layers, similar to the fog computing architecture which seamlessly integrates network edge devices and cloud servers [Yi et al., 2015]. FC is a geographically distributed computing architecture that ubiquitously connects heterogeneous devices at the edge of the network to collaboratively provide elastic computation, communication and storage services [Hu et al., 2017]. Fog computing provides different instances of fog resources to the underlying networks like wireless sensor networks (WSNs), virtual sensor networks (VSNs) and VANETs. We describe the three layers of the VFC architecture below and present it in Fig. 2.4.

1. **Vehicular layer**: The first layer of the VFC architecture comprises smart vehicles. These vehicles are rich in embedded sensors like built-in cameras, radar and GPS. For example, a Tesla car has 8 cameras. These vehicles are used to collect data as they are moving through traffic. A smart vehicle is estimated to generate approximately 25 Gb/s in a single day: 20-60 Mb/s from cameras, 10 Kb/s from radar and 50 Kb/s from GPS [Huang et al., 2017]. Due to the large amount of data generated, especially video data, it is expensive to upload all the generated data to edge servers for permanent storage and processing. These smart vehicles are also rich in computation, communication and storage resources. In our work, we consider that the generated data can be processed not just at the vehicle generating data but also at nearby vehicles that move closer to the data-generating vehicles. Our work focuses on scaling and placing services on the group of closely moving vehicle nodes.

2. **RSU/edge server layer**: The data generated and processed at the group of vehicles is then sent to the stationary RSU and edge servers which form the second layer of the VFC architecture. These edge servers can be deployed in different parts of the city to support VFC applications. The RSU manages the state of the vehicles that execute services and also manages the state of the service deployed on the vehicular cluster. The RSU then implements more complex processing tasks if needed and reported them to the cloud servers. The RSU can also be used for persistent storage for local requests related to video streaming, parking navigation and smart traffic light management. The data collected, processed and stored at the RSU has more scope and relevance in space and time.

3. **Cloud layer**: The cloud layer is composed of a traffic management server [Ning et al., 2019]. The cloud server is used for city-wide monitoring and stores data that has information relevant for a long period of time. Only the data processed at the vehicular

layer and the RSU layer is sent to the Cloud such that redundant raw data with only local scope and context is not sent to the remote servers. More complex analytics including training large data sets for object detection or classification is also executed at the cloud server.

2.2.1 Applications with different level of processing on the vehicular fog

Vehicular applications range from safety-related applications like lane changing and accident alerts to over-the-top services for infotainment and to make the commute more comfortable through parking recommendations etc. Such applications have different processing requirements and delay tolerance. Vehicular applications are also classified based on the update cycle of the applications[Zhu et al., 2018]. With the availability of data generated from vehicles, we focus on data-centric applications and classify them on the basis of the proportion of processing carried out at the vehicular fog.

- **Crowdsensing application**: Application based on crowdsensing use vehicles as moving sensors for sensor data collection. Applications of this category include measuring the traffic density at an intersection in real-time, or surveying road conditions for road traffic monitoring. 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. Thus, the data is reduced to 70-80% of its original size before being sent to the cloud for executing compute-intensive tasks, possibly applying complex machine learning to the data.

- **Processing-intensive applications**: Applications in this category have local context and have real-time processing requirements. One example of such an 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. Such applications can be considered compute-intensive and require higher processing capacity and so large amounts of unnecessary data should not be uploaded to the cloud before processing.

  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 20% of its size and only relevant information is sent to the client or the RSU for broadcasting the information. Another example of a processing-intensive application is the detection of vulnerable pedestrians and cyclists to alert the drivers in the region and increase the overall safety of the commuters. Such applications will have even smaller delay-tolerance in comparison to applications related to studying the traffic conditions, road conditions and usage patterns of restaurants and cafes.

- **Autonomous driving assistance**: A large plethora of vehicular applications are now based on driving assistance or autonomous vehicles. Autonomous vehicles generate 3D maps to sense and navigate the environment [Zhu et al., 2018]. Vehicles with embedded cameras can be used to contribute to generating 3D maps from multiple sources to generated a high definition and accurate map. Such applications are also very latency-sensitive and the generated data has very limited scope in terms of locality and time period.

2.3 Suitability of Service Placement on a Vehicular-Fog Ecosystem

From a management perspective, we argue that the Intelligent Transport System (ITS) is not a homogeneous domain. Indeed, there are two different regimes, each with differing management challenges. Even though the scenarios and the services might appear to be the same, the differences in resource availability and mobility behaviour are such that a single deployment and management strategy is not advisable.

The first of these regimes are concerned with slow-moving, “urban” traffic, with closely-spaced vehicles and relatively easy access to supporting roadside networking infrastructure. The vehicles on the network edge are mobile, but the effort required to identify and maintain connectivity with the nearest network gateway (for routing purposes) is not significantly more than what would be needed with pedestrians with terminal devices. Some applications relevant in an urban scenario are:

From a management perspective, we argue that the ITS is not a homogeneous domain. Indeed, there are two different regimes, each with differing management challenges. Even though the scenarios and the services might appear to be the same, the differences in resource availability and mobility behaviour are such that a single deployment and management strategy is not advisable.

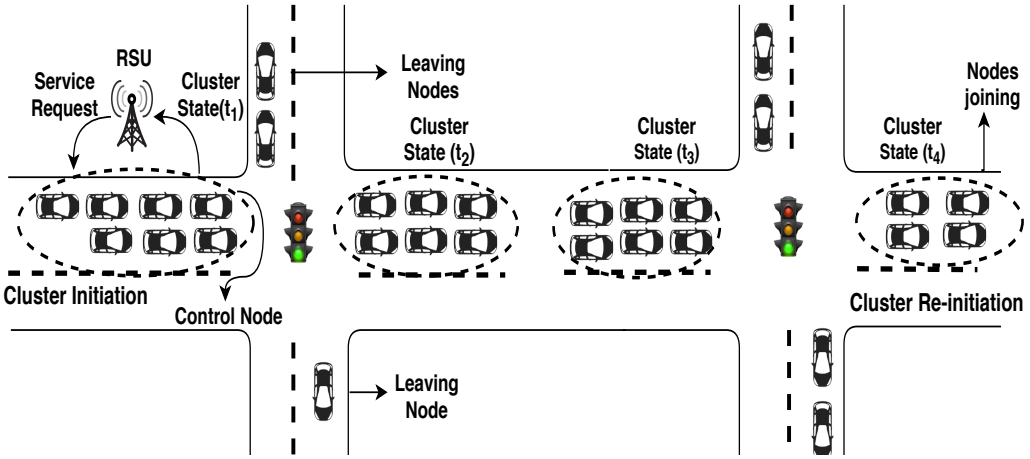


Fig. 2.3 Vehicle Clusters form, but membership changes over time. Clusters accept service chain placement requests from RSUs and perform service chain scaling and placement.

The first of these regimes are concerned with slow-moving, "urban" traffic, with closely-spaced vehicles and relatively easy access to supporting roadside networking infrastructure. The vehicles on the network edge are mobile, but the effort required to identify and maintain connectivity with the nearest network gateway (for routing purposes) is not significantly more than what would be needed with pedestrians with terminal devices.

2.3.1 Urban Scenarios:

In this section, we present some data-centric applications relevant in an urban scenario. These application have different processing and delay requirements. We aim to deploy such application on slow-moving vehicles in urban sections of the cities.

- **Vehicles as moving sensors (video crowdsourcing)**: Monitoring and surveying is an expensive task, critical for data collection and aggregation in *smart cities*. Vehicles in urban scenarios spend a lot of time in slow-moving traffic. This time can be utilized to build opportunistic clusters that collaborate to collect and process surveillance data. This information can be application-specific, like collecting the density of greenhouse gases, gathering information on the popularity of vehicle models, reporting accidents, and other emergencies for safety measures. The processing components can be mapped to the network based on the scope of generated data: (i) Example: Pedestrian detection at a local coffee shop to gauge its popularity. Such applications can be processed at the Fog nodes as the data has limited spatial scope (ii) Vehicles can also stream data, which is pre-processed at the network edge, but is sent to the core network for further application-specific processing. For example, insurance companies may provide an

incentive for providing video streams related to erratic driving behaviour or accidents. Such data can be used by multiple applications and has a wider scope.

- **Assisted driving**: Applications for situational awareness, lane-changing, and accident prediction have a very small delay deadline. Most autonomous vehicles navigate using a high-definition 3D map of real-time vehicle movement and the location of nearby entities [Zhu et al., 2018]. This application requires a lot of videos generated in a short validity period. This data is then trained in real-time for accurate situational awareness of the road users. Such applications are compute-intensive and delay-sensitive. These time-sensitive applications cannot be entirely deployed at the remote cloud and require newer service placement techniques to utilise the computing capacity at the fog/edge.

- **Over-the-top services**: Passengers in vehicles can be provided services like *mobile gaming*, where the terminal devices get streamed audio and video of the game, but the game is processed at the edge of the network. The vehicles have enough computation to act as thin clients, whereas it becomes easier for gaming companies to upgrade and develop games on a uniform platform at the edge of the network.

The second regime of ITS is quite different. Here the vehicular traffic is moving quickly, typically on the open road. Vehicles are not as close to each other for safety reasons. There may be roadside infrastructure, particularly cell towers, but it is probably not as dense as it would be in an urban setting. The rapid movement of the vehicles poses mobility problems because vehicles move in and out of a coverage area in much shorter timescales. Therefore the network state has a much higher rate of change than would be the case for pedestrians with hand-held devices.

2.3.2 Distributed service placement on mobile nodes

To facilitate the use of the dynamic IoV ecosystem for deploying services, we require the study of the novel and distributed applications that can adapt to the challenges of a dynamic network. It also requires the study of mobility patterns of vehicles to understand the availability of slow-moving or dense vehicular flows that can be used for service deployment. We look at the following key challenges involved in the application deployment on the IoV:

- **Service Offloading**: The main goal of task offloading in fog computing is to effectively distribute applications over the fog nodes to improve the QoS by reducing the task response time and improving network bandwidth and computation resource utilization [Hussein and Mousa, 2020]. The task offloading in backend clouds can lead to network

congestion at the backend and results in higher delay due to relatively higher network latency. The increase in IoT devices will result in large amounts of generated data that requires data filtering instead of uploading redundant data to the cloud, which will lead to increased overhead at the backend network. Thus, fog computing provides computation offloading closer to the network edge, resulting in lesser data transfer time and reducing the amount of network transmission. To optimally place tasks in the fog computing paradigm, the following aspects are considered [Misra and Saha, 2019]:

- The optimal location for task offloading needs to be discovered. Typically the offloading decision is between fog or remote nodes.
- The optimal fog nodes needs to be selected if tasks are offloaded to the fog layer. This will be covered in the next sub-section.
- In some cases, the optimal path selection is also considered while offloading tasks or the focus of task placement is to find optimal computation nodes on which services can be mapped.

Many existing works in Fog computing study the most resource-efficient and low-latency location for deploying applications, either on the device, the network edge or in the Cloud. However, the sensing nodes or the devices and the fog nodes are decoupled on two different layers. Another key challenge in utilising the IoV ecosystem for application deployment is to use distributed applications instead of traditional static applications due to the dynamism in the moving network leading to node and link failures. The task offloading strategies in VFC have additional complications due to information asymmetry and uncertainty [Zhou et al., 2020]. The availability of computation and communication resources varies in the VFC due to the mobility of vehicles. The global state of the vehicular network is not known as opposed to the case of resource allocation in static data centre networks. As the number of vehicles and the number of tasks to be deployed increases, the complexity of the exhaustive search to match tasks to the vehicles also increases.

- **Node Selection and Service Mapping**: The heterogeneous fog nodes have different capabilities, and this needs to be advertised through the resource estimation step. To filter and select the candidate nodes in the system, the nodes are ranked based on their current computation resources and network conditions to determine the most appropriate node for service deployment [Phan et al., 2021]. In the case of distributed services, we need to identify multiple fog nodes in a vehicle cluster, with similar mobility patterns, which can result in higher reliability in service execution. Thus,

identifying close moving clusters and then mapping services on these clusters is a crucial step in service deployment in the dynamic IoV ecosystem. As the Fog services in this study are broken down into smaller, co-dependent instances, the selected vehicle clusters need to be stable and move closely for the period of service execution. The multi-component instances also need both link and node mapping, which is sometimes solved by only considering link or node mapping, in other cases, links and nodes are mapped one after the other or mapped jointly, based on the complexity of the problem.

- **Efficient Service Placement**: The service placement problem borrows from the VM-bin placement problem which is used to model static resource allocation in the cloud computing paradigm. The optimal placement plan is to place the VMs to a minimum number of physical machines such that the vector sum of any physical machine does not exceed its resource limit [Shi et al., 2013]. In the case of service placement in VFC, the placement problem becomes more complex owing to the availability of resources and its dependence on vehicular mobility. Unlike cloud servers that are rich in processing capacity, each vehicle has limited processing capacity and hence services are broken down further, thus, the placement scheme needs to accommodate for the distributed services. Once we have selected nodes based on their resource availability and mobility patterns in the case of VFC and scaled services according to the resource requirement of the service, we require an efficient service placement plan, such that the service placement objectives are met. Different services, with different update cycles, have varying goals that define a successful service placement. For example, services on resource-constrained sensors or thin clients would aim to optimise the energy usage without degrading the performance of the application significantly. Similarly, safety-related applications in the vehicular scenario related to driving assistance like lane changing applications would have very strict latency applications. Some novel applications like using VFC nodes for crowdsensing and processing the collected data would require efficient resource utilisation for the monetary feasibility of the service deployment.

  More resource provisioning approaches need to be introduced due to the dynamism in the network and the heterogeneity of both the applications and the available resources on fog nodes. The service placement problem can also be optimized for objectives that are in accordance with the novelty in the VFC scenario. Due to the dynamism in the network, the completion of the task should be an objective for the placement. Similarly, for crowdsensing applications, the quality of collected data is a measure of the QoS and should be maximised for effective service placement. The lifecycle of the selected vehicle nodes is another way to identify how good was the selection of

the group of fog nodes that were assigned for service deployment. Similarly, more objectives can be introduced with respect to vehicles that provide services, in terms of benefit maximisation, that will encourage drivers to lease their resources [Tang et al., 2021].

- **Flexible Service Models**: The QoS metrics like service latency and bandwidth requirements are key factors in choosing a node to execute a service. A selected node may fail or lose connection to other nodes due to the dynamism in the network. Also, one vehicle node may not have enough resources to execute a video data processing task. Thus, instead of static service templates, it is crucial to deploy services that can be scaled in real-time according to the amount of data collected through the application. It also increases the robustness of the service by having multiple service instances as a fail-safe in case nodes hosting an instance of the task fails.

  For the online management of services, the real-time QoS feedback can help in migrating services for load balancing or in case of a node failure, to improve the performance of the application. The adaptive deployment of services is a key feature of Fog Computing wherein services can be scaled up or down based on the application requirement and the real-time feedback to improve the QoS/QoE. This requires efficient infrastructure estimation, a flexible service model and scheduling policies to support flexible application deployment.

- **Service Discoverability**: One of the Service-Oriented architecture principles is the need for the services to be available and accessible for discovery. As each *thing*/device has different functionality, services must be used efficiently and securely with the minimum human intervention. This requires means to describe the functionality and availability of sensors and virtual camera instances. Service discoverability also requires communication protocols to manage existing *things* in the network, as well as to adapt to the changing network dynamics. The dynamic scaling of instances requires services to be aware of the number of active instances of another service, to make the decision of task offloading.

- **Advanced QoS/QoE monitoring metrics**: Due to the variability in-network and the need to estimate the available resources and application requirement, newer QoS metrics need to be developed. These metrics should account for real-time resource estimation, mobility awareness, and flexible service scaling. Some metrics need to be application performance-centric and others benefit the service provider, ie. in terms of network and resource efficiency. Another inter-dependency is the involvement of users who are willing to subscribe to these services as well as those willing to lease resources

for the service execution. Similarly, QoE metrics like service availability rate, service access rate, expected service processing time, etc. need to be considered. However, it is also notable that predicting QoS and QoE in real-time is not straightforward due to the huge number of real-time interactions. Thus better QoS and QoE prediction schemes need to be developed.

- **Service Migration**: The mobility of nodes and unstable wireless links require service instances to be migrated in case a node fails or leaves the vehicle cluster. This service migration is different from the service migration between geographically distant data centers or between base stations due to vehicle mobility. This service migration is related to the service instances placed on vehicles and migrating them in case the service becomes unavailable. The migration decision needs to be made in real-time with minimum overhead due to the limited resources at the edge of the network. The process of migration also requires that the latency and downtime for accessing the service instance is minimum during the process. If the migration is related to the mobility of the node, the site of migration needs to be decided according to the mobility prediction of surrounding vehicles, to keep the service instance close to the user, especially considering the size of the service instance. The migration can also be performed to improve the QoS, as discussed in the QoS/QoE monitoring topic. In the special case of vehicle clusters, service management instances can also be migrated to RSUs ahead of time, based on the predictable mobility of a group of vehicles.

2.3.3 Mobility models for mobility prediction of vehicles

The MCC paradigm focuses on user mobility which leads to variable demands in the application at the Fog or Cloudlet level. In the Fog computing paradigm, the end devices are also potential application hosts, which makes resource management and task placement a more complex problem. In the case of the vehicular network, the mobility of vehicles can be predicted based on the historical mobility pattern of vehicles. Vehicular flow follows predictable mobility patterns through different days of the week and during peak and off-peak traffic hours. Such time periods and zones need to be identified to initiate vehicle clusters that can be used for service deployment. At the microscopic level, the use of navigation systems in smart cars can also help in predicting the location of a vehicle in the next time step. The use of mobility prediction can increase the reliability when services are placed on moving Fog nodes. The increased reliability can decrease the overhead caused by re-configuring services due to node or link failure related to the mobility of end devices. The urban and

highway scenarios and the differences in mobility patterns have already been detailed in Section. We now classify vehicle mobility models into macroscopic and microscopic models.

- *Macroscopic Mobility models*: These mobility models focus on the aggregated behaviour of vehicles and focuses on the vehicle dynamics including vehicular flows or counts, the average speed of vehicles, traffic density etc. Such information is available at many open source datasets mainly collected by transport authorities of different cities or made available through uber movement data [1]. Such vehicular parameters are enough to identify zones or segments in the city where such vehicle clusters can be initiated. The vehicular flows and average speeds also help in identifying the speeds at which vehicles navigate a particular road segment during a particular time of the day. This study can help in detecting slow-moving, high density and sometimes jammed traffic. Our study focuses on closely-spaced vehicles that can be used for application deployment.

  Even though vehicles in highways and freeways have more predictable trajectories, due to the topology of long highways, the vehicles move at very high speeds. The management of a vehicular cluster at higher speeds is more complex. There would be higher management overhead due to the faster dynamics in the vehicle cluster. We have focused on identifying busier freeways with slow-moving traffic that can be leveraged for service deployment. There is also a gap in the literature in studying the predictability of traffic flow and the density of traffic at busy sections of both urban intersections and freeways.

- *Microscopic Mobility Model*: This mobility model is based on the individual behaviour of each vehicle and its interaction with surrounding vehicles. Even though there are some open data for vehicular trajectories, many of the datasets are taxi or buses traces. It is useful to leverage the more predictable mobility patterns of taxis and buses, however, our work is based on utilising a general smart vehicle on the road. Most of the microscopic mobility models are studied through simulation generated from road traffic simulators like SUMO and network simulators like NS3, Omnetpp and Mininet.

  At urban intersections, vehicles are slow-moving and are probable to take one of the trajectories, constrained by the topology of road segments. In our work, we suggest that certain vehicles will have a fixed trajectory over the week, say from a residential area to a business park. This mobility behaviour can be used to select a candidate over another candidate. Many people may have a similar driving trajectory, which can further assist in selecting a group of vehicles for task allocation and execution. Such

[1] https://movement.uber.com/?lang=en-US

traffic patterns can be used to assign tasks that need data gathering for surveying traffic conditions on a certain segment of the road or to check the popularity of a coffee shop or supermarket etc. over a period of time. Such vehicles or groups of vehicles can be subscribed to for delivering this survey over a longer period. Such applications can be used as an alternative to deploying additional infrastructure for surveying tasks.

2.3.4 Flexible Programmability:

A programming framework is required for programmers to write device-independent code for flexible deployment. Even though we do not focus on flexible programmability in this work, it is an important aspect of distributed service provisioning. The code can be compiled by retrieving the device capability and services can then be deployed based on device functionality. The services can also be mapped to devices based on the device functionality, ie. sensing, communication, or compute-intensive task is mapped to the node with specific functionality. We highlight the key features of flexible programmability:

- The distributed services also require communication APIs to manage an asynchronous application deployed on different Fog nodes. The components also need to synchronise with components on the other network topology.
- The orchestration or coordination of different application instances needs to be automated. Certain applications have directed flow, which requires processing the flow according to the flow order specified by the service template. This requires the management of the application execution sequence.
- The scalability of applications is required to cater to the increasing number of IoT devices and application users at the network edge. As most of these applications are distributed, the application instance deployment also needs to be flexible and scalable, to increase or decrease service instances based on dynamic workloads

2.3.5 Vehicular radio access technologies (V-RATs)

Vehicular communication and access technology have been researched extensively for more than a decade now. The advancement in vehicular technologies has led to the enablement of applications for safety, infotainment and over-the-top services. The communication entities in a vehicular network can include any device connected to a network like vehicles, RSUs, fog nodes, base stations, pedestrians with hand-held devices etc. The information exchanged between vehicles and any other communication entity is called vehicle-to-everything or (V2X)

communication. V2X type communication includes both V2V and V2I communication modes.

The leading RAT for V2X communication is cellular networks and DSRC [Abboud et al., 2016]. Many commercial vehicles are now equipped with 3G or 4G/LTE-powered wireless access. We give some details about RATs for connected and autonomous vehicles below:

- **DSRC**: The main motivation for DSRC deployment is to enable traffic safety applications and improve traffic efficiency. The DSRC spectrum of 75 MHz is divided into seven channels including a control channel (CCH) and the other six service channels (SCH). The CCH is the highest priority channel reserved for safety-related applications whereas SCH can be used for both safety and non-safety-related applications. The DSRC supports a bandwidth of 3-27 Mbps per channel and allows two SCHs to combine to form 20MHz channel width to provide a higher data rate of 54 Mbps [Singh et al., 2019]. To utilise the DSRC spectrum in the vehicular network, the IEEE 802.11p and Wireless Access in Vehicular Environments (WAVE) protocol stack have been developed. The IEEE 802.11p is developed by adding V2V and V2I support in the IEEE 802.11a standard.
- **LTE-Advanced/4G**: Is the primary communication standard for V2I communication. Unlike the previous versions of 3G technologies, LTE uses an "all-in-one" approach with everything over IP. The LTE-A has a feature of carrier aggregation which combines multiple channels to provide higher data rate support. The LTE provides shorter round trip times, more high spectral efficiency, high bandwidth and mobility support. It has been tested in literature for its suitability for V2I communication.
- **Wi-Fi**: is an unlicensed band that is deployed to provide high-speed internet connectivity to devices. The Wi-Fi stack was not designed to support the mobility of vehicles, but improvements have been made in IEEE 802.11 to provide support for fast roaming via IEEE 802.11r, improved bandwidth and security for V2V and I2V communication.
- **5G**: To support communication between automated vehicles and infrastructure, it is required to provide ultra-low latency, very high data rates and high reliability. 5G provides a scalable, and highly reliable platform to support novel applications including vehicular applications related to self-driving cars. The 5G-enabled communication provides proximity services that can be used to discover devices and services in a locality. This will especially help in service and device discovery in the moving vehicular networks. The proximity services reduce the traffic congestion at core networks by enabling high data rate and secure device-to-device communication platform [Shah et al., 2018]. The proximity services also help in beaconing safety-

related messages to nearby vehicles which were largely constrained by the available bandwidth in IEEE 802.11p.

2.3.6 Communication Management

We do not cover communication management in this work, but we briefly present communication management in IoT networks. IoT protocols are classified as data-centric, message-centric and resource-centric [Teo and Kadir, 2006]. The data-centric approach is based on a publish-subscribe message to send information from a data source to the sink. MQTT is a message-oriented protocol, where sensor devices are directly connected to compute devices, whereas compute nodes are connected to back-end servers via brokers. The distributed services need to communicate at intra-Fog and inter Fog-Cloud levels to deploy the hierarchical Fog computing paradigm. Due to the constrained devices, many Constrained Application Protocol (CoAP) and REST-based communication protocols are developed but have the issue of overhead costs.

For interoperability between sensor clouds (using peer-to-peer communication), wireless and wired mobile edge devices, compatible protocols are required for seamless hierarchical communication. Providing device abstraction and virtual networks are required to manage geographically distributed Cloud and Fog domains. The communication between resource-constrained IoT devices/sensors uses the Constrained Application Protocol (CoAP). The communication between sensors and compute nodes also need to be facilitated for seamless data processing tasks. The next level of communication is required between Fog and Cloud nodes for coordinating the execution flow among different service components, hosted at a different level of the network. The communication between end devices could be peer-to-peer, could be via Edge nodes or could be via Cloud nodes, based on the latency imposed in the design.

2.4 Literature Review

This section presents the literature review of fog computing, VCC and VFC. We also present the related works on task allocation schemes and scheduling schemes in vehicular fog which is the main focus of our research work. We classify the task offloading schemes based on the application type they consider i.e. latency-sensitive and data-centric application types and the QoS measure they aim to optimize.

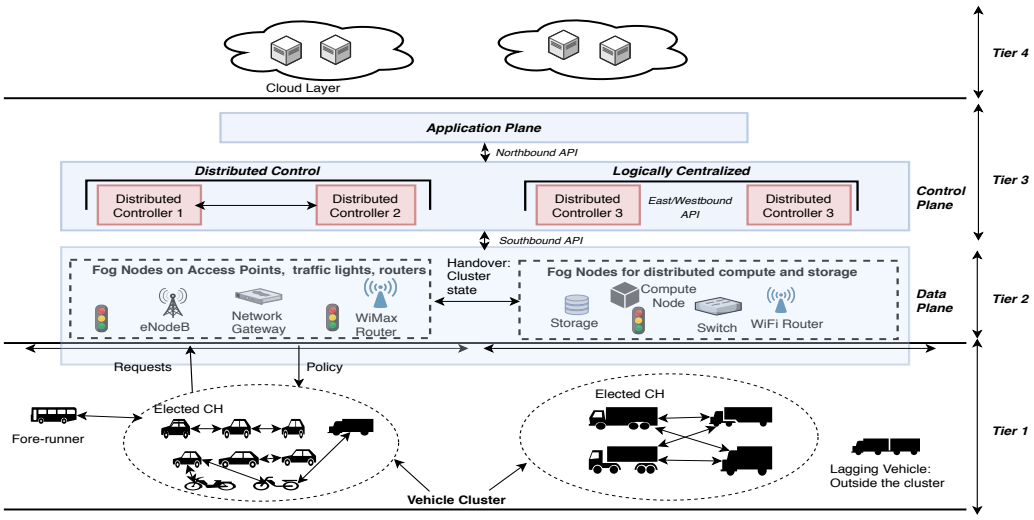


Fig. 2.4 The use of distributed controllers to manage the resource and mobility state of the four tiers of the VFC architecture as Fog nodes, the data plane as eNodeB, and network gateways, the control plane and the Cloud computing layer.

2.4.1 Fog Computing Architecture, Feasibility and Use Cases

With the emerging IoT paradigm and the enormous number of heterogeneous and ubiquitous end devices, there is abundant computation and communication capacity available at the edge of the network. These resources can be used to support latency-sensitive and bandwidth-hungry applications and to reduce the burden of increasing service requests at the data centre. There is an increasing demand for mobile multimedia services including mobile gaming, audio, and video streaming which contributes to network traffic and can cause network load and poor QoS/QoE. These technological trends have led to the emergence of MEC which reduces the traffic load on the backbone network by utilizing the edge resources for local computation, network control, and storage [Chen et al., 2016]. Another distributed computing paradigm, called Fog Computing introduces a system-level, horizontal architecture that distributes resources and services of computing, networking, storage, and control anywhere between end devices and cloud[2]. The highly virtualized Fog paradigm supports services and applications with widely distributed deployments. The Fog is suitable for providing streaming services to moving nodes, emergency and healthcare applications, gaming, and augmented reality with very low latency requirements.

This transition from Cloud to Fog required different programming models, architecture, service offloading and placement schemes to enable heterogeneous and geo-distributed devices to provide computation or networking resources to service requests in the vicinity.

[2]https://opcfoundation.org/markets-collaboration/openfog/

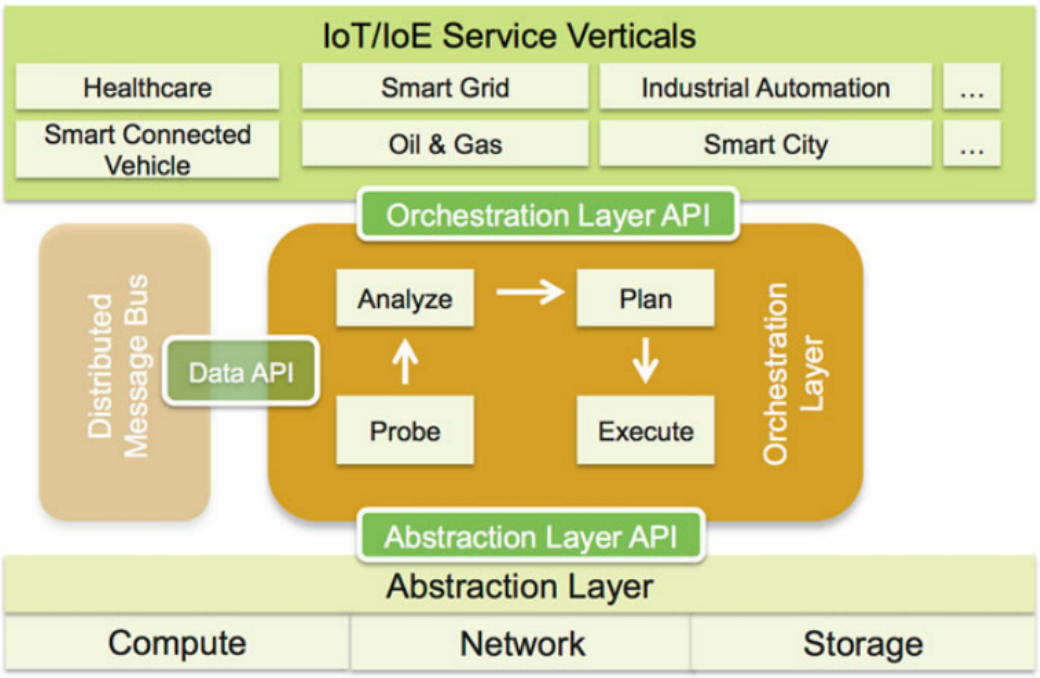


Fig. 2.5 Components of Fog architecture introduced in [Bonomi et al., 2014].

These issues in Fog have been studied widely in literature, which we summarize here. From an architecture point of view, it requires horizontal and virtual scaling of service requests according to the delay constraint, geographical and temporal scope, and computation requirement of the application. [Sarkar et al., 2018] introduced the three-tier Fog Computing architecture with terminal nodes as *Tier 1*, which form a location-based logical virtual cluster that transmits data to Fog Instances at *Tier 2*. The Fog layer comprises devices like router, gateway, and switch, to process and analyse data to be either sent to Cloud if it requires historical data-based analysis or is processed locally. The limited storage and processing capacity at the Fog layer is to be used for latency-sensitive applications, or applications that require location or context awareness. The Cloud layer forms *Tier 3*, which receives limited or controlled requests from the Fog layer. Fig. 2.1 represents the use of distributed controllers to manage the resource and mobility state of the four tiers of the VFC architecture as Fog nodes, the data plane as eNodeB, and network gateways, the control plane and the Cloud computing layer.

[Bonomi et al., 2014] introduced a high-level description of Fog's software architecture that is presented in Fig. 2.5. It consists of the Fog Abstraction layer for seamless resource management for the heterogeneous Fog platform by providing generic APIs for monitoring, controlling, and provisioning resources. The layer also supports virtualization for multiple OSes or running service containers for multi-tenancy. They also introduce a distributed

Policy management based service orchestration layer for life-cycle management of Fog services. A small software agent *foglet* monitors the state of the devices and deployed services using abstraction layer APIs, a distributed database stores application data and metadata for service orchestration, and a service orchestration module is responsible for the distributed policy-based routing of application requests to an appropriate service instance. Similar to [Bonomi et al., 2014], [Yi et al., 2015] also introduced a fog computing platform composed of authentication and authorization, offloading management, resource management and VM scheduling components. They also introduce the design goals and challenges of a fog computing platform. They discuss the choice of virtualisation technology for the platform, the latency in the platform, network management and security and privacy concerns in the platform.

Aazam and Huh [2016] introduced a *Fog Smart gateway (FSG)* which gathers data from underlying nodes. It has enhanced capability for carrying out pre-processing or interoperability-related tasks. An FSG has contextual awareness based on receiving feedback from the application and hence communicates data only when required. They introduced a layered architecture based on tasks undertaken by the Fog. The architecture has a physical and virtualization layer that consists of physical nodes, WSNs, virtual nodes, and virtual sensor networks. The layer above it monitors the underlying nodes and networks. The layer manages task allocation for underlying nodes and their energy consumption for effective management. The pre-processing layer performs data filtering and trimming. The temporary storage layer keeps the data until it is uploaded to the cloud. The private data for health care services or location-aware data is secured in the security layer. The transport layer uploads the data to the cloud. They focus on the functional aspect of different components in the Fog for efficient and in-time scheduling and management of resources.

Hong et al. [2013a] introduced a programming architecture called Mobile Fog for IoT applications that are geo-spatially distributed, large scale and latency-sensitive. To orchestrate the highly dynamic and heterogeneous resources at different levels of network hierarchy, they introduce a PaaS programming model. The model provides a simplified programming abstraction that supports the dynamic scaling of applications during runtime. The Mobile Fog architecture runs the same application code on various devices including smartphones, smart cameras, connected vehicles, fog computing nodes and the cloud. It simplifies the scaling and heterogeneity problems in large scale application deployment. The architecture also provides control interfaces to manage applications and communication APIs to enable interaction between the distributed application components. Giang et al. [2015] also introduced a programming architecture that deals with distributed data flow (DDF) model. The application topology is expressed as a directed graph where each node represents an

application component. The model allowed developing and deploying an application for heterogeneous nodes according to their resource capabilities.

2.4.2 Vehicular Cloud and Fog Computing

Gerla [2012] 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 novel applications for VCC including urban sensing especially uploading videos for congestion and pavement conditions that other vehicles could access and also suggested carrying computations on these clouds for the collaborative reconstruction of an accident scene or urban population map etc. Some other works in VCC included modelling the traffic flows to represent the mobility behaviour of the vehicles. In [Boukerche and Grande, 2018], three mobility models are considered: a car-following model which is a microscopic model in which every vehicle's behaviour is characterized, a traffic stream model in which behaviour of individual cars is not considered and vehicle stream parameters are considered, and stochastic traffic models, wherein roads have grid topology and traffic follows the random movement. This model does not consider interactions between vehicles or other correlated parameters like vehicle density, speed, and flow rate.

Similarly, many papers explored the VFC concept, which focuses on on-demand resource provisioning using on-board units in parked or moving cars. Sookhak et al. [2017] introduces a Vehicular Fog architecture with a Fog sublayer and a vehicular cloud sublayer. They also elaborate on a decision manager who computes the task completion time and assigns the task to the required sublayer. Zhu et al. [2018] analyzes video crowdsourcing using VFC. They first analyse the availability of vehicle fog nodes using real-world trace data. They then evaluate the serviceability of this model by estimating the network performance over two access technologies: LTE and DSRC. Ning et al. [2019] introduced the three-layer VFC architecture to enable distributed traffic management to minimize the response time of city-wide events. A VFC-enabled offloading scheme is formulated as an optimization problem by leveraging moving and parked vehicles. They minimize the expected response time by assigning traffic flows to the cloudlet and parked and moving vehicles.

2.4.3 Programming model for Future Internet Applications

Future Internet applications are large-scale, latency-sensitive, and geographically distributed. Unlike traditional web applications, future Internet applications run on distributed mobile and sensor devices. The programming model for these applications have challenges like synchronization between distributed instances, device heterogeneity, mobility of devices, and

thereby lead to reliability and security issues. These applications need to run dynamically without causing more network usage or control overhead.

Programming models for deploying services at Network edge: Fog devices are network devices that are equipped with additional storage and compute power towards the edge of the network. Thus, it is essential to judiciously manage resources to match the resource capacity of traditional servers. Aazam and Huh [2015] presented a service-oriented resource management model. Fog not only provides services on an ad-hoc basis but also, has to estimate the consumption of resources so that they can be allocated in advance. Resource prediction allows more efficiency and fairness at the time of consumption. As mentioned, the requests can be made from objects or nodes as well as devices operated by people. Therefore, prediction and pre-allocation of resources also depend upon the user's behaviour and the probability of using those resources in the future. In the model, they have taken into account the variance of relinquish probabilities, which helps to determine the actual behaviour of each customer

Hong et al. [2013b] introduced Mobile Fog, a Platform as a Service (PaaS) programming model to provide programming abstraction and support applications that scale dynamically at run time. The high-level programming model simplifies development on a large number of heterogeneous and distributed devices and supports automatic scaling based on workloads and available resources capacity of all nodes in the network. Mobile Fog processes are mapped to distributed computing instances in Fog and Cloud. The application logical structure is a hierarchical model where processes on the edge are leaf nodes, processes at the cloud become the root node while the process on the Fog layer becomes intermediary nodes. These processes have a communication path between them, through the Mobile Fog communication API. The hierarchical communication API *send_up* and *send_down* and point-to-point communication API called *send_to* are developed to encourage in-network processing and run-time scaling to support the mobility of nodes.

Wang et al. [2018] introduced fog-based architecture and programming model for distributed coordination within fog computing nodes in the smart grid. The computing coordinator periodically gathers information on remaining resources in fog nodes, assigned tasks, etc. It also supports the collaboration of computing nodes to carry out complex tasks. In their programming model, the mobility of the terminal node is encountered by initiating migration within compute nodes using two APIs. State on_migration_start returns state object for flow information, waiting for migration. Void on_migration_end is invoked by the target device after receiving a request message from the migration initiator.

Some applications developed for deployment in the Fog are based on Distributed Data Flow(DDF) model. In the DDF programming model application is modelled as a collection

of modules, which constitute the data processing elements. This application model allows representing application components as a directed graph with vertices representing modules and directed edges showing the flow of data between modules. Two models that are suitable for fog computing architecture are:

- Sense-Process-Actuate Model: The information collected by sensors is emitted as data streams, which are acted upon by applications running on Fog devices, and the resultant commands are sent to actuators.
- Stream Processing Model: The stream processing model has a network of application modules running on Fog devices that continuously process data streams emitted from sensors. The information mined from the incoming streams is stored in data centres for large-scale and long-term analytics.

2.4.4 Feasibility of using vehicles as infrastructure

Hou et al. [2016] introduced the concept of using under-utilised vehicular resources for service provisioning. They focus on four types of scenarios of moving and parked vehicles as communication and computation infrastructure. They derived a relationship between communication capability, connectivity and mobility of vehicles. The computation capacity is analyzed by vehicle density, staying time and the incoming and outgoing process models of vehicles, which resulted in predictable computing capacity for both moving and parked vehicles. Another system, called JamCloud [Xiao et al., 2019] also works on estimating the computation capacities by introducing performance metrics like the average number of mobile clouds that satisfy computation demands. They identify a unique application of using these mobile clouds at intersections to carry 5G BS's Baseband signal processing tasks, apart from using the mobile cloud for vehicle's computational requirements.

Xiao et al. [2019] studied several statistical features of a VFC system. They predicted the potential computation capacity of a VFC and studied the relationship between the communication capacity and the communication range of the vehicular fog. They also studied the temporal and spatial distribution of potential computation capacity on a city-wide scale using visualisations. These parameters can help in designing a more efficient vehicular fog system and optimise the computation capacity allocation schemes.

2.4.5 Task Allocation and Scheduling in Vehicular Fog

Service placement, task allocation and task scheduling are the main focus of our research. In this section, we aim to classify research works based on the application type they focus

on. We classify the placement and scheduling schemes based on latency-sensitive and data-intensive applications. In our review, we also classify recent works based on the QoS parameter they are aiming to optimize. There are many works undertaken to reduce the latency of applications through task offloading and deployment decisions in a VFC system. We also highlight some related works that focus on crowdsensing type applications.

2.4.6 Latency-sensitive applications

We further classify the task allocation schemes for latency-sensitive applications and their allocation schemes in a fog computing paradigm and a VFC ecosystem. The task allocation in VFC has added complexity compared to scenarios with static infrastructure because of the following issues [Skarlat et al., 2016]:

- Because of the mobility of vehicles there is frequent handovers between smart vehicles and RSUs which makes resource management more challenging. The mobility states of moving vehicles have not been studied extensively for utilising close-moving vehicles for the service execution. How long do vehicles stay together and how can this information be utilised for service deployment is an important parameter to make VFC systems more robust.
- The wireless channel conditions and content popularity are not known before the service deployment. The wireless channels conditions are time-varying and difficult to predict.
- The computation task or content also needs to be divided into components/segments to support computation offloading or edge caching. How to divide and place these service components is also an added complexity to the resource management problem.

2.4.6.1 Task allocation in fog computing for latency-sensitive applications

Table 2.1 List of publications focusing on task allocation for latency-sensitive applications.

Papers	*Task allocation problem*	*Optimizing Metric*
Xu et al. [2017]	Auction-based mechanism for resource contract establishment and latency aware scheduling technique.	Maximising utility for edge computing infrastructure provider and service provider.
Gu et al. [2018]	Addresses a joint radio and computational resource allocation problem for fog computing: using student project allocation (SPA) game, to provide a distributed solution.	Optimising the user satisfaction as well as system constraints such as service delay, transmission quality, power control.
Ni et al. [2017]	Dynamic resource allocation strategy for Fog Computing based on priced timed Petri nets (PTPNs) also propose method for credibility evaluation of both users and fog resources.	Considers price cost and time cost to complete a task.
Zeng et al. [2016]	Effective resource management and task scheduling mechanism to balance workloads between client side or edge side.	Minimising task completion time in Fog computing based Software-defined embedded systems.
Intharawijitr et al. [2016]	Policy based selection of target Fog server for task placement.	To minimise the rejected workload to the total number of workloads in the entire system, called blocking probability.
Skarlat et al. [2016]	Suggest the independent execution of Fog applications, without the involvement of Cloud, by introducing a Fog Middleware.	Monitors computation resource usage and QoS metric, i.e. execution time.

Applications that have strict latency requirements need to have proper resource configuration and task placement to reduce the execution time of the application. [Zeng et al., 2016] have

worked on an effective resource management and task scheduling mechanism for minimizing task completion time in Fog computing based software-defined embedded systems. Unlike the traditional embedded system, the task image in their model is not fully loaded into the embedded system but is placed on an edge server. It is retrieved on-demand during run-time. The I/O interrupts are also handled by edge servers instead of a standalone embedded system. However, due to limited resources on the network edge, certain task performs better when placed on clients. They balance workloads between the client-side or edge side to minimize computation and transmission latency of all requests.

The communication time in exchanging data between different fog nodes also has a significant impact on application performance. [Intharawijitr et al., 2016] introduces a policy-based selection of target Fog server for task placement. They select the target using a random policy, the lowest latency policy, and a maximum available capacity policy. They work for an objective to minimize the rejected workload to the total number of workloads in the entire system, called blocking probability. They also calculate the total latency of an accepted workload, which is composed of computing and communication delay, to be less than the maximum tolerant latency of a service. Their results show that selecting Fog nodes that provide the lowest latency given the current state of the system, perform better than other policies in terms of blocking probability, by executing workload in a shorter time and releasing resources sooner for accepting other workloads.

In [Oueis et al., 2015b], radio access points are clustered for additional computation capacity to execute a requested task. They propose adaptive sizing and resource management of these computation clusters. The objective of the optimization problem is to minimize power consumption in the cluster while meeting the latency constraint of each user request. They compare their model to *no clustering* schemes which result in no power consumption in communication but poor QoE. They also compare it to the *static clustering* scheme which results in higher power consumption to achieve task completion without latency violation.

In [Skarlat et al., 2016], a conceptual framework for fog resource provisioning is introduced. They suggest the independent execution of Fog applications, without the involvement of Cloud, by introducing a Fog Middleware comprising of *Fog Orchestration Control Nodes*. Their framework highlights other important aspects for the composition of Fog colonies that comprise of *Listener* that receives task requests, *Monitor* that observes service execution and the *database* that stores the current system state as well as the received service request. They also cover the instantiation, deployment, starting, and stopping of services through REST APIs. They also introduce components to scale the services horizontally to other Fog colonies based on monitoring the computation resource usage and the QoS metric, ie. execution time.

Goudarzi et al. [2021] introduced an application placement technique for concurrent IoT applications in Edge and Fog computing environments. They propose 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. The aim is to optimize the execution time and energy consumption of IoT devices. They also introduce a failure recovery mechanism to overcome potential failures in the execution of tasks in runtime. Table 2.1 summarises all the publications discussed in this section, highlighting the task allocation problem and the optimizing metrics.

2.4.6.2 Task allocation in VFC for latency sensitive applications

Yadav et al. [2020] have investigated the joint energy and latency tradeoff based, efficient task offloading and computation resource allocation for VFC while considering both mobility and end-to-end latency constraints. They assume participating vehicular nodes to be either taxis or buses. They consider a hierarchical model which consists of cloudlet nodes or mini data centres comprising of RSUs, gateway routers, set-top boxes and vehicular nodes. the second layer comprises user equipments (UEs) equipped with sensors, cameras, GPS devices and onboard computers. The third layer is the cloud data centre which is a standalone computational platform. They consider a mobility model based on the dwell time of vehicular nodes in the service zone based on [Mohd Zaini et al., 2016]. The model uses a geographical model for predicting the dwell time within WLAN to derive a probability for the vehicular node to move out of the coverage area. The benefits brought by the proposed scheme are: reduces the risk of overload, minimize the offloading failures, maximize the energy saving, minimising service latency, and reducing the time complexity.

Dai et al. [2019] propose the use of BS and RSUs for executing resource-intensive tasks and caching resources on BSs whereas small and localised content is cached at the RSU. They introduce a cross-layer, two-tier infrastructure using the small cell BS and RSU for the task offloading. They also highlight key issues in existing edge computing and caching schemes that make them incompatible to be used in the vehicular scenario. Firstly, wireless channel condition and content popularity are assumed to be known in advance by existing schemes. Secondly, the high mobility of vehicles results in dynamic communication topology, especially between vehicle pairs, which makes the task offloading in vehicular networks more complex. They then propose a deep reinforcement learning approach named Deep Deterministic Policy Gradient (DDPG), for the edge computing and caching problem. However, they considered a simplistic mobility model where each vehicle is assumed to be driving at a constant speed on a road segment. They do not study the relative mobility of vehicles but consider the sojourn time with the RSU.

Ma et al. [2019] 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. They have introduced a task offloading and take-back scheme which is introduced when vehicles hosting the tasks leave the current platoon. The re-offloading is carried out by the leaving node instead of sending the request to the service requester, which effectively reduces the latency of the re-offloading process and improves the task execution rate. Lee et al. [Lee and Lee, 2020] also modified an existing RL-based algorithm to make efficient resource allocation decisions leveraging vehicles' movement and parking status to minimise service latency.

Zhao et al. [2019] 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. [Zhu et al., 2018]. They evaluated the availability of client nodes that generate data in proportion to the vehicle Fog nodes that process the data, using the processing capacity of onboard units. However, they focus solely on the data transmission problem in the model. The mobility of vehicle nodes makes the task allocation problem more challenging in VFC. Zhu et al. [Zhu et al., 2019] introduced an event-driven dynamic task allocation framework designed to reduce average service latency and overall quality loss. The task allocation problem is modelled as an optimization problem considering constraints on service latency, quality loss and fog capacity. The optimisation maintains the trade-off between service latency and quality loss. They focus on video streaming and real-time object recognition applications.

Liu et al. [2021] investigated a service scenario of task offloading under a three-layer service architecture, including resources of vehicular fog, fog server and central cloud. The three layers are used cooperatively to solve a probabilistic task offloading (PTO) problem. To solve the PTO problem they introduce an alternating direction method of multipliers (ADMMs) and particle swarm optimization (PSO), to divide the problem into multiple unconstrained subproblems that iteratively reach an optimal solution. The objective of the PTO is to minimize the weighted sum of execution delay, energy consumption, and payment cost. Liang et al. [Liang et al., 2021] suggest the use of public transport facilities like buses and taxis as fog nodes to reduce the randomness of vehicle movement with fixed bus trajectories. To solve the interruption problem caused by vehicle mobility as well as the problem of delay and reliability loss, they introduced a low-latency information

distribution scheme for VFC. They study network topology dynamics to evaluate and predict the connection status between fog nodes and the adjacent vehicles. The scheme recalculates an optimized relay route for the vehicle if a fog node finds that a vehicle may move outside its communication range in a future period.

Tan et al. [2022] decomposes the joint optimisation problem into decomposed sub-problems of task offloading and resource allocation. The task offloading problem decides where the task is to be executed whereas the resource allocation problem characterises how much computation and communication resources are allocated to the tasks. They develop a decentralised convex optimisation approach that decomposes a holistic Mixed Integer Non-Linear Problem (MINLP) into a hierarchy of convex optimisation problems. The optimisation criteria include the total latency of all tasks and the energy consumption in both vehicles and RSUs. They also presented the convexification procedure to transform the discrete optimisation problem for task offloading into a continuous convex one. Probabilities for offloading targets replace the integer design variables of deterministic task offloading targets. de Mendonça et al. [2022] investigates the trade-offs on the operation of fog nodes under different vehicle densities and network conditions and formalizes a Time Constrained One-Shot Open First Price Auction for resource allocation in VFC. They also conclude that current wireless network standards may dictate processing limits despite the availability of processing power of fog nodes.

Qiao et al. [2018] considered advanced driver assistant systems and autonomous driving as the use case for a distributed and collaborative task offloading scheme with a guarantee of low communication and computation latency. They work on removing redundant computation tasks based on task similarity and computation capacity. Vehicles are partitioned into the task computing sub-cloudlet to provide underutilized communication and computation resources. Vehicles with lesser similarities are partitioned into the task offloading sub-cloudlet to assign their computation tasks to edge infrastructures. Table 2.2 presents all the publications summarised in this section along with the task allocation problem and the optimising metric.

Table 2.2 List of publications focusing on task allocation for latency sensitive applications in VFC.

Papers	*Task allocation problem*	*Optimising Metric*
Yadav et al. [2020]	Hierarchical model which consists of cloudlet nodes or mini data centres comprising of RSUs, gateway routers and set-top boxes.	Minimise energy consumption and service latency.
Dai et al. [2019]	Introduced a cross layer, two-tier infrastructure using the small cell BS and RSU for the task offloading.	Maximise system utility.
Ma et al. [2019]	Platoon-assisted vehicular edge computing system based on the stability of the platoon in vehicular networks.	Optimises price decisions and computing resource allocation.
Zhao et al. [2019]	Designed a collaborative optimisation scheme where offloading decisions are made through a game-theoretic approach and resource allocation is achieved using the Lagrange multiplier method.	Task processing delay, cost of computation resource, and the normalisation factor.
Zhu et al. [2019]	An event-driven dynamic task allocation framework.	Minimise average service latency and overall quality loss.
Liu et al. [2021]	To solve a probabilistic task offloading (PTO) problem under a three-layer service architecture.	Minimise the weighted sum of execution delay, energy consumption, and payment cost.
Liang et al. [2021]	Solve the interruption problem caused by vehicle mobility as well as the problem of delay and reliability loss.	Minimise latency.
Tan et al. [2022]	Decomposes the joint optimisation problem into decoupled subproblems of task offloading and resource allocation.	Minimise total latency and energy consumption.
Qiao et al. [2018]	A distributed and collaborative task offloading scheme with a guarantee of low communication and computation latency.	Low communication and computation latency.

2.4.7 Data-centric applications

Table 2.3 List of publications focusing on task allocation for data-centric applications.

Papers	*Task allocation problem*	*Optimising Metric*
Shi et al. [2015]	Present an alternative to the hierarchical view on fog-computing by enabling device clouds to interact in a P2P fashion with smart device/sensor clouds.	Enabling seamless access between smart devices and sensors and mobile device resources using the IoT protocol CoAP.
Nazmudeen et al. [2016]	Framework for distributed data aggregation between sensors and Smart Grid Cloud applications.	Fog-based approach for data processing outperforms traditional approaches in-terms of increasing the virtual capacity of PLC and improved response times.
Oueis et al. [2015a]	Algorithm for fog cluster formation and load balancing is designed for computation offloading on small cell clusters.	Proposed a joint optimisation of computational and radio resources that minimises the power consumption per user while meeting all the latency constraints imposed by each user.
Yu et al. [2018]	Study joint application placement and data routing to support all data streams with both bandwidth and delay guarantee.	Proposed algorithms greatly improve the quality-of-service of the IoT applications compared to the heuristics.
Ni et al. [2018]	Fog assisted secure data-deduplication scheme. Chameleon hash function is used for contribution claim and reward retrieval.	To improve communication efficiency.

These applications are also crucial for managing the enormous data generated in the IoT ecosystem. Some important works related to the data-centric application are:

In [Nazmudeen et al., 2016] a framework for distributed data aggregation between sensors and Smart Grid Cloud applications is introduced. Currently, the big data collected from consumers is stored in a centralized place for processing and forecasting the energy demand. This becomes a bottleneck for efficient data collection due to the limited bandwidth capacities of power line communication. Their fog-based approach to data processing outperforms traditional approaches in terms of increasing the virtual capacity of PLC and improved response times.

The data flow between distributed nodes in Fog computing affects bandwidth usage and service latency and is an important consideration in service provisioning. In [Oueis et al., 2015a], an algorithm for fog cluster formation and load balancing is designed for computation offloading on small cell clusters. The small cell cloud cooperates to form computation clusters subject to momentary and local resources available, base station deployment scenarios, offloaded applications delay constraints, and power consumption budgets. They propose a joint optimization of computational and radio resources that minimizes the power consumption per user while meeting all the latency constraints imposed by each user. The algorithm has a customizable design where metrics, scheduling rules, and clustering objectives can be set according to specific applications and network requirements.

Application provisioning in Fog computing comprises two sub-problems node selection, for local processing as well as for potential forwarding or routing. Then optimal path needs to be formed for the in-network processing of data. In [Yu et al., 2018], the authors present both single and multiple application provisioning in an IoT environment. For multiple applications, they consider both parallel processing using multiple instances as well as non-parallelizable applications. Vehicular micro cloud has been studied as virtual edge servers for efficient connection between cars and backend infrastructure in [Hagenauer et al., 2017]. They use map-based clustering at intersections, as intersections have a line of sight in multiple directions which results 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.

Few papers have considered user mobility, specifically in mobile-sensing based applications. Ni et al. [2018] proposed a fog-assisted crowdsensing based framework by which tasks are allocated to users based on user mobility. They also consider another important issue in mobile crowdsensing, ie. a fog-assisted secure data-deduplication scheme to improve communication efficiency. They also address another contradictory problem, where a user providing duplicate data also needs to be rewarded, as this data increases trustworthiness.

For this issue, they use Chameleon Hash Function by which mobile users can claim their contribution and receive rewards.

Both multi-source data acquisition and distributed computing in Fog-computing based intelligent vehicular networks are studied by [Zhang et al., 2017]. They introduce a hierarchical, QoS-aware resource management architecture, but consider the Fog servers as static. They discuss the four functions of building regional cooperative fog computing-based IoV architecture to deal with the big data generated in smart cities. the functions include mobility control, multi-source data acquisition, distributed computing and storage, and multi-path data transmission. The hierarchical resource management model includes both energy-aware and QoS-aware resource management. Table 2.3 summarises the publications discussed in this section along with the task allocation problem addressed and the optimising metric that is targeted.

2.5 Summary: Challenges and Limitations

This chapter first presented the detailed background of the evolution of VANET to the IoV ecosystem where vehicles are utilised as processing and communication infrastructure. We also give details on VFC architecture, applications for vehicular use cases and the suitability of service placement in the VFC ecosystem. We then give a detailed outlook on key challenges involved in the application placement in the VFC scenario. We first discuss the main goal of task offloading in VFC which is to effectively distribute applications over the fog nodes to improve the QoS by reducing task response time and improving network bandwidth and computation resource utilization. We then extend on node selection and service mapping techniques that need to be adapted to determine the most appropriate node for service deployment. We then give details on flexible service models, service discoverability, advanced QoS/QoE and service migration required in adopting services on a distributed and dynamic VFC platform.

The background also gives details on mobility models for mobility prediction of vehicles, which is an important parameter to consider for leveraging the under-utilised vehicular resources. Even though the vehicular communication networks are not the primary focus of this research work, it is important to study VRATs and communication management to understand how vehicular networks operate and what are the resource limitations and shortcomings of the moving network. Next, a very detailed literature review has been presented with a focus on fog computing and VFC architecture, the feasibility of using fog computing principles for leveraging vehicular resources and discussed the use cases considered in the literature. We also discuss works undertaken in introducing programming

models for future internet applications. We discuss the works undertaken to study the feasibility of using vehicles as infrastructure. To the best of our knowledge, the feasibility studies for using vehicles as infrastructure are very limited. The stability and mobility patterns of vehicles need to be studied to build service models and placement schemes relevant to the VFC scenario.

We give a very detailed review of task allocation and scheduling in both fog and VFC paradigms. We classify the recent works based on the QoS objectives and application types. Based on our literature review, we observed the following gaps in the literature:

- We also observed that many important schemes and algorithms in the VFC ecosystem consider very simplistic mobility models. They either consider parked vehicles or taxis and buses as the fog nodes which have much more predictable mobility patterns. Most existing works do not consider the dynamics and mobility patterns of vehicles with respect to other vehicles in the vicinity. There is also limited work undertaken to study the predictability of vehicular flow in urban city centres, and the estimated computation and communication capacity of moving vehicle clusters. We address this research gap in **RQ 1. How feasible is it to use vehicle clusters for service execution? How to estimate the aggregate computation and communication capacity of these vehicle clusters?**

- Secondly, many existing works on VFC consider scenarios, where vehicles are coupled with RSUs, and computation is carried out on these vehicle nodes with tight coupling and management from stationary nodes. Our work studies service placement on moving vehicles, where closeby vehicles are utilised for the service execution. In some applications with a local context, the end-to-end application is executed entirely on the vehicle clusters. We address this research gap in **RQ 2. How can closely moving vehicles be used for service provisioning?**

- Thirdly, most existing works consider static services with pre-determined CPU and memory requirements. These service models are not suitable for the dynamic vehicular network, especially with node and link failures due to the mobility of nodes. In our work, we introduce flexible and scalable services that increase the resilience of the service and utilise heterogeneous resources on different vehicular nodes. We address this research gap in **RQ 3. How to resolve the vehicle cluster selection, service scaling, and service placement problem using community detection and graph-based algorithms?**

- There are limited works undertaken using real data from the vehicular testbed. We use a federated learning scheme to implement an object detection application for

detecting pedestrians and cyclists using video data from the testbed. We address this challenge in **RQ 4. Can a federated-learning-based scheme be introduced to deploy a distributed object detection service using video data from multiple video sources?**

Chapter 3

Macroscopic and Microscopic Mobility Modelling and System Model for the Service Placement Problem

3.1 Introduction

The future generation of vehicles will have both increasing computational demands due to the adoption of compute-intensive applications as well as would have powerful computational resources in the form of System on Chips (SoC) [MEOLA, 2020]. Many of these *smart* vehicles will use a lot of the processing capacity for self-driving tasks including moving object tracking, path planning, route planning, etc. In our model, we encourage the use of inbuilt sensors in moving vehicles for collecting data as well as using unused processing capacity for hosting applications to process the collected data. Thus, the mobility and behaviour patterns of vehicles become an important parameter in service placement decisions in our work. A lot of existing works that introduce procedures and methods to utilise vehicles for their computational capacity consider the mobility of vehicles as user equipment with respect to stationary RSUs. They use sojourn time as a measure of the time spent by a vehicle in a cell or coverage area while using a communication channel [Wang et al., 2019]. In other cases, only parked vehicles [Zhang et al., 2019b] or vehicles with a more predictable trajectory like buses and taxis are considered as potential fog nodes [Ge et al., 2020].

With the advent of *smart* vehicles, the mobility patterns of vehicles can be identified and utilised to study congestion patterns in urban traffic. The influx of communication capabilities has led vehicles to be independent communication networks that we aim to leverage for service placement in this work. A major drawback in utilising vehicular networks

as infrastructure is the lack of vehicular mobility data including macroscopic features for private vehicles. To fill this gap of unavailability of realistic vehicular mobility data we focus on the following aspects of urban vehicular mobility: 1. We show that vehicular flows can be predicted at intersections using their density data. 2. We simulate microscopic vehicular mobility models, calibrating them with real macroscopic data. 3. We study the relationship between vehicular speeds and vehicular occupancy/density to understand if there are threshold speeds at which slow-moving vehicular clusters can be initiated. 4. We also estimate the available aggregate communication and computation resource capacity of these moving vehicle clusters.

We identified a gap in recent works, where the relative mobility of vehicles with respect to neighbouring vehicles is not considered to make task offloading decisions on closely-moving vehicles. Either analytical mobility models are introduced in the service placement model or the mobility of vehicles is considered as part of simulations or vehicular traces, which includes parameters like location and speed [Zhang et al., 2019a]. In some cases, a very simplistic mobility model is considered with all vehicles travelling at the same speed and in the same direction [Dai et al., 2019]. In this chapter, we use macroscopic vehicular data at intersections to study the predictability of vehicular flows and use these macroscopic mobility data to calibrate the microscopic mobility model for the vehicles.

We first present the motivation for using moving vehicles as infrastructure for service placement in §3.2. As part of the motivation, we introduce a generalised vehicular flow model at an intersection to classify traffic flow into six different driving profiles. We employ multivariate linear regression (MVLR) model that considers vehicular flow for seven consecutive weekdays to predict the traffic flow in §3.2.1. To compare the performance of our model with competing schemes we use a random forest-based regression model and ARIMA model for time-series forecasting. As part of the motivation, we also analyse the aggregate communications and computation capacity estimation for a vehicular network in § 3.3.

Even though the density or occupancy of vehicular traffic is an important parameter in vehicular mobility studies, we have not identified existing works in VFC literature that considers vehicular density as a parameter for service placement. To study the level of congestion in urban traffic, we take data from the California Department of Transport (Caltrans) [Author, 2020] dataset from detectors on the I-405 freeway, known as one of the busiest freeways in the USA. We provide the speed versus occupancy graphs to analyse the usage patterns of freeways on different days of the week in §3.4. We also utilise a level of service (LOS) parameter to understand traffic conditions at different times of the day. In §3.5, we provide a brief on how microscopic trajectories are simulated to generate the CCP of the

vehicles, which is an important parameter for our service placement problem. Finally in §3.7, we define the system model, the network topology, and the service model used in the service placement scheme introduced in this dissertation.

3.2 Motivation

Our work is motivated by the increasing number of smart cars with embedded sensors that can connect to other cars, and the unresolved issue of vehicle congestion—especially in urban areas. To make the service scaling and placement scheme more robust 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.

This chapter focuses on the following aspects to study vehicular mobility and the use of mobility patterns for vehicular cluster selection and placement:

1. **Predictability of vehicular flows at intersections**: We first study whether traffic flows in an urban setting are likely to be predictable over the course of a day. We use an MVLR-based scheme for predicting vehicular flows. We also highlight comparable mobility schemes in the literature and select random forest and ARIMA models as the competing schemes for predicting vehicular flows.

2. **Aggregate communication and computation cost estimation**: We study whether a slow-moving cluster will accommodate sufficient communications capacity between vehicles to facilitate service operation.

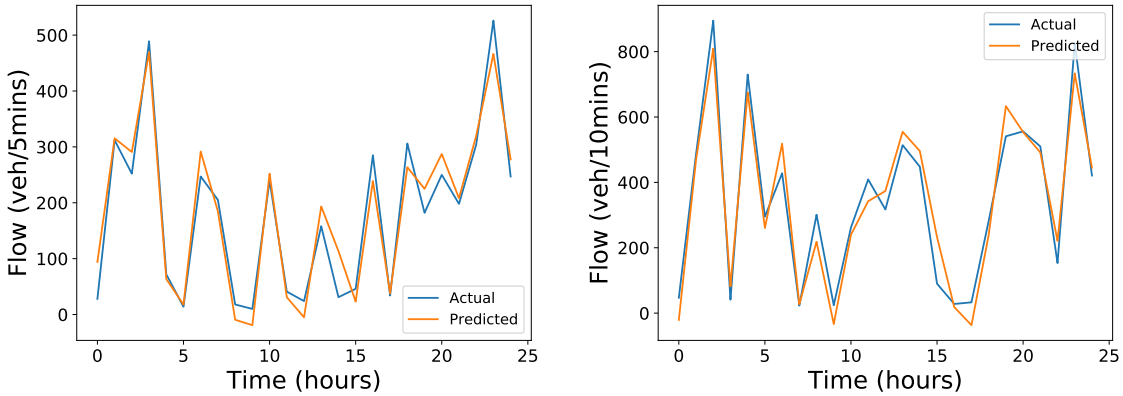
3. **Density of vehicular traffic in highly congested urban areas**: We study how closely spaced vehicles are in congested urban areas and if there is a threshold speed at which vehicle clusters can be initiated. The vehicular density is an important parameter to use in moving vehicles for executing distributed services. As services are broken down into data-dependent tasks, these tasks are required to be placed on slow-moving vehicles, to reduce service failures due to high mobility. We use California Transport data to observe the patterns of vehicular speeds in correlation with vehicular density for weekday and weekend traffic. We also use a LOS parameter to study the traffic conditions at different times of the day.

3.2.1 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

Fig. 3.1 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.

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.



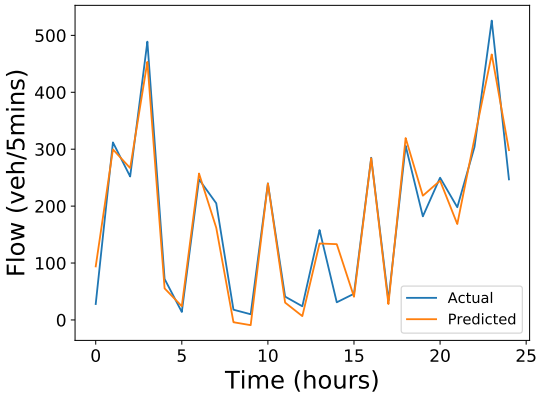
(a) LR model for traffic prediction every 5 minutes

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

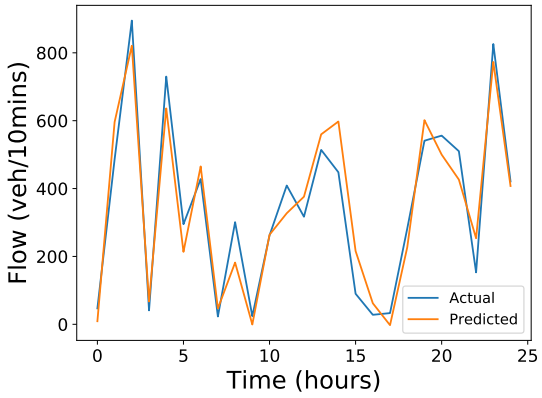
Fig. 3.2 Traffic Prediction using real vehicle density data; these data depict the consistent and predictable vehicle densities at the intersection using a linear regression model.

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 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. 3.1), 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

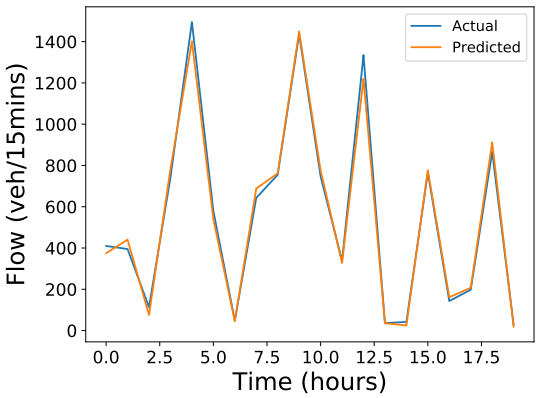
[1] https://trafficdata.tii.ie/publicmultinodemap.asp, available: 6/02/22.



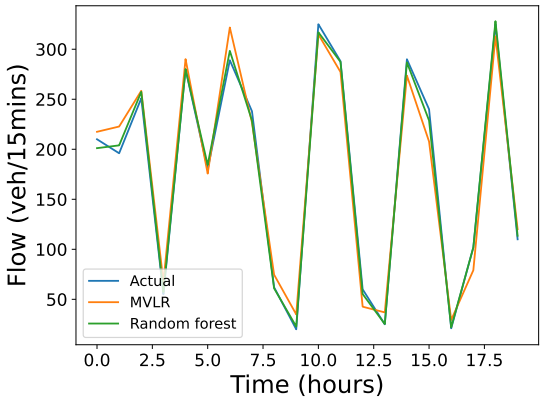
(a) Multivariate linear regression model for traffic prediction every 5 minutes



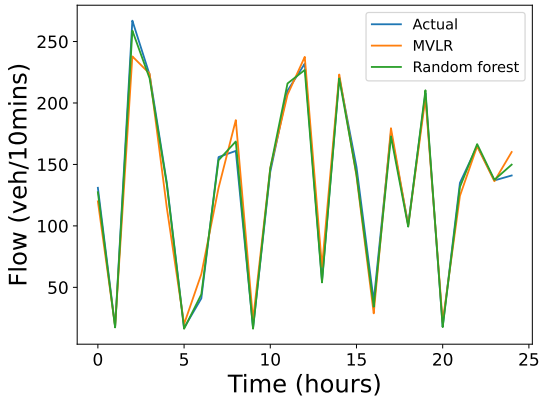
(b) Multivariate linear regression model for traffic prediction every 10 minutes



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

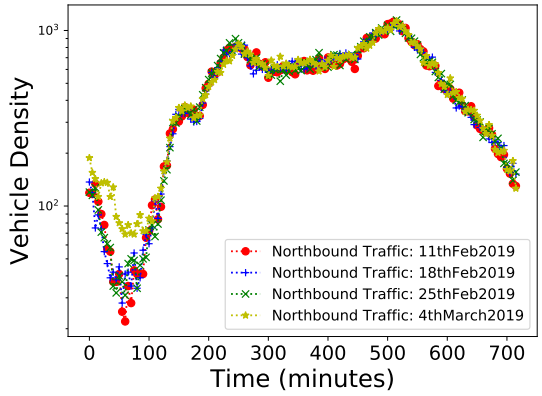


(d) Multivariate linear regression model for traffic prediction every 15 minutes, April 2020

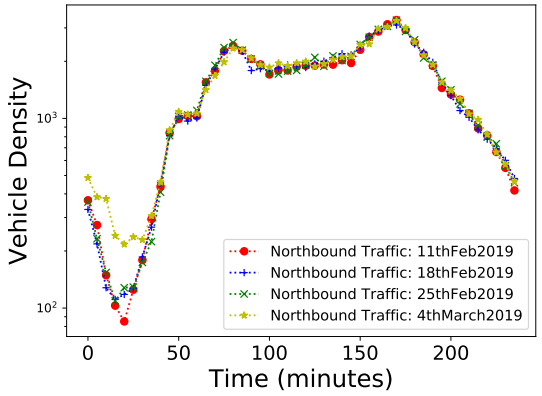


(e) Comparison of MVLR with random forest for traffic prediction every 10 minutes, April 2020

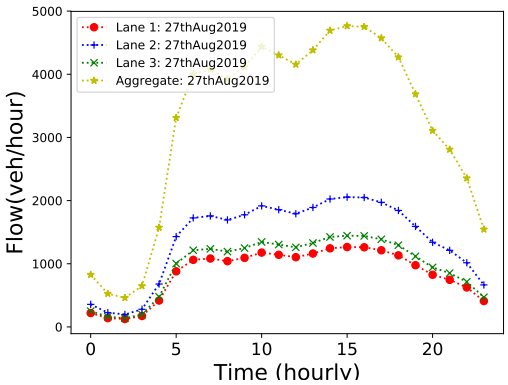
Fig. 3.3 Comparison of MVLR with competing schemes.



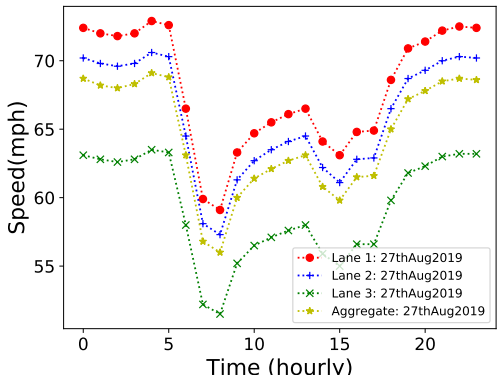
(a) Vehicle density recorded every 10 minutes



(b) Vehicle density recorded every 30 minutes



(c) Vehicular flow over a period of 24 hours



(d) Vehicular speed over a period of 24 hours

Fig. 3.4 Vehicle density, vehicular flow and vehicular speed recorded in different time intervals.

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 seven consecutive weekdays to use a MVLR model. We analysed the data to study how vehicular flow patterns change over time. We noticed that vehicular flow varies significantly between weekdays and weekends. Hence, we consider only weekday traffic data. We also analysed that traffic flow also varies significantly during bank holidays and holiday season. Thus, we chose a window of seven consecutive weekdays as it generated most accurate results for vehicle density prediction. 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. 3.2a) and 10 minutes (Fig. 3.2b). 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 3.1 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. 3.3a), 10 (Fig. 3.3b) and 15 (Fig. 3.3c) 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. 3.3a, 3.3b and 3.3c. 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. 3.3e and Fig. 3.3d depict predicted vehicle flow using MVLR, considering seven consecutive days during the lock-down, with an R^2 score of 0.98 and 0.987.

We also compare the prediction of MVLR with random forest regression, as depicted in Fig. 3.3e and Fig. 3.3d, 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 3.2. The random forest performs marginally better than MVLR. MVLR

Table 3.1 The r-value, p-value and standard error for predictability of the six flows at the intersection.

Flows	*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 3.2 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

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. 3.4a) and 30 minutes (Fig. 3.4b). The figures depict the consistent and predictable vehicle density data for both northbound and southbound traffic for all four weeks.

3.2.2 Comparative mobility models

To predict the vehicular flows at intersections, we introduce a multivariate linear regression (MVLR)-based scheme that accurately predicts vehicular flows at an intersection. To compare our schemes to existing models, we did a literature survey specific to mobility models in vehicular fog computing (VFC) and traffic flow modeling studies. Lee and Lee [2020] formulated the problem of allocating the limited fog resources to vehicular applications such that the service latency is minimized, by utilizing parked vehicles. The paper considers a

simple mobility pattern where vehicles are generated based on sine functions with a period of 10-time slots. For the realistic vehicular mobility scenario, they used vehicular mobility traces in Zurich to model vehicles arriving and departing from parking lots. In both cases, the model is not comparable to our mobility model. Zhou et al. [2020] proposes a novel resource management and task offloading framework for VFC-enabled autonomous driving. In this work, a time-slotted model is adopted where many vehicles or potential fog servers in the coverage area of base stations vary in different slots due to the mobility of vehicles. They assume vehicles to be stationary through consecutive time slots and do not consider real vehicular data to model mobility. Noorani and Seno [2018] proposed a method to improve data sharing in vehicle-to-vehicle communications to cover communication coverage holes. They propose intersection-based routing by leveraging fog computing and software-defined networks. However, the intersections have been simulated using SUMO and OpenStreetMaps. They do not use real data to study vehicle mobility behavior at intersections.

Xiao et al. [2019] do not study service provisioning on moving vehicles but analyze the potential computing capacity of a vehicular fog. They take microscopic traces specifically of taxis in Beijing. Even though they have considered many intersections in their study, their mobility modelling is not comparable to our flow model of the intersection. Our model uses macroscopic vehicular data to make inferences on the predictability of vehicular flow and the opportunistic clusters are initiated in an area of dense traffic and are not specific to taxis. The percentages of vehicles that are taxis would be much lower and would be widely spaced to other taxis, resulting in lower density and sparse communication between vehicles. Wang et al. [2016] suggests the use of both time-series methods based on historical data and machine learning methods for short-term traffic prediction. The time-series methods forecast the future traffic flow via internal statistical features of observed data sequences. To compare the performance of our MVLR-based flow prediction model for the intersection, we use both random forest regression and ARIMA modelling for time-series forecasting.

Random Forest (RF) is a supervised learning algorithm and is an ensemble learning method. It uses the Bagging technique of data randomization and uses multiple Decision Trees such that all calculations are run in parallel and there is no interaction between the Decision Trees. The randomness is first introduced in the model by creating k unique sample sets of training data, which helps in making decision trees more uncorrelated. The second set of randomization comes from splitting each node in every Decision Tree using a random set of features. RF has been used for traffic flow prediction in literature [Liu and Wu, 2017] and is a powerful technique for comparing against our linear MVLR approach. The RF achieves an r-squared value of 0.99 in comparison to our approach and performs marginally better than our MVLR approach with an r-squared value of 0.97. RF also has a disadvantage in that

Table 3.3 Comparative mobility modelling schemes in literature.

Comparative schemes	*Real vehicular data used*	*Scenario*	*Comparable to our flow model*
Lee and Lee [2020]	Yes	Parking modeling	No
Zhou et al. [2020]	No	Not specified	No
Noorani and Seno [2018]	No	Simulated for inter sections	No
Xiao et al. [2019]	Yes	Microscopic traces used	No
Wang et al. [2016]	Yes	Macroscopic mobility model for intersections	Yes

it cannot extrapolate and the predicted values are never outside the training set value for the target variable whereas a linear regression model can extrapolate based on the data. It is safe to say that the two models can be used for short-term traffic prediction interchangeably.

We also use the ARIMA model which stands for autoregressive integrated moving averages. ARIMA model is widely used for time series forecasting [Alghamdi et al., 2019], and we use the model for non-stationary, short-term prediction for vehicular flows. We first convert the non-stationary data to stationary data by differencing and then testing the normality of the dataset. After pre-processing the data we use the ARIMA model to fit the time series. However, the R squared error is much lower for the ARIMA model (0.69) and the RMSE and MAE scores are much higher compared to the MVLR and RF models.

3.3 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 been done in great detail via customized theoretical studies [Chen et al., 2018; Grossglauser and Tse, 2001; Mao et al., 2013]. We calculate the effective capacity of the vehicular network obtained using a cooperative scheme from Chen et al. [2018].

Theoretical Capacity: We consider the closed-form expression of effective available capacity specified by Chen et al. [2018], 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 ($\mathrm{BW}_{\max}$), given as:

$$\mathrm{BW}_{\max} = \frac{L}{d}\min\{W_I(1-\exp^{-2\rho r_I}), W_I(1-\exp^{-p\rho 2r_I}) + \frac{W_V.c_2(d-2r_I)}{c_2.R_C + p - p\exp^{-2pr_o}} + \exp^{-p\rho 2r_o}\} \tag{3.1}$$

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. [2019], 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.

3.4 Mobility Patterns of Vehicles in Highly Congested Urban Areas

Even though it is known that vehicular congestion is a major problem in both urban sections of the cities and busier freeways, it is crucial to study the mobility patterns of vehicles to decide in which sections of the city can vehicular clusters be initiated for data collection and processing. Our service model requires vehicles to be closely spaced to each other to deploy the distributed data-dependent applications. Hence, we strictly focus on very slow-moving vehicles as a high-speed vehicle cluster would lead to more service failures and require many service re-configurations, increasing the management overhead. The traffic congestion is estimated using either the density or the occupancy of a road segment. Density is a measured, spatial quantity that represents the number of vehicles averaged over a spatial distance(per lane or mile), whereas occupancy is an observed value collected by detectors[2]. Occupancy is calculated as the percentage of time in which the vehicles are passing over the detector. We chose occupancy as a measure to see how closely spaced vehicles are and at what occupancy do the vehicles become slow-moving or reach a complete breakdown condition.

The speed versus flow (number of vehicles passing a detector) and the occupancy versus flow graphs are standard traffic theory plots that are commonly used in transport research. However, we plot the speed versus occupancy graphs to understand the correlation between how closely spaced vehicles are and if there is a threshold speed at which vehicles come to a halt. We take the data from the California Department of Transport (Caltrans) dataset from detectors on the I-405 freeway, known to be one of the busiest freeways in California. We plot the speed versus occupancy graphs from a detector on the freeway. As can be seen in Fig.3.5, the speed of vehicles tends to zero for lane 5, declining from 40% to 60% occupancy, whereas for lane 4, the speed gets at 10 mph after 20% occupancy. Similarly, for lane 3, the speed gets lower than 20 mph and tends to zero after 35 % occupancy. We do piece-wise fitting for the speed and can see a breaking point after which vehicular speeds stabilizes to slower speeds for all lanes and in some cases vehicles stop completely. This highlights that there

[2]http://pems.dot.ca.gov/

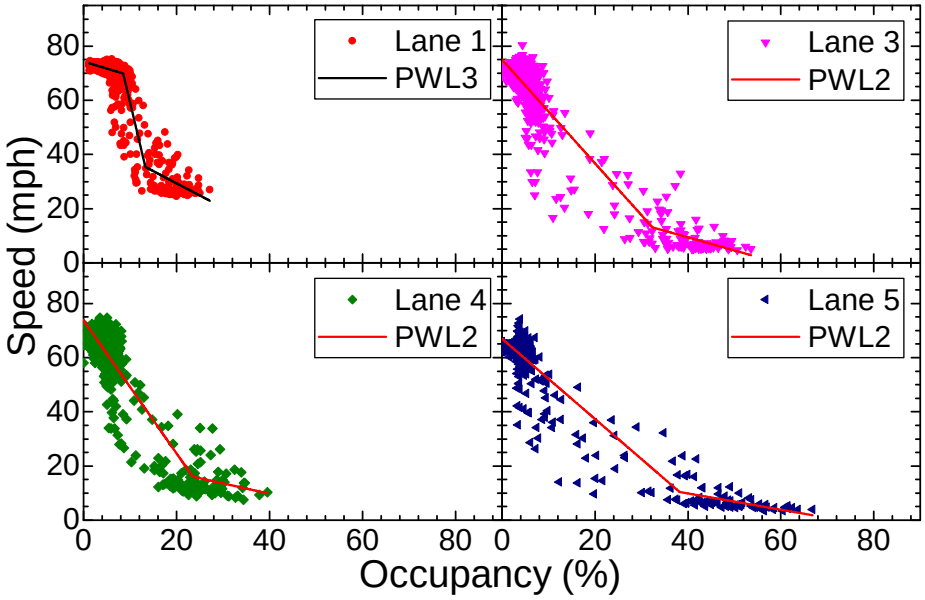


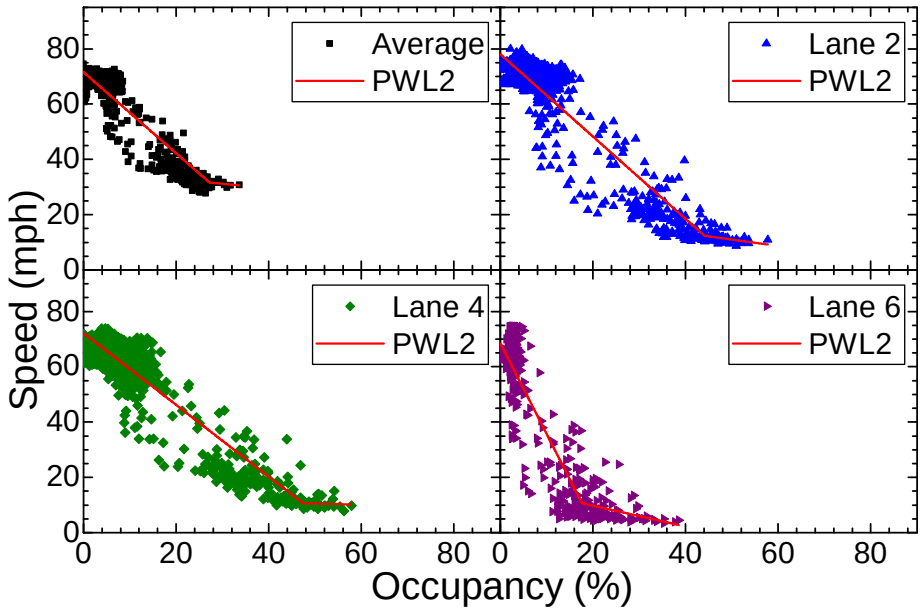
Fig. 3.5 Speed versus occupancy graph for a detector on the I-405 freeway for seven consecutive days.

are enough vehicles that are closely spaced in busier road segments, moving at very slow speeds. This also helps in identifying where and when clusters can be initiated based on the available vehicular occupancy. However, the threshold speed of vehicles slowing down and congesting is different for different road sections. It is not practical to infer vehicle speeds at which clusters can be initiated. Occupancy percentage is a more appropriate measure to study how closely-spaced vehicles are. The occupancy is also independent of the shape of road segments.

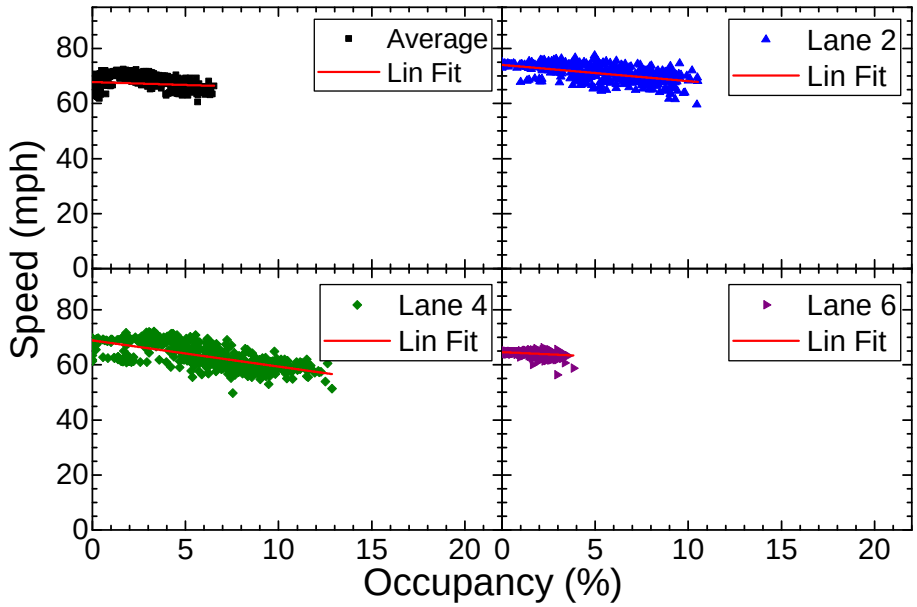
We have observed that the freeways follow very different occupancy and speeds during different days of the week. We compared the occupancy versus speed graphs for the same detector on the I-405 freeway on weekdays (Fig. 3.6 (a)) and weekends (Fig. 3.6 (b)). We observed that high occupancy is observed only on the weekdays. Such analysis is crucial to study when can vehicle clusters be initiated and which road segments get congested.

There is another standard way to study the traffic flow conditions. Caltrans PeMS (https://pems.dot.ca.gov/) provides a parameter called the LOS, which uses vehicle density to analyze the quality of service or the condition of the traffic along a freeway. The freeway LOS is a way to classify the traffic condition into a grading system ranging from A to F. The relationship between LOS, density and the capacity of a freeway is defined in the Highway Capacity Manual (HCM2010) and is summarized in the table 3.4.

The percentage of vehicles in each LOS profile is plotted for different times of the day for the I5-N freeway for the first week in August 2019 in Fig. 3.7. As can be noted, there is



(a) Speed versus occupancy graph for a detector on the I-405 freeway on weekdays



(b) Speed versus occupancy graph for a detector on the I-405 freeway on weekends

Fig. 3.6 Speed versus occupancy graphs on weekday and weekend.

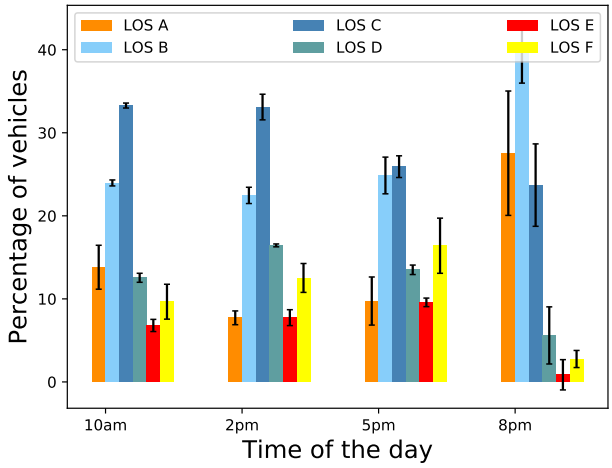


Fig. 3.7 The LOS parameter for the I5-N freeway to understand the traffic conditions at different times of the day. The y axis shows the percentage of vehicles in each grade of LOS. LOS A depicts virtually free flow to LOS F which represents breakdown condition.

Table 3.4 Relationship between density, volume and the LOS on a freeway.

LOS	*Density (veh/mi/lane)*	*Volume/Capacity*	*Traffic Flow Description*
A	$0 \leq x = 11$	$x \leq 0.3$	Virtually free flow; completely unimpeded.
B	$11 \leq x \leq 18$	$0.3 < x \leq 0.5$	Stable flow with slight delays;reasonably unimpeded.
C	$18 \leq x < 26$	$0.5 < x \leq 0.71$	Stable flow with delays; less freedom to maneuver.
D	$26 \leq x < 35$	$0.71 < x \leq 0.89$	High density, but stable flow.
E	$35 \leq x < 45$	$0.89 < x \leq 1.0$	Operating conditions at or near capacity; unstable flow.
F	$45 \leq x$	$1.0 \leq x$	Forced flow, breakdown conditions.

a significant percentage of vehicles in LOS E which depicts unstable flow due to near full density of traffic, and LOS F which depicts flow breakdown conditions. The percentage of vehicles in LOS E and F constantly increases from 10 am to 5 pm. Almost 20% of vehicles are in LOS F at 5 pm, which represents breakdown or traffic jam conditions. The percentage of vehicles increases significantly in LOS A and B at 8 pm, which depicts free flow and a stable flow of traffic.

This analysis also gives an estimate of available vehicle density at different times of the day. This also reduces the uncertainty in resource availability and identifies time slots when it's viable to initiate a vehicular cluster. The LOS profile of vehicles gives a better idea of vehicular traffic being in free-flow or stationary condition in comparison to threshold speeds. We only intend to initiate clusters in areas that are known to have high-density traffic during certain times of the day.

3.5 Microscopic and Macroscopic traffic trajectory data

The utilization of the predictable vehicle trajectory and available processing capacity requires knowledge of both the microscopic behaviour of individual vehicles as well as the overall macroscopic traffic flow dynamics. In our work, we use both macroscopic and microscopic vehicular data to make predictions on the future trajectory of each vehicle in a region with dense and slow-moving traffic. Our model uses real macroscopic data to calibrate the simulation model at intersections to generate microscopic trajectories of vehicles. In essence, our model provides a way to use minimal data i.e., vehicular flow data, to select a robust vehicular cluster for service placement. The basis of selecting the vehicular cluster, based on their historic mobility patterns, is to select vehicles that are more probable to stay together for the duration of service execution. This will lead to less task failures due to vehicle nodes leaving the cluster.

From a networking point-of-view, a well connected multi-hop cluster will have multiple paths from one node to another in the cluster. This will ensure the successful execution of the distributed service with data-dependent tasks. To ensure the selection of a well-connected cluster, we use community detection algorithms like Louvain method and Girvan and Newman method, to identify the most well connected nodes in the vehicular cluster. As depicted in Fig. 3.8, the urban intersection with more dense and slow moving traffic can be identified using their speed versus occupancy graphs. Once density is estimated, the vehicular flows can be predicted using historic mobility patterns of vehicles on different road segments. Using these vehicular flows, we model the microscopic mobility trajectories of each vehicle. We first extract road network using Open Street Map (OSM). We then use a road traffic

simulator called Simulator for Urban Mobility (SUMO) to model the microscopic trajectory of each vehicle. We use real vehicle flow data to calibrate the microscopic car-following model to make it as realistic as possible based on the flow model in §3.2.1. After generating calibrated traffic data in SUMO, we run a number of simulations to derive the CCP of each vehicle. The vehicular network is represented as a directed graph. The joint CCP of the two vehicles is used as the weight of each edge in the graph. The edge weight is used for selecting nodes that are more probable to stay as members of the vehicle cluster. We evaluate the quality of the selected nodes, based on a graph centrality measure, which is detailed in Chapter 5.

The microscopic trajectory of individual vehicles cannot be used explicitly because of privacy concerns over the use of user trajectory data by third parties. Therefore, to reduce privacy concerns, microscopic data can be considered at and between specific intersections, so there is no need to know the trajectory for the entire journey of a vehicle. This method can be predicted as the joint probability of vehicles starting at a road segment, say RS_j, and ending at the road segment say RS_k, expressed as:

$$\begin{aligned} &P(src = RS_j, dest = RS_k) \\ &= P(src = RS_j).P(dest = RS_k) \end{aligned} \tag{3.2}$$

The macroscopic traffic data includes flow level variables like traffic flow rate, traffic density, and average velocity of the traffic stream. This data is easier to collect, using the vehicle counter and cameras commonly installed in cities for traffic management purposes. Macroscopic models also include deriving the relationship between traffic speed, flow rate, and density to estimate slow-moving vehicle traffic to initiate vehicle clusters. Most traffic estimation studies utilize both microscopic and macroscopic data to estimate vehicle trajectories.

3.6 Vehicular Fog Marketplace

The problem of deploying edge servers and utilizing the traffic density in urban centres can be resolved by introducing a vehicular fog marketplace, where vehicles can temporarily lease some of their video capturing, sensing, computing, and networking capabilities. This marketplace would include *consumers* in the form of service providers looking for reliable vehicular resources to capture and process information. The computational resources they seek can be used for applications that go beyond our motivating use case of crowdsourcing. Such additional use cases include intensive machine learning applications that can be implemented

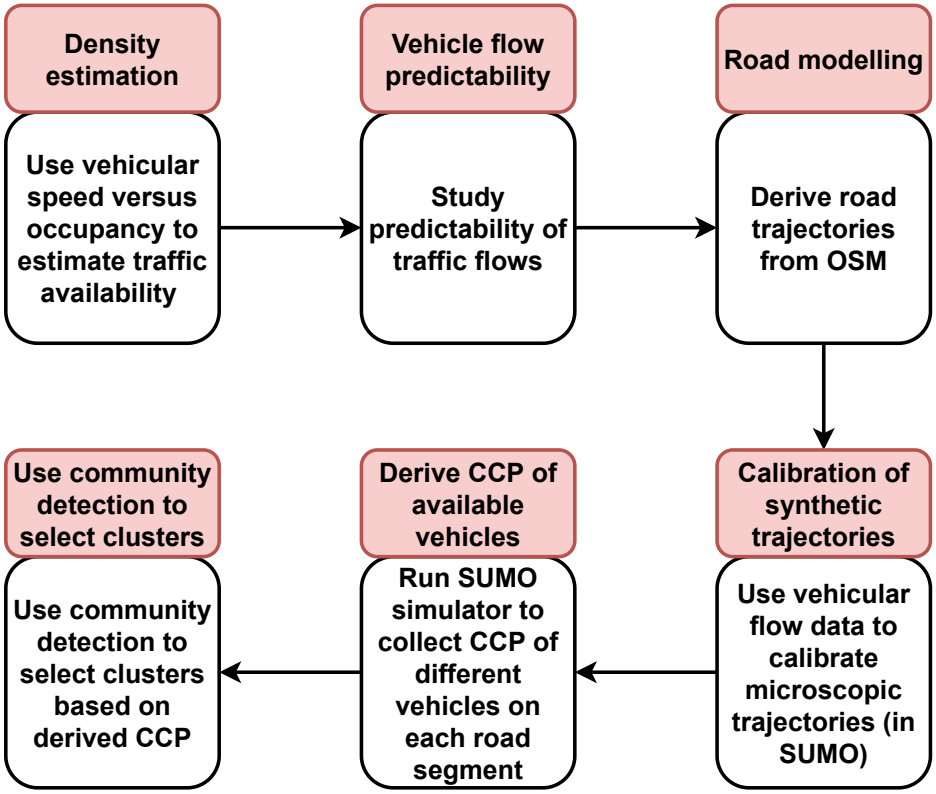


Fig. 3.8 The process of density estimation, vehicle flow predictability, road modelling, calibration of microscopic trajectories, to derive CCP of available vehicles and selection of vehicle cluster.

in a distributed manner. The *producers* in the form of participating vehicles offer to host applications that pay a fair price while leasing the least amount of resources. The service provider aims to process most of the information on the vehicle cluster and collect as much data as possible, subject to the limitations of the infrastructure made available, collectively, by the vehicles in the cluster.

The service providers could also be transport or motor authority, willing to collect surveillance data to improve the traffic and other aspects of the city. Surveying is an expensive task and requires a lot of human hours and resources. The process of data collection, as well as processing, can be initiated on moving vehicles, as proposed in our work. These vehicles can be incentivised by getting free parking or waiving their toll fee, in return for leasing their resources for service deployment. Thus, a marketplace can evolve based on the historic performance of service completion by different vehicles. The vehicle owners can also decide which service to agree to based on the incentives of performing the service. In the longer run, such a set-up can provide a cheaper way for service providers to extend their infrastructure and collect ad-hoc data in different locations. It can also provide a means for vehicle owners to earn credit while they are stuck in urban traffic.

This marketplace has been explored in the context of implementing complex, distributed machine learning models in an edge computing marketplace [Yerabolu et al., 2019]. The approach has been compared to job completion time with third-party cloud providers. A similar marketplace needs to be studied for a moving vehicular cluster use case, in terms of monetary cost and task satisfaction rate.

3.7 Service Scaling and Placement Scheme

For efficient placement of services on virtualized vehicle resources, we require the RSUs to act as service coordinators. With this role, the RSU should have knowledge of the system state of the vehicle cluster which is communicated by the selected CN in that cluster. These nodes would have high connectivity to all the other nodes within the cluster. The mobility of vehicles is what makes the service placement problem in vehicular networks different from the service placement in cloud networks. It makes resource availability a function of time owing to both resource usage and vehicle dynamics. We formulate the placement of Tasks on virtualized vehicle nodes as an optimisation problem. The solution of the optimisation problem is a service configuration mapped onto a cluster formed between closely associated vehicles, with efficient communication links for end-to-end service execution.

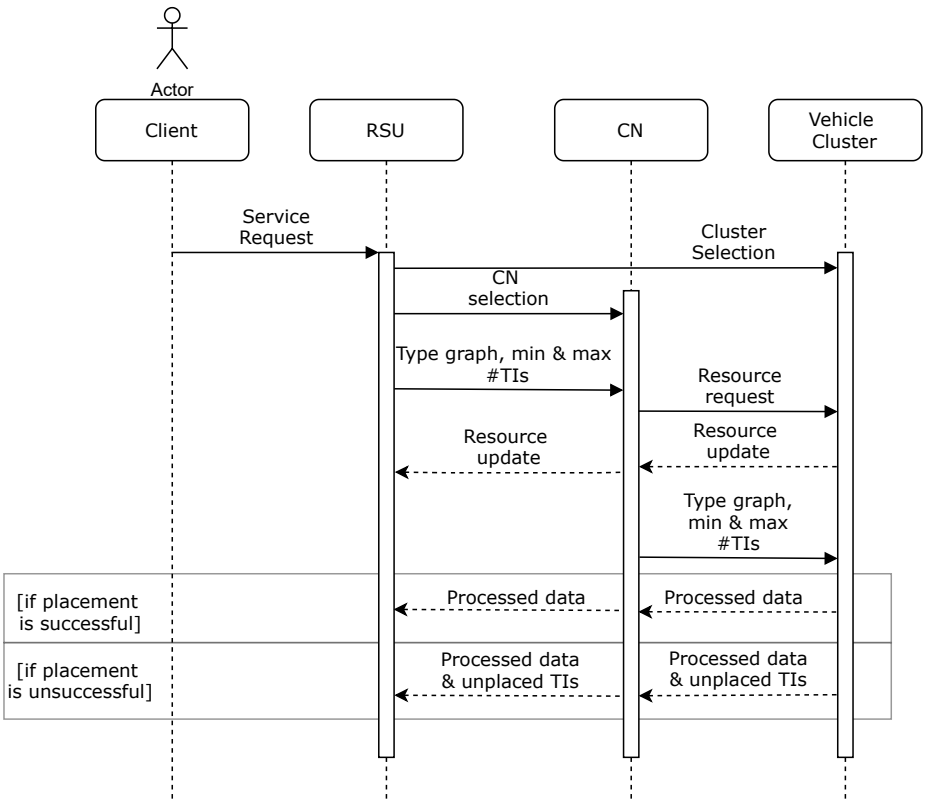


Fig. 3.9 System Model depicting the management between the RSU, CN and the vehicle cluster.

3.7.1 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.

3.7.1.1 Terminology

The terminology used in the system model is presented below:

- **Vehicle Clusters**: We consider vehicle clusters as micro cloud-like entities [Higuchi et al., 2018], 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 chapter). 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. 3.9. 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. 3.10. 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. 3.10) 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 [Zhang et al., 2020].

- **Service Placement**: The process of placing the scaled Instance graph (in Fig. 3.10) 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 optimised based on minimising 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 optimised Instance graph onto the physical vehicle nodes. We jointly consider both TI mapping as well as the route/flow mapping, between the placed TIs.

3.7.2 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

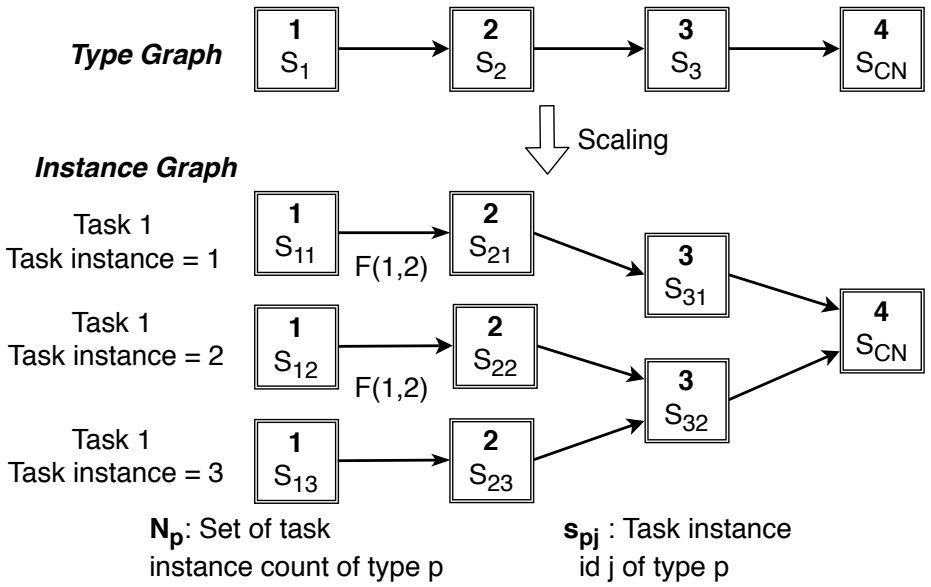


Fig. 3.10 Service model depicting tasks and their inter-dependencies. The Type graph is scaled to the Instance Graph based on the resource state of the vehicle cluster.

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.

3.7.3 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.

When setting up the problem, each of the node capacities C_{ik} and the bandwidth limits $\{B(i_1, i_2), i_1 \neq i_2\}$ need to be supplied as input. After placement, A_{ik} represents the allocated resource of type k at node i. The CN of the cluster is represented as i_o. It acts as a gateway between service nodes and external resources such as the RSU/client. $\mathbb{I}_o$ represents the total number of participating nodes in the service-based cluster. Note that $\mathbb{I}_o \geq \mathbb{I}_{min}$, the minimum number of nodes required for service execution. $\mathbb{I}_o$ is a subset of all $\mathbb{I}$ nodes in the opportunistically formed cluster.

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.

3.7.3.1 Service Mapping based on Requests

The RSU takes t_o time to estimate the available resources of all nodes in an opportunistic (location-based) vehicle cluster via the CN(i_o). On receiving a service placement request, the RSU initiates a deployment which is forwarded to the CN in the cluster. The service placement request is in the form : $R =< t, \mathbb{S}, t_T >$ where t is the time instant at which the

deployment request is received and $\mathbb{S}$ is a set of Tasks to be deployed on available vehicle nodes moving in close proximity.

The RSU derives a more comprehensive specification of the service based on local information like link capacity and node resource availability, as follows: $R =< t, G(\mathbb{S}_{pj}, e), \mathbb{N}_p, \{D_{p,k}\}, t_T >$. $\{D_{p,k}\}$ represents the demand for resource type k of each Task s_p. t_T is the *delay threshold* of one such service request. The time duration from t to t_T is the maximum time allowed for service provisioning and execution. G is the graph (nodes as Tasks and edges as dependencies between them) used to define the arrangement of Tasks and their inter-dependencies as given in Figure 3.10. The CN needs to place Tasks according to the derived request R.

3.8 Summary

In this chapter, we have highlighted the predictability of vehicular flows using a multivariate linear regression-based model that predicts flows accurately and compared the model to other competing schemes like random forest and the ARIMA model for time-series forecasting. We present a detailed discussion on the motivation behind using moving vehicles as infrastructure. We first introduce a flow model for intersections which can be used to select different traffic flows to initiate service placement. We also made aggregate communications and computation capacity estimates for such vehicle clusters, initiated at urban intersections. We then studied the density of vehicles wrt to their speed using data from the California Department of Transport (Caltrans) dataset. The study shows how vehicular speeds decline as the density or occupancy of traffic increases. We also highlight how generalised threshold speeds to initiate vehicle clusters cannot be predicted as speeds depend on the geometry of different road segments.

In the next part of the chapter, we introduce the system model, the network mode and the service model used in the service placement scheme introduced in this work. The section also presented the terminology used in the rest of the dissertation.

Chapter 4

Scaling and Placement of Linear Service Chains on Moving Vehicular Clusters

4.1 Introduction

The network states in a dynamic and virtualized network constantly change over time, requiring flexible service provisioning and orchestration. One such scenario is when a group of distributed and connected IoT devices with sensing and computing capabilities collaborate to perform computation on locally generated sensor data [Barcelo et al., 2016]. With virtualisation and loose coupling, we can map services to resources in a shared infrastructure. These principles also enable service reconfiguration in response to changing network topology and service demands [Salahuddin et al., 2015]. The service model required for utilising these distributed IoT devices is set of smaller components called 'Tasks'with specified data flow dependency between them. These components are required to be mapped on the vehicle node infrastructure with available resources and enough bandwidth capacity between selected nodes. Each Task has a resource requirement specified as computational, memory and required link capacity. In the case of IoT applications, the service also requires access to sensors and other devices like video cameras. In the case of mobile nodes, the availability of resources is also dependent on the location of each node in space and time and its connectivity to a managing entity. In the special case of vehicular networks, location and connectivity can be predicted using mobility models based on historic routes of vehicles [Jang et al., 2017]. Predictable mobility patterns arise in the case of moderately dynamic, relatively slow moving, urban traffic, supported by RSUs. With such patterns, flexible provisioning and orchestration of services is easier.

Vehicles will be one of the most important agents in the emerging IoT ecosystem, owing to their embedded sensors and built-in cameras, which can be used to capture contextual data for object detection and surveillance [Ning et al., 2019]. 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 [Huang et al., 2017]. The increasing number of *Smart* vehicles and overall vehicular traffic has inspired the concept of VFC [Hou et al., 2016; Lee and Lee, 2020], 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 [Ning et al., 2019] and Mobile Edge computing-based models [Li et al., 2019; Ning et al., 2019] which provide ubiquitous connectivity and location-aware network responses at the edge of the network, complemented with cloud computing in the network core.

Vehicular applications include services that utilize the data sensed by the vehicles and RSUs in real-time for applications like lane changing, automated driving using motion detection or for video conferencing on the go [Suh et al., 2018]. Such applications have strict deadlines for decision-making and need to place intelligence at the network edge to make these decisions in real-time. As vehicles gain more sensing and computing resources, the idea of leasing those resources for use by third parties is gaining attention. This is because vehicles can also be seen as 'moving sensors' and as information collection agents. This data can be aggregated and sent to the cloud for application-specific processing [Ni et al., 2017]. Illustrative third-party applications include road traffic monitoring and evaluating the popularity of a particular model of a car by analysing images captured by dash cameras. In this chapter, we consider how such applications could be deployed onto moving vehicle clusters. The reliability of a service deployment on mobile nodes needs to be managed. The management system needs to take account of resource limitations including the limited bandwidth in vehicular networks, which is a reason to minimise the data being exchanged between the nodes and from the nodes to the cloud for more complex processing.

In our work, 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 [Ho et al., 2020],[Du et al., 2020]. The VFC-based data collection and processing applications can be swiftly deployed to monitor the compliance of both vehicles and pedestrians to lock-down 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 [Ye et al., 2016] and taxis [Xiao et al., 2019] 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 [Zhou et al., 2020]. 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.

Unpredictable mobility is a major problem when deploying distributed and collaborative services on infrastructure at the network edge. We focus on providing stable opportunistic service-based vehicle clusters managed by a local controller, supplemented with mobility models of the vehicle nodes, to select nodes that have a greater probability of staying within the cluster until service execution ends. Network congestion is further reduced by electing CNs that act as gateways to access points. As depicted in Fig. 4.1, one vehicle and the RSU, act as managing entities to collect and update both the resource and mobility states of the cluster and enable flexible service scaling.

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 utilise the aggregate mobility behaviour 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. The competing objectives of distributing the service widely and of minimising bandwidth usage make the service placement problem more challenging to solve.

We take the specific case of using in-built cameras in cars 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 cars. Moreover, having replicas of computing tasks enables the utilisation 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

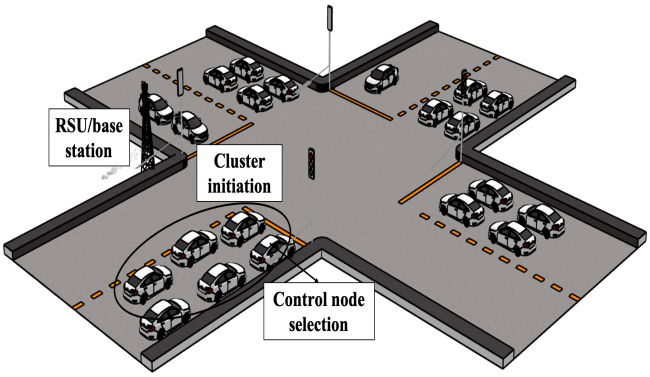
suitable vehicle so the service chain still works, instead of re-configuring the entire service. As the tasks are data-dependent, even if the cluster is computationally rich, it needs to be split into smaller TIs if the link capacity between host nodes is not enough. The task placement also needs to 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.

In this chapter, service placement is formulated as an integer linear programming (ILP) problem, whose solution is an optimal service placement plan for an opportunistically formed vehicle cluster. The goal of the ILP is successful service execution, while keeping bandwidth usage to a minimum. We 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 [Feng et al., 2017] and a clustering-based solution introduced in [Hu et al., 2021]. The experimental results demonstrate that the proposed scheme outperforms baseline approaches in terms of efficient resource utilization.

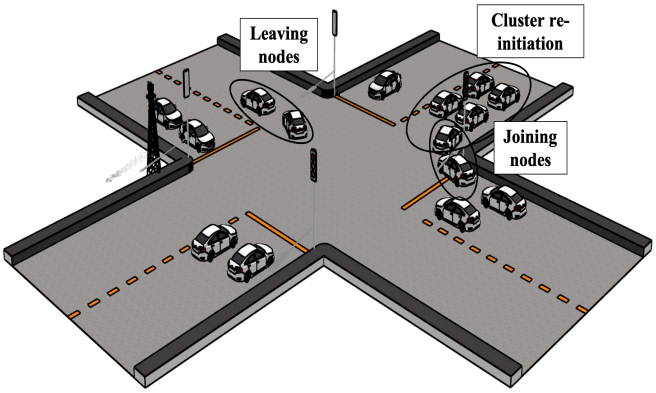
The chapter is organised as follows: In §4.2, we define the node resource constraints and the mathematical formulation of the service mapping and resource allocation problem. In §4.2.4, we introduce application types and run an experiment for an application scenario on a Fog simulator. We then discuss solving the optimisation problem, the simulation setup and our results in §4.2.5. We also discuss the performance of our model in comparison to other schemes. The chapter then presents the constraints for a more sophisticated and flexible service scaling model with multi-hop vehicular cluster in §4.3. In §4.4, we present applications and results for the bi-objective service placement plan, compared to other baseline approaches. Finally, in §4.6 we conclude the work, highlight how it addresses RQ2. and gives an outline of our future work.

4.2 Service Scaling and Placement Scheme with Single-hop Cluster

The placement solution consists of two parts: *service mapping* and *resource allocation*. The first part expresses the association between the TIs and the (vehicle) host nodes. The binary variable $M(p, j, i)$ indicates that TI s_{pj} is placed at node i at time t $\in [t_0, t_T]$. The second part indicates the amount of resource type k at each node i that is allocated to all TIs placed at that



(a) Vehicle cluster at state t_1



(b) Vehicle cluster at state t_2

Fig. 4.1 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.

node and is denoted by A_{ik}. This is calculated by summing the allocations for all TIs $s_{pj,k}$ with resource type k placed on node i. For clarity, all notations are presented in Table 4.1.

4.2.1 Node Resource Constraints

4.2.1.1 Node Capacity Constraint

The minimum resource *demand* (of type k) to host a TI s_{pj} on node i is given as $D_{pj,k}$. This TI is mapped to host i which is denoted by the binary ((0,1)-valued) 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. A minimum system resource must also be reserved for the operations required for the vehicle's normal operations. Thus, available capacity for hosting services at every node i is represented as C_{ik}, which is the *capacity* of the node. The resource constraint is formally presented as:

$$\begin{aligned} &\forall i \in \{1,\ldots,\mathbb{I}\}, k \in \{1,\ldots,\mathbb{K}\}, \\ &\sum_{\forall p,j} M(p,j,i).D_{pj,k} \leq C_{ik} \end{aligned} \tag{4.1}$$

where the decision variable $M(p,j,i) \in \{0,1\}$ is set to 1 to indicate the placement of TI s_{pj} 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. Indeed constraint 4.1 ensures that

- *for each TI* s_{pj} placed on node i, its requirements for resource type k can be met;
- *for each node* i, the total (across TIs placed on that node) demand for resource type k does not exceed the capacity of that node.

4.2.1.2 Component_Instance-Node Anti-Collocation

: Some services require TIs to be placed at different nodes to increase the fault tolerance of the system in case a vehicle leaves the service-based cluster. Also, a sensing or image capturing service might require instances to be anti-collocated to increase the area of coverage. This constraint is expressed as:

$$\forall p \in \{p^{\text{anticoll}}\}, i \in \{1,\ldots,\mathbb{I}\} \sum_{\forall j} M(p,j,i) \leq 1 \tag{4.2}$$

Constraint 4.2 is applied directly to the mapping instance variable $M(p,j,1)$. Note that it is applied at each node i and to a (subset) of Tasks $\{p^{\text{anticoll}}\}$ and uses the fact that the

sum of indicator variables $M(p,j,i)$ counts the number of TIs placed at a given node i for a non-collocatable Task s_p.

4.2.1.3 Full Deployment

: We require that *all* the Tasks of the distributed service are mapped to selected vehicle nodes at deployment time for the service to execute correctly.

$$\forall p, \sum_{\forall j,i \in \{1,\ldots,\mathbb{I}\}} M(p,j,i) \geq 1 \tag{4.3}$$

Constraint 4.3 sums the indicator variable $M(p,j,i)$ to count the number of instances (labeled by j), across all nodes (labeled by i) of a given Task s_p that have been placed. At least one such placement is required for each Task s_p to ensure full deployment.

4.2.1.4 Adjacency Constraint

When placing Tasks on nodes, it is more efficient to ensure that the placement plan takes account of both Task dependencies and 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 1 hop away from each other. Otherwise, multi-hop routing is needed, resulting in more management complexity and potentially more overhead. This adjacency constraint takes the following form:

$$\forall p_1, j_1; p_2, j_2; F(p_1,p_2) > 0, d(\mathbb{I}(M,p_1,j_1), \mathbb{I}(M,p_2,j_2)) \leq 1; \tag{4.4}$$

where $\mathbb{I}(M,p,j) = i$ when $M(p,j,i) = 1$, and $d(i_1,i_2)$ is the network hop distance between the nodes indexed as i_1 and i_2. By removing the need for multi-hop routing, it becomes much easier to ensure that enough bandwidth is reserved to support peak data flows between TIs. However, this comes at a cost: it might prove more difficult to find a feasible placement.

4.2.2 Link Constraints

4.2.2.1 Bandwidth Constraint

The bandwidth requirement between two components s_{p_1} and s_{p_2}, where the latter requires data from the former, is represented by $F(p_1,p_2)$ Kb/s. Furthermore, it is assumed that all TIs s_{pj} of Tasks s_p are identical and hence have the same bandwidth requirements. For reasons of simplicity, we consider only one-directional traffic, *from* Task s_{p_1} *to* s_{p_2}, assuming that the

reverse data flow is negligible. However, the model can easily be extended to consider duplex communication needs by adding extra constraints of the form described like constraint 4.5.

$$\forall i_1 \in \{1,\ldots,\mathbb{I}\}; i_2 \in \{1,\ldots,\mathbb{I}\}; i_1 \neq i_2 \sum_{\forall p_1,j_1;p_2,j_2;p_1\neq p_2} M(p_1,j_1,i_1)F(p_1,p_2)M(p_2,j_2,i_2) \leq B(i_1,i_2) \tag{4.5}$$

where $i_1 \neq i_2$ and $B(i_1,i_2) \neq 0$. This ensures that, for each node pair labeled by i_1 and i_2, the total bandwidth requirement, for all TI pairs $s_{p_1j_1}$ and $s_{p_2j_2}$ placed on nodes i_1 and i_2 respectively, is $F(p_1,p_2)$, which does not exceed the bandwidth limit $B(i_1,i_2)$ between the two nodes.

4.2.2.2 Mobility Constraints

Due to the mobility of vehicle nodes, we add constraints based on the dynamics of urban slow moving traffic. The *affinity* of service-bearing vehicle nodes $\{i\}$ is the probability they will stay together until the service finishes execution and is calculated using *Cluster Transition Probability*, as follows.

Cluster Transition Probability: The initial state is when the opportunistic cluster is formed and coordinated by the CN at time t_0. The transition to the next state determines the number of nodes that stay with the cluster and are able to communicate with the CN at time t_1. At each instant, we estimate this probability of each node to stay connected (possibly over multiple hops) with the CN for the duration of service execution (t_T). Ongoing connectivity is needed to minimize service migrations and reconfigurations. This transition probability is based on the historical mobility patterns followed by vehicles, recorded over a period of time. This probability is assumed to be known at each instance and can be calculated using trajectory prediction techniques, which we do not investigate in this paper. The nodes with the highest probability of staying with the CN (labeled as i_0) are selected for service placement, as this maximises the probability of successful service execution.

Transition in this case means "from the present state to the next *desired* state", so it is a numerical measure of cluster affinity. For node i, the transition probability *of staying connected to the CN* from time t_1 to t_2 is represented by $P_i(t_1,t_2)$:

$$P_i(t_1,t_2) = P\{d(i,i_0)_{t_2} < d_{\max} | d(i,i_0)_{t_1} < d_{\max}\} \tag{4.6}$$

where $d(i,i_0)_{t_2}$ is the network distance (hop count) between node i and CN i_0 at time t_2, and $d_{\max}$ is the maximum permissible hop count for the two nodes to be considered connected and hence members of the same service cluster.

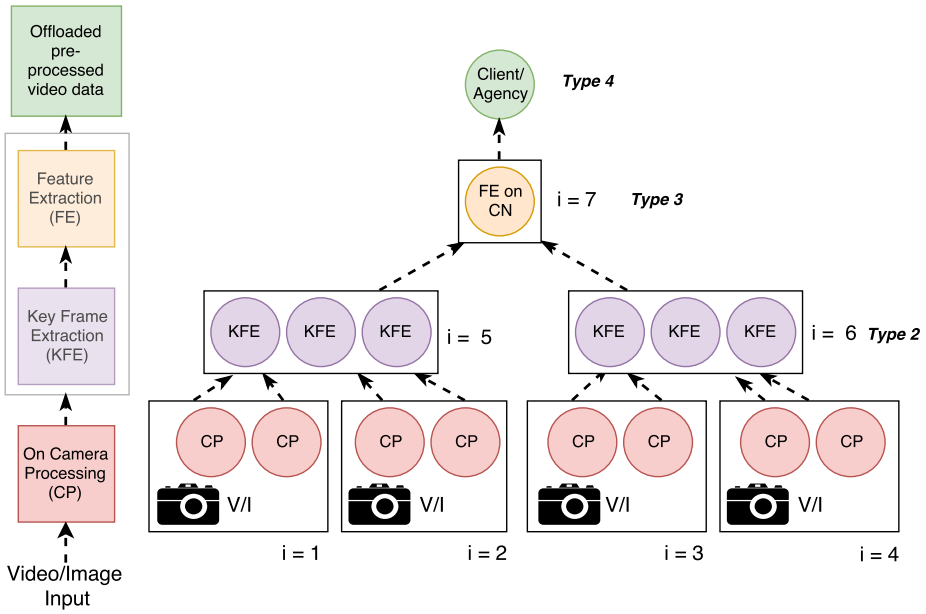


Fig. 4.2 Service Instance graph for video streaming application mapped on a service cluster of 7 vehicle nodes.

The conditional probability of the selected service nodes to stay within the service cluster from time t_1 to t_2 is given by:

$$P(t_1, t_2, \mathbb{I}_0) = \prod_{i=1}^{\mathbb{I}_0} P_i(t_1, t_2) \tag{4.7}$$

where $\mathbb{I}_0$ is the number of nodes participating in the service cluster. In practice, this probability needs to exceed a threshold $P_{\min}$ for the service completion probability to be acceptable, leading to the following service cluster cohesiveness constraint:

$$P(t_1, t_2, \mathbb{I}_0) \geq P_{\min} \tag{4.8}$$

4.2.3 Optimization

The number of TIs placed on different nodes govern the amount of data flow between nodes based on selecting the closest one-hop neighbours. TIs need to be distributed across nodes because of the anti-collocation constraint (eq. 4.2), node resource constraint(eq. 4.1) and bandwidth constraint (eq. 4.10). The adjacency constraint (eq. 4.4) and cohesiveness constraint (eq. 4.8) tend to encourage placement on nodes that are adjacent in network terms. The requirement for full deployment (eq. 4.3) encourages the placement to use more nodes to ensure that a full placement can be made, but the cohesiveness constraint (eq. 4.8) tends to ensure that as few nodes are used as possible, as the more nodes that are used, the more likely it is that at least one of the nodes will leave the cluster, other factors being equal.

Having a feasible placement is not enough, particularly when several feasible solutions are possible. Several options are possible:

1. the total bandwidth used should be minimised;
2. the relative resources (memory, CPU, etc.) used per node should be minimised;
3. the relative bandwidth used per link should be minimised.

The last of these is particularly attractive when services are to be placed on a vehicle network and the goals include:

- link bandwidth is a scarce resource, so as much as possible should be reserved for other purposes;
- the placement should be robust in the sense that, if management operations need to be applied later in response to context changes, there is adequate bandwidth to support service reconfiguration and similar actions.

The objective function is:

$$\sum_{\forall i_1,i_2;i_1\neq i_2} \min\left(\sum_{\forall p_1,j_1;p_2,j_2;p_1\neq p_2} \frac{\left(\tilde{F}(p_1,j_1,i_1,p_2,j_2,i_2)\right)}{B(i_1,i_2)}\right) \tag{4.9}$$

where

$$\tilde{F}(p_1,j_1,i_1,p_2,j_2,i_2) \equiv M(p_1,j_1,i_1)F(p_1,p_2)M(p_2,j_2,i_2).$$

Note that the inner term is the ratio of the bandwidth needed by the service on each link, compared to the available bandwidth on that link. Clearly the relative bandwidth usage will be minimal if the TIs are widely distributed. However, these relative bandwidth usage ratios are summed, so to minimise the sum of ratios, it is advisable to have as few such ratios as possible. Hence the objective function itself tries to balance the competing objectives of wide versus narrow placement strategies. We follow the service model given in Figure 4.2 but it can be generalized for any application that can be decomposed as described above.

4.2.4 Video Streaming Application for Pedestrian Detection

The aim of this example is to emphasize how slow moving, connected vehicles with abundant resources, can be utilized for a specific application, instead of utilizing one vehicle for data collection and then offloading this data to cloud or RSUs for further data processing.

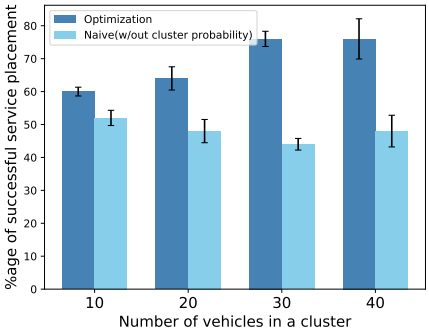


Fig. 4.3 Placement of 5 TIs on different sized cluster.

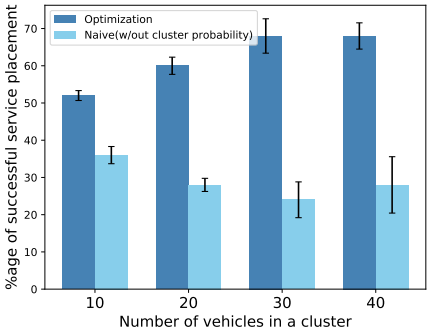


Fig. 4.4 Placement of 7 TIs on different sized cluster.

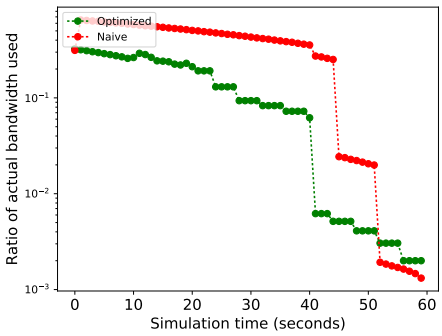


Fig. 4.5 Ratio of bandwidth usage: 5 TIs on cluster of 20 vehicles.

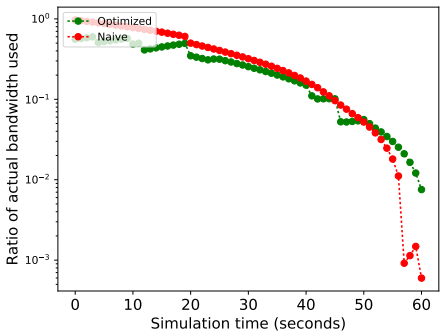


Fig. 4.6 Ratio of bandwidth usage: 7 TIs on cluster of 20 vehicles.

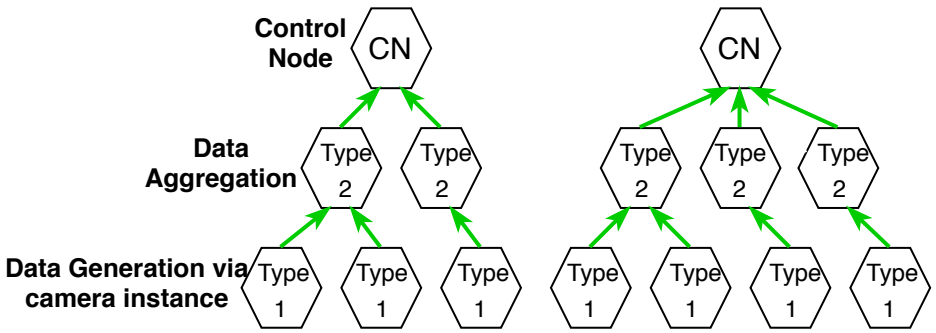


Fig. 4.7 Different service requests with varying levels of SCIs.

Example: We map a video streaming request to selected vehicles in a service cluster based on specific requirements of a service, which in this case is detecting pedestrians using dash/smart cameras in vehicles. The Tasks and dependencies of one such sample video streaming application are defined as a Type Graph as depicted in Figure 2. The cardinality on the instance graph depicts the maximum and minimum number of allowed connections among TIs of a particular type. The *Type 1* TI streams video from selected cameras in vehicles as input to *Type 2* TIs. For maximum coverage, the TIs are anti-collocated to have a wider field of view, and to increase the robustness of the application. The data streams to six *Type 2* TIs, possibly the nearest neighbour nodes with available resources. The *Type 2* TI reduces the video streams to frames with a detected pedestrian, which reduces the data flow to 30-40% of its original size.

The *Type 3* instances can be a decision making entity, which requires more intelligence for application like real-time traffic management, with very low latency requirements. But we focus on a monitoring-type application, where the reduced data flow is routed to the Client, via the CN, for decision making. The goal of the application is to reduce the bandwidth usage by choosing the nearest nodes for in-network processing and minimising data flow volume sent from vehicular network to infrastructure. For our sample pedestrian detection system, the classification task is subdivided in terms of data operations (TIs). Our placement model maps these classification tasks/TIs to nodes in the service-based cluster.

4.2.5 Simulation and Evaluation

The ILP formulation is optimised using the Gurobi solver. Finding a solution for a service with 7 TIs takes an average of 26 minutes for a cluster of 40 vehicles on an Intel i7-6500U running at 2.50 GHz. We evaluate the performance of the resulting ILP solution in a SUMO and Mininet-WiFi [Dos Reis Fontes et al., 2017] simulation to validate our claim that it places TIs on the most suitable vehicle nodes. The simulation scenario is that a number of

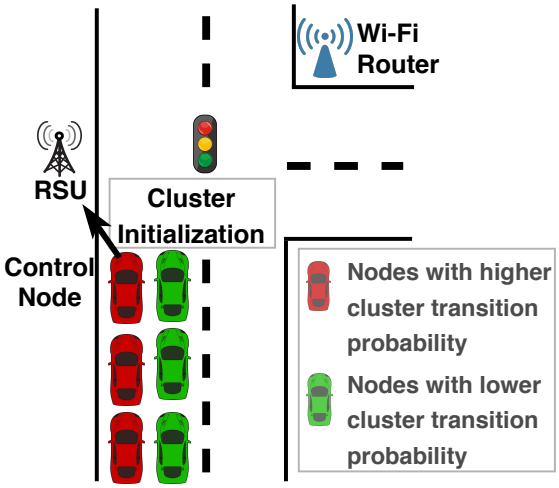


Fig. 4.8 Description of the simulation scenario.

vehicles wait at an intersection and each can either continue straight on or take a right turn, thus leaving the cluster, as depicted in Figure 4.8. A request is sent from the RSU to form an opportunistic cluster and get the transition probability $P_i(t_1,t_2)$ for every vehicle in the cluster from time t_1 to t_2. We assume the transition probability for each vehicle in the cluster is known at each intersection.

The simulation selects a single vehicle node as a CN that is one hop away from the RSU and, at that time t, has the highest transition probability of staying on the intended route. The CN acts as a gateway between the RSU and all the members in the cluster. The RSU sends service requests from the client to the CN, which then selects nodes for a service-based cluster, based on the highest transition probability at time t_1 to stay with the CN at time t_2, s.t. $P(t_1,t_2,\mathbb{I}_o) \geq P_{min}$). The minimum cluster transition probability threshold for selection of the service cluster is chosen as 0.5, to ensure a fairly stable service cluster.

We simulate the same setup for 10, 20, 30 and 40 vehicle nodes and the cluster transition probability is randomly distributed from 0 to 1. The resource types for each vehicle can be small, medium or large, defined in Table 4.3. A cluster is resource-rich if it has 50% large capacity nodes, 25% medium and 25% small capacity nodes. A resource-poor cluster has 25% large, 25% medium and 50% Small capacity nodes. For the evaluation in this paper, we ensure there is enough processing resources by considering only resource-rich clusters.

The cluster mobility and resource state are maintained by the local CN. The nodes that leave the cluster can no longer communicate with the nodes within the cluster, as the flow entries relating to leaving nodes are deleted. In a real life scenario, a vehicle taking the right lane might still be able to stream videos until it has lost connectivity with the CN/access

point. However, from an application point of view, it is important to maintain the minimum required QoS level, which partial flows might not be able to provide.

Successful Service: Once the placement is complete, the service is considered successful when all streaming video data flows are received at tier-2, which process the video, compressing it to 30-40% and sending the reduced data flow from each instance in tier 2, to the CN, which sends the aggregated data to the client. We run the simulation for an application with 5 and 7 instances (as depicted in Figure 4.7) to analyze the performance of the ILP against a naive approach that ignores cluster transition probability.

For the case of 5 TIs (Figure 4.3), the percentage of success is higher, 60% for 10 vehicles, compared to the naive approach. We observe a 50% success for 7 instances. These percentages are calculated by averaging over 25 simulation runs, varying the cluster transition probability and the availability of resources on each node, and hence selecting different CNs. The success percentage can be improved by treating different TIs differently, such that capturing/sensing TIs are anti-collocated (ie. we place a single TI on selected node), as more than one video capturing TI on a single node does not contribute to the service. The processing TIs (of *Type* 2,3) can be collocated (packed on the same node, depending on node and link capacity) to reduce the delay in data processing and to use network resources efficiently. The anti-collocation constraint reduces the available nodes to 10, for 5 and 7 instances, but can be increased by collocating TIs on fewer nodes. As the resources increase, and the cluster gets more dense, the percentage of success increases to 80% whereas the success in the naive approach is lower and does not depend on the number of vehicles or the infrastructure capacity. For the case of 7 instances(Figure 4.4), we get significantly lower successful service completion for the naive approach, and up to 70% successful service completion for a cluster of 30 and 40 vehicles.

To measure network resource usage, the simulation calculates the cumulative transferred bytes/sec between nodes in the service cluster. We compare it to the total available bandwidth capacity of the network at each instance, to compute the ratio of actual bandwidth usage. The bandwidth used in the naive approach is significantly higher than our approach for the case of 5 TIs (Figure 4.5). This is attributed to selecting the node by using the heuristic of nearest hop distance for placing TIs, thereby reducing overall bandwidth usage. For the case of 7 TIs (Figure 4.6), higher successful service completion requires significantly higher bandwidth but still performs better than the naive approach. The bandwidth usage is calculated for a single case of 20 vehicles in a cluster. For all cases, we observe decreasing bandwidth usage over simulation time, and a significant decrease in usage around 40th second of simulation due to the data flow reduction at *Type 2* TI. A *successful service* implies a stable cluster for

both optimized and naive scenarios, thus, we see that the less constrained, naive approach performs better in the last 10 seconds of the simulation.

4.3 Service Scaling and Placement Scheme with Multi-hop Clusters and Detailed Service Model

The initial infrastructure and service model introduced in this chapter did not consider multi-hop routing, which is necessary to model a realistic vehicle cluster which is equipped to host distributed services on them. We now introduce multi-hop routing as an incremental improvement to the model. We also introduce detailed service model constraints to ensure that the data flow collected by Type 1 TIs is processed before reaching the CN. The constraints also ensure that the order of TIs is maintained according to the service chain description in the *Type graph*. This model also considers real vehicle density data to include the mobility patterns of vehicles as an intrinsic part of the infrastructure model.

4.3.1 Infrastructure Constraints

The infrastructure constraints considered in the ILP are presented below:

4.3.1.1 Bandwidth Constraints for the multi-hop cluster

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 4.10.

$$\forall i_1 \in \{1,\ldots,\mathbb{I}\}; i_2 \in \{1,\ldots,\mathbb{I}\}; i_1 \neq i_2 \sum_{\forall p_1,j_1;p_2,j_2;p_1\neq p_2} M(p_1,j_1,i_1)F(s_{p_1},s_{p_2})M(p_2,j_2,i_2) \leq B(i_1,i_2) \quad (4.10)$$

where $i_1 \neq i_2$ and $B(i_1,i_2) \neq 0$. Constraint 4.10 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 4.11 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:

$$\forall i_1 \in \{1,\ldots,\mathbb{I}'\}; i_2 \in \{1,\ldots,\mathbb{I}'\}; i_1 \neq i_2 \sum_{\forall p_1,j_1;p_2,j_2;p_1 \neq p_2} m(p_1,p_2,j_1,i_1)F(s_{p_1},s_{p_2})m(p_1,p_2,j_1,i_2) \leq B(i_1,i_2) \tag{4.11}$$

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 $s_{p_2 j}$.

4.3.2 Distributed 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.

4.3.2.1 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 4.12 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:

$$\forall i_1 \in \{1,\ldots,\mathbb{I}\}; i_2 \in \{1,\ldots,\mathbb{I}\}; i_1 \neq i_2 \sum_{\forall p_1,j_1;p_2,j_2;p_1 \neq p_2} m(p_1,p_2,j_1,i_1)C(F'(s_{p_1 j}, s_{p_2 j})) \leq C(s_{p_2 j}) \tag{4.12}$$

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).

4.3.2.2 Flow Conservation Constraint

Constraint 4.13 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:

$$\forall i_1 \in \{1,\ldots,\mathbb{I}\}; i_2 \in \{1,\ldots,\mathbb{I}\}; i_1 \neq i_2 \sum_{\forall p_1,j_1;p_2,j_2;p_1\neq p_2} F(s_{p_1 j}, s_{p_2 j})\alpha_{p_2 j} \leq F'(s_{p_1 j}, s_{p_2 j}) \tag{4.13}$$

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].

4.3.2.3 Task Order Constraints

Constraint 4.14 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.,

$$\forall i_1 \in \{1,\ldots,\mathbb{I}\}; i_2 \in \{1,\ldots,\mathbb{I}\}; i_1 \neq i_2 \sum_{\forall p,j;p+1,j_2;p\neq p+1} M(p,j,i_1)F'(s_{p_1 j}, s_{p_2 j}) \geq M(p+1,j,i_2) \tag{4.14}$$

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 4.10 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 4.15:

$$\forall i_1 \in \{1,\ldots,\mathbb{I}\}; i_2 \in \{1,\ldots,\mathbb{I}'\}; i_1 \neq i_2 \sum_{\forall p,j;p+1,j_2;p\neq p+1} m(p,p+1,j,i_1)F'(s_{p_1 j}, s_{p_2 j}) \geq m(p,p+1,j,i_2) \tag{4.15}$$

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.

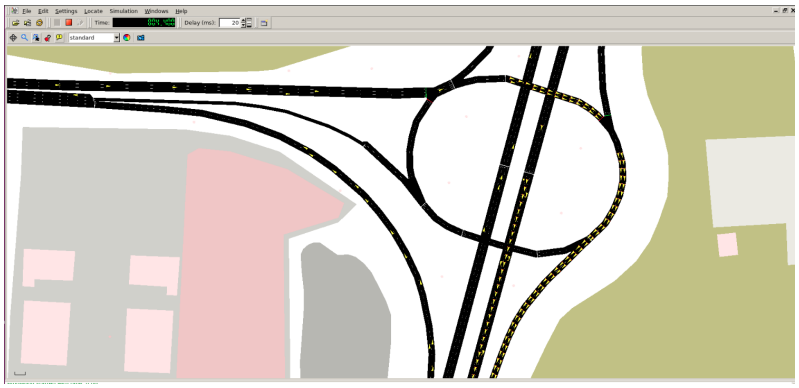


Fig. 4.9 The SUMO simulation for the intersection in Dublin, Ireland.

4.3.3 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 confidence score. As the history of a given vehicle builds up with time, its road exit probabilities are updated and its 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 §3.2.1. For simulations, we extract the Dublin intersection road network, as depicted in Fig. 4.9, 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 behaviour 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 [Maglaras and Katsaros, 2016], 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. 3.1), 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.

4.3.4 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:

$$\text{NodeCost}(i_1) = \sum_{\forall i_1,i_2;i_1\neq i_2} (1 - P_{(t_1,t_2)}(i_1)).(D_{pj,k}/C_k(i)).M(p,j,i) \tag{4.16}$$

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:

$$\begin{aligned}\text{LinkCost}(i_1,i_2) = \sum_{\forall i_1,i_2;i_1\neq i_2} (1-P_{(t_1,t_2)}(i_1,i_2)).\\ (F(p_1,p_2)/B(i_1,i_2))\Big(m(p,p+1,j_1,i_1).m(p,p+1,j_2,i_2)\\ +M(p,j_1,i_1).M(p,j_2,i_2)\Big)\end{aligned} \tag{4.17}$$

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 $s_{p_1 j}$ 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 $s_{p_1 j}$ at node i_1 and the other TI $s_{p_2 j}$ 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 $(F(s_{p_1 j}, s_{p_2 j}))$ with the available bandwidth $(B(i_1,i_2))$ at each link that forms part of the service placement.

4.3.5 Bi-objective Optimisation Function

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

4.3.5.1 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 minimised for efficient service placement. The hop count between two placed TIs is minimised when:

$$\forall i_1 \in \{1,\ldots,\mathbb{I}'_{(p_1 j)(p_2 j)}\}; i_2 \in \{1,\ldots,\mathbb{I}'_{(p_1 j)(p_2 j)}\}; i_1 \neq i_2 H(i_1,i_2) = \sum_{\forall p_1,p_2} m(p_1,p_2,j,i), \min\ H(i_1,i_2)), \tag{4.18}$$

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.

4.3.5.2 Total Cost of Service Placement

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

$$\min \sum_{\forall i_1, i_2; i_1 \neq i_2} \lambda_1 \text{LinkCost}(i_1, i_2) + \lambda_2 \text{NodeCost}(i_1) \tag{4.19}$$

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 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 optimisation takes care of the number of TIs deployed, with the objective of minimising total cost. Thus, as depicted in Fig. 4.10, 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 optimisation, the first objective function effectively gets a higher priority than the next. We give an explanation of this choice in Section 4.4.2. Thus, minimising the hop counts is the first priority of the optimisation and then equal priorities are given to both node and link cost.

4.4 Results for Service Placement for the Bi-objective Service Placement Problem

4.4.1 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 [Ang et al., 2019]: 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

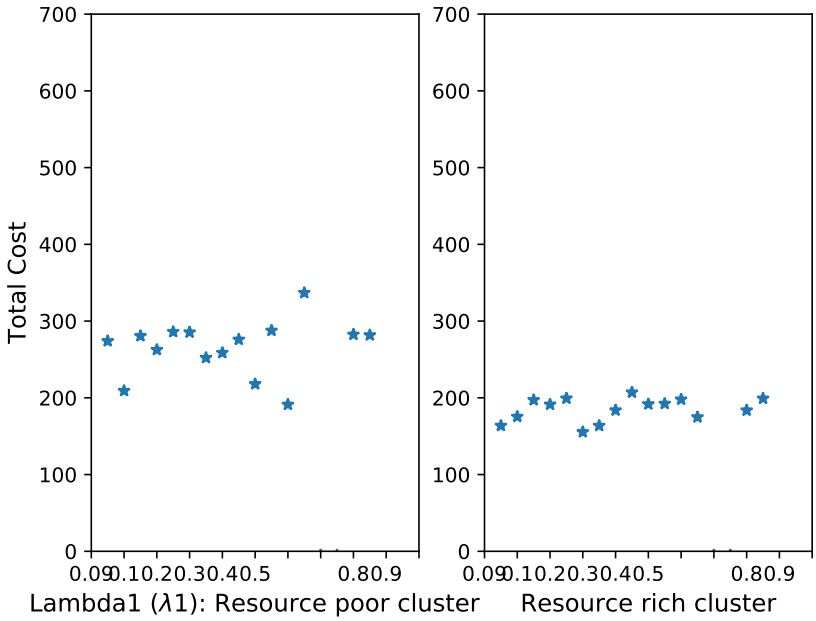


Fig. 4.10 Sensitivity analysis for resource-rich and resource-poor clusters, we get total cost values in a similar range for almost every weight. We chose both λ_1 and $\lambda_2 = 0.5$, as it gives lower cost in both cases and other values do not significantly affect cost.

relevant for a Fog computing scenario, where the computation and storage capacity is limited.

- Data Reliability: Other applications 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:

4.4.1.1 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

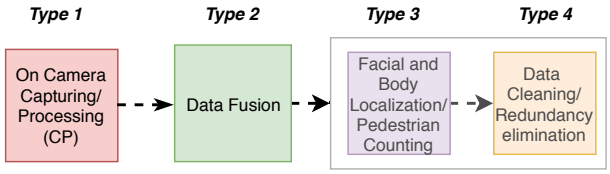


Fig. 4.11 Sample application for pedestrian detection

hence, most of this data should be processed locally, based on the available resources on the vehicle cluster. For this application, 1 to 6 camera or *Type 1* TIs are used in the Evaluation section (4.4.2), 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. 4.11. 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 these applications on an existing simulator called Yet Another Fog Simulator (YAFS) [Lera et al., 2019]. 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 the 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. 4.12, we consider the average node utilisation in the placement of service described as Application I. As suggested by the authors in [Lera et al., 2019], we calculate the node utilisation as the sum of the service times at each node divided by the total simulation time. We compare the average node utilisation between:

- Cloud placement: all tasks placed in Cloud
- Edge/RSU placement: all tasks placed on edge/RSU

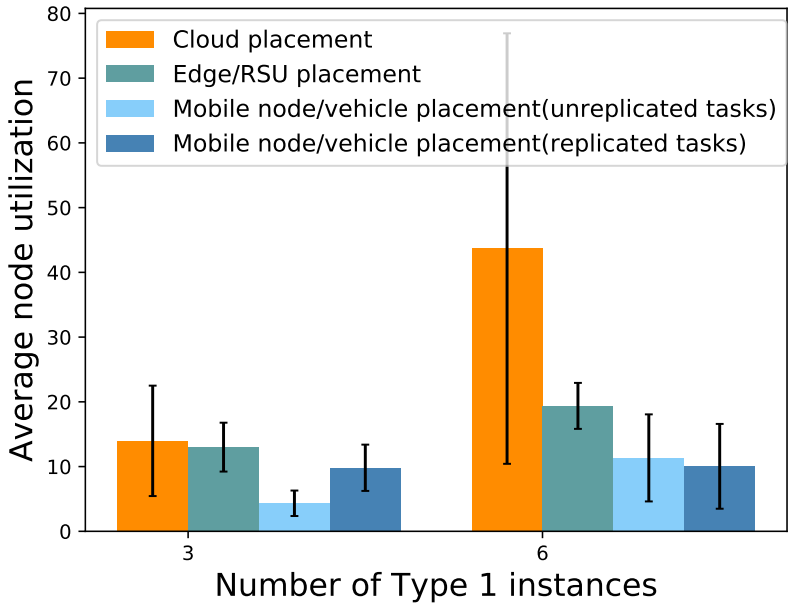


Fig. 4.12 Comparing different placement techniques using the average node utilisation in the case of cloud placement, edge/RSU placement, mobile node/vehicle placement (without replicating task instances), mobile node/vehicle placement (with replicated task instances: **our case**) for Application I.

- 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. 4.12, 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 results in better node utilisation, 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 utilisation compared to all the other approaches. This validates that our service model of replicating tasks is efficient from a node utilisation point-of-view, as compared to other placement approaches.

4.4.1.2 Application II: Low-processing video streaming application

Application II uses vehicles as moving sensors for video collection. Applications of this category include 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.

4.4.2 Evaluation

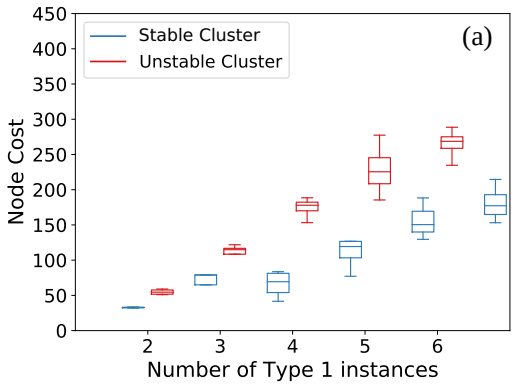
We solve the constrained optimisation problem using the Gurobi Optimiser, which is a powerful mathematical solver, on an Intel i7-6500U dual-core processor running at 2.50 GHz. The solver uses a Linear Programming (LP) based branch and bound algorithm to solve the Mixed Integer Programming (MIP) problem.

We place Applications I and II together on a vehicle cluster with 10 nodes, since more nodes in a cluster increase 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 optimisation for the longer chain length, which takes a much longer time to find a solution, especially for a higher number of video generating instances, with a 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.

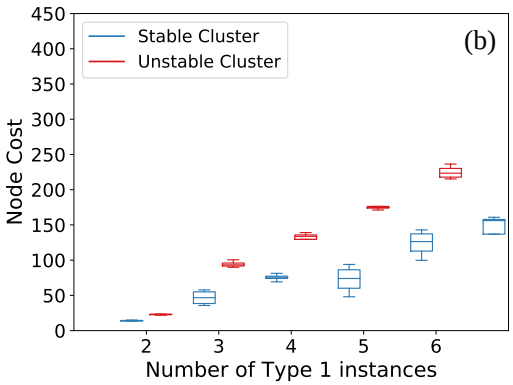
The 'type graph' is scaled as an 'instance graph', with data dependency and resource requirements. We use a service chain description similar to [Dräxler et al., 2018], without making it bidirectional. We impose multi-tenancy in the model, as it is beneficial to share TIs between applications, especially when more than one task replica is placed on the vehicle cluster.

For this chapter, 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

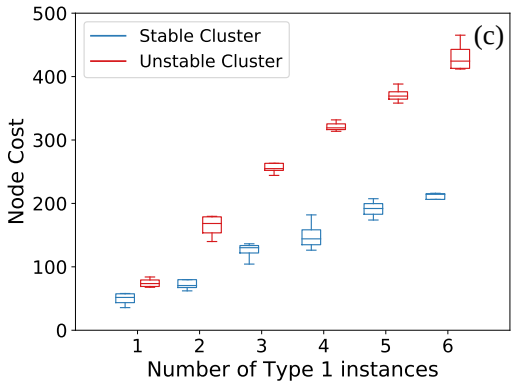
mobility behaviour, in terms of the CCP is based on the mobility pattern of each vehicle, collected over its previous trips in this area. We derive the CCP by running the calibrated SUMO simulator, using the real vehicle density data from Dublin traffic, as explained in §4.3.3. We have broadly classified 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].



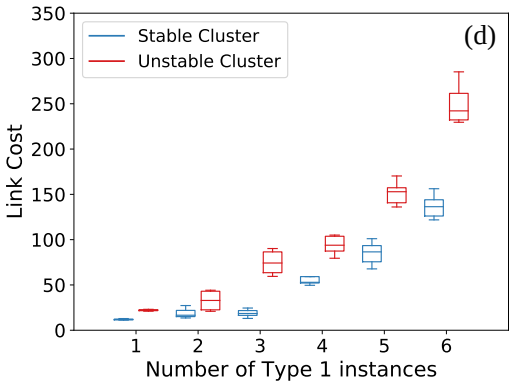
(a) **Case A:Node cost for resource-constrained cluster and low data rate**



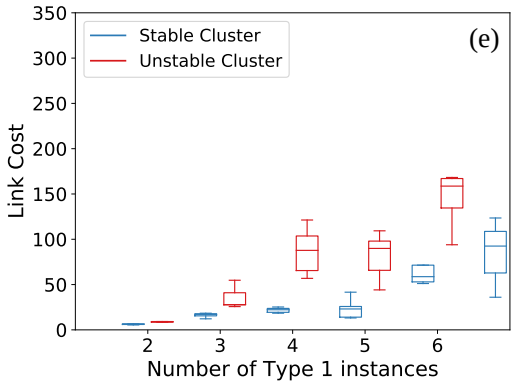
(b) **Case B:Node cost for resource-rich cluster and low data rate**



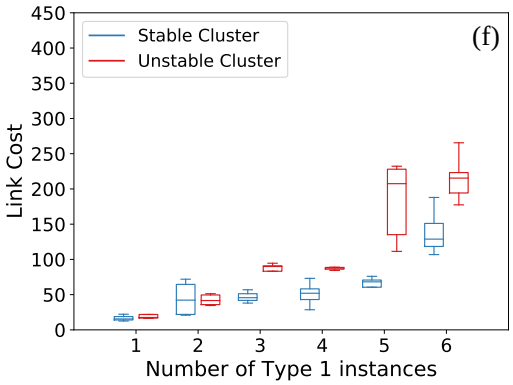
(c) **Case C:Node cost for resource-rich cluster and high data rate**



(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. 4.13 Node and Link Cost for Case A,B & C.

Table 4.1 Table of Notation.

Notation	Representation
s_p	Task of *type* p
s_{pj}	Task instance of *type* p and count j
$\mathbb{I}$	No. of nodes available for the formation of vehicle cluster
$i \in V$	i is the index of a vehicular node with resources to be used by the task instances to be placed on this node total set of V nodes
k	One of $\mathbb{K}$ types of resources: CPU, memory, sensing
$C_k(i)$	Available capacity of each vehicle node i for resource type k
$(i_1, i_2) \in E$	directed edge of the graph represents the link between any two vehicle nodes i_1 and i_2
$B(i_1, i_2)$	Link capacity between two *nodes* i_1 and i_2
$P_{(t_1,t_2)}(i_1)$	Cluster Cohesion Probability of a single vehicle node i_1 from time t_1 to t_2
$P_{(t_1,t_2)}(i_1, i_2)$	Joint Cluster Cohesion Probability of two nodes i_1 and i_2 to stay together from time interval t_1 to t_2
$\mathbb{N}_{s_p}$	Number of task instances for a task s_p
$\mathbb{S}$	Set of all SCs
D_{pjk}	minimum demanded amount of D_{pjk} units of resource type k for task instance s_{pj}
$F(p_1, p_2)$	Flow demand between task s_{p_1} and s_{p_2}
$C(F(s_{p_1 j}, s_{p_2 j}))$	Processing requirement for the flow from task instance $s_{p_1 j}$ to $s_{p_2 j}$
$M(p, j, i)$	$\begin{cases} 1, & \text{if the task instance } s_{pj}, \\ & \text{is placed at (mapped to) node } i \\ 0, & \text{otherwise} \end{cases}$
$m(p_1, p_2, j, i)$	$\begin{cases} 1, & \text{if the node i is mapped to be a} \\ & \text{forwarding node for data flow} \\ & \text{between instance } s_{p_1 j} \text{ to } s_{p_2 j} \\ 0, & \text{otherwise} \end{cases}$
$\mathbb{I}'_{(p_1 j)(p_2 j)}$	is the set of all nodes on the path between task instances $s_{p_1 j}$ and $s_{p_2 j}$
$C(s_{p_2 j})$	is the processing capacity of task instance $s_{p_2 j}$
$F'(s_{p_1 j}, s_{p_2, j})$	represents the flow that has been processed at task instance $s_{p_1 j}$ or is forwarded from $s_{p_1 j}$ specifically for processing (not forwarding).
α_{pj}	represents the data reduction/processing factor for a task with *Type p* resource.
$H(i_1, i_2)$	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}

Table 4.2 Simulation parameters.

Parameters	*Range*
Node Density	10-50
Service Cluster P_{min} Threshold	0.5
Vehicle Node Resource Profiles	Small(S),Medium(M),Large(L)
Resource Profile Mix	50% L, 25% M, 25% S
Number of Application Instances	5, 7

Table 4.3 Resource profiles by size.

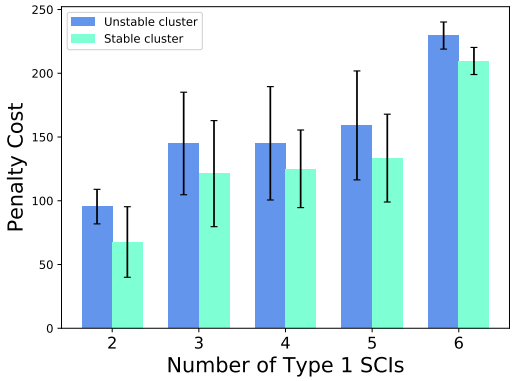
Resource Profile	*vCPU*	*vDisk* Mb	*vBandwidth* Mb/s
Large	5	500	6
Medium	3	250	4
Small	2	100	2

We consider three use cases for solving the optimisation. 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 fewer resources than the unstable cluster. For Case B, a resource-rich case (Fig. 4.13b) with lower data rates, the node cost is significantly less, compared to Case A (Fig. 4.13a), 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 utilisation. But in this case, the stable cluster still used less resources 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.

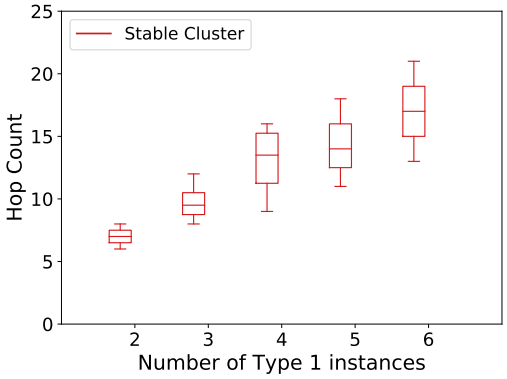
For link cost, the resource-constrained cluster (Case A) has significantly higher resource usage (Fig. 4.13d). The nearby nodes might not have enough processing capacity, so dependent TIs need to be placed on farther nodes, leading to more link utilization. The link capacity is also less in the resource-constrained cluster which adds to the cost. The Link Cost in the resource-rich cluster (Case B) (Fig. 4.13e) 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 rate of the video data in a resource-rich cluster (Case C), the unstable cluster utilises much more computation resources (Fig. 4.13c). The difference between stable and unstable cluster node costs 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. 4.13b). For the link cost in this case (Fig. 4.13f), for fewer Type 1 TIs, stable and unstable clusters incur almost the same cost. The variability increases as the number of video instances increases.

4.4.3 Penalty Function

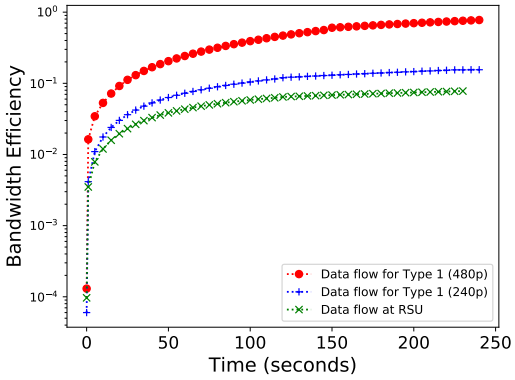
We introduce a Penalty function to reflect the cost of nodes hosting TIs leaving the cluster. This penalty is related to both link and node costs. If a TI-hosting node leaves the cluster, the node cost incurred by it does not contribute to the service requirement. In case of



(a) Penalty cost.



(b) Total hop count.



(c) Bandwidth efficiency at Type 1 TIs and RSU.

Fig. 4.14 Penalty cost, total hope count and bandwidth efficiency at Type TIs and RSU.

service reconfiguration, the leaving node will have to send the service state back to the CN, which requires bandwidth usage. We penalise placements where TIs are placed on nodes with a probability lower than the threshold probability (0.6). There will be another added cost on re-configuring the failing service, but we do not consider service reconfiguration costs in our current model. The Penalty Function for placing a TI on a node with a Probability less than a pre-decided threshold probability $(P_{Threshold})$ is given as:

$$\begin{aligned} P_{(t_1,t_2)}(i_1) &\leq P_{Threshold} \\ (P_{(t_1,t_2)} - P_{Threshold}) &\leq 0 \\ -P_{(t_1,t_2)}(i_1) &\leq -P_{Threshold} \end{aligned}$$

The Penalty function for selecting two nodes with a joint probability (to stay with the cluster) less than the threshold probability is given as:

$$P(\lambda_{(i_1,i_2)}) = \lambda(P_{(t_1,t_2)}(i_1,i_2) - P_{Threshold}) \tag{4.20}$$

where λ can take values like 5,10,..,100, based on how strongly we want to penalise task placement on nodes with lower CCP. As shown in Fig. 4.14a, for a resource-constrained cluster (Case A), the penalty cost added to the objective function for an unstable cluster is higher than the penalty for a stable cluster. For the resource-rich cluster (Case B), the model places very few TIs on nodes with low CCP, and hence the additional penalty cost is zero, in almost all the resource-rich test cases, for both stable and unstable clusters. This suggests that the model considers mobility in an intuitive way. We also plot the best and worst case hop count (Fig. 4.14b), for the same resource-constrained cluster (Case A), when the total hop count is first minimised for service selection. It is observed that the hops increase significantly as the number of Type 1 TIs increases.

4.4.4 Mininet-WiFi Simulation

We emulate the scenario of a vehicle cluster with 10 nodes for Application II. We simulate two video streaming nodes that send 4 minutes long videos in different resolutions (240p and 480p) and at different data rates. We use a SUMO-Mininet-WiFi [Dos Reis Fontes et al., 2017] set-up, where the RSU receives a service monitoring request at the same intersection in Dublin and forwards the request to the CN. We use the SUMO simulator to generate urban mobility at a busy intersection. Mininet-WiFi maps the cars to an emulated, software-defined network with virtualized WiFi stations and access points. The video stream is forwarded to a participating node that aggregates the data before sending it to a CN that forwards it to the RSU. We monitor the bandwidth usage for the two collected streams and compare it to the bandwidth efficiency of the processed stream received at the RSU. As depicted in Fig. 4.14c, we get significantly better bandwidth efficiency at the RSU because of the data aggregation in the model.

4.5 Service Scaling and Placement Plan Procedure

In this section, we explain our procedure for service scaling and placement. As presented in Algorithm 4.1, 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 4.1 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 i1,i2 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(i1,i2), Cost = Service Placement()
7:              else if Unplaced then
8:                  Scale new TI of Type n
9:                  H(i1,i2), Cost = Service Placement()
10:             if Type n is placed then
11:                 total_hop += H(i1,i2)
12:                 total_cost += Cost
13:             if Type n to Type m placed then
14:                 if H(i1,i2) ≤ min_hop && Total Cost ≤ 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(Pi(t1,t2), Ck(i), B(i1,i2), Dpjk, C(F(sp1 j,sp2 j)), F(sp1,sp2), C(sp2 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 i2
24:             Calculate H(i1,i2)
25:             Calculate Cost = λ1 Link Cost(i1,i2) +
26:             λ2 NodeCost(i1)
27:             return Cost,H(i1,i2)
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 minimum total cost of an existing service placement plan to find the placement with the lowest objective value in lines 13 to 19.

4.5.1 Comparison of MIP with baseline approaches

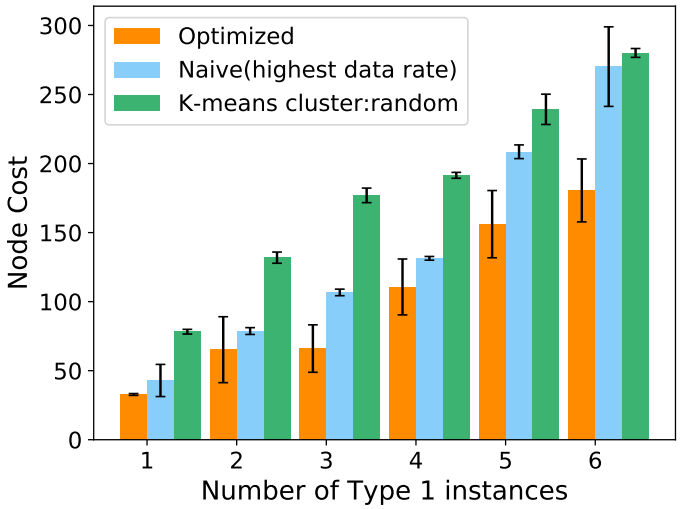


Fig. 4.15 Comparison of node cost in the optimal, naive scheme and a clustering scheme for Case A:resource-constrained cluster and low data rate.

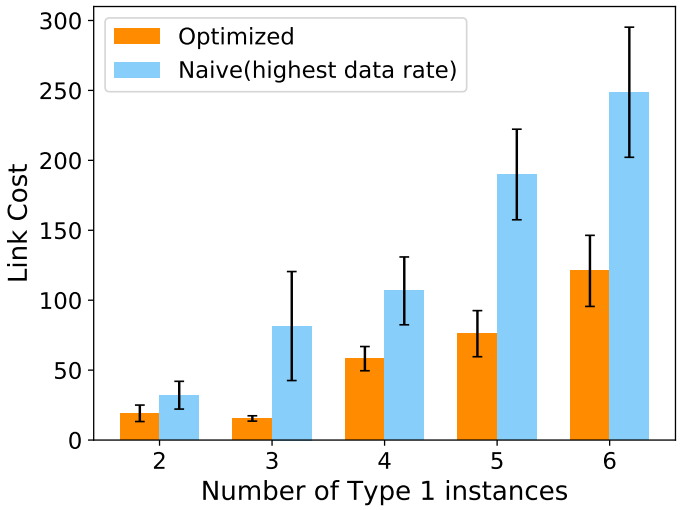


Fig. 4.16 Comparison of link cost in optimal v/s naive scheme for Case A:resource-constrained cluster and low data rate.

We compare the optimal solution of our model to an autonomous vehicular edge computing-based naive solution introduced in [Feng et al., 2017] and a clustering-based solution introduced in [Hu et al., 2021]. The work in [Feng et al., 2017] 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 minimising 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 [Zhu et al., 2019], they preferentially select nodes with the highest available link and node capacities.

The clustering-based approach in [Hu et al., 2021], 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. As depicted in Algorithm 4.2, the mobility and resource state of the cluster are collected from all participating vehicle nodes (lines 2-4). The first centroid is selected randomly, the remaining k-1 centroids are selected based on the maximum squared distance from the nearest centroid (lines 5-9). The classical k-means clustering is used to form k clusters (line 10). 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 [Hu et al., 2021] only considers the processing cost, and hence, we only use the node cost to compare their clustering approach to other approaches.

Algorithm 4.2 KMeans Clustering

Input: Mobility and resource state of vehicles: M$[t_1, t_2]$ = <vehicle_speed, CCP, $C_i(k)$ >
Output: Selected vehicle cluster

1: **procedure** CLUSTER FORMATION ▷ Using K-Means Cluster
2: **for** each vehicle node i **do**
3: Collect $M_i[t_1, t_2]$
4: Collect dataset of all available nodes $\mathbb{I}$ $M[t_1, t_2] = [M_1, M_2 ... M_{\mathbb{I}}]$
5: **for** j from 2 to k **do**
6: **for** each vehicle i **do**
7: Calculate $d_i = \max ||x_i - C_j||^2$
8: u_j = argmax(d_i)
9: Get k initialized cluster centers: $u = [u_1, u_2, \ldots u_k]$
10: Clusters = KMeans(u)
11: **for** each cluster in Clusters **do**
12: $ICD_k = \sum(||d_i, Cent_k||^2)$
13: Selected cluster = min(ICD_k)

As seen in Fig. 4.15, 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. 4.16 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 does not take account of node mobility, so is more likely to fail. By contrast, our objective reduces the cost of service execution *and* select 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.

4.6 Conclusion

This chapter 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 utilising 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 vehicle cluster. We introduced two detailed mathematical models for the mobility-aware scaling and placement of distributed services based on resource-rich and resource-poor as well as stable and unstable cluster states. The first model introduced in Section 4.2, uses single-hop communication between different vehicles in the cluster and uses constraints like anti-collocation to distribute placed TIs on different nodes to ensure a resilient service placement. Similarly constraints like adjacency and cohesiveness ensure that the nodes with placed TIs are not very far from each other in-terms of network distances. Thus, this service placement model ensures distributed service placement without increasing the communication resources available in the vehicle cluster. This service placement aims to find an optimal service placement plan ensuring full deployment of the distributed service. In Section 4.2.5, the service placement plan is evaluated using the percentage of successful service placements and the ratio of bandwidth usage for the placement.

In Section 4.3, a more realistic service model and system model is introduced. Different from the model introduced in Section 4.2, this service placement model considers multi-hop clusters. This service placement model introduces service level constraints that ensure the TI's are placed in order and the data flows are processed at each TI mentioned in the service model. This service placement model optimises the service placement cost which is

Table 4.4 Comparison of two service placement schemes introduced in the chapter.

Service placement plan with single-hop clusters (Section 4.2)	*Service placement plan with multi-hop clusters (Section 4.3)*
The model considers single-hop clusters and focuses on anti-collocation, adjacency, full deployment type constraints. The service placement focuses on a balanced service placement in terms of distribution of tasks without over-utilising available resources.	This model considers multi-hop clusters. This service placement model considers mobility as an an intrinsic part of the model by minimising the resource, weighted by the CCP of the vehicular node. Thus, the model does not only consider available resources but also the availability of the resources based on their historic mobility pattern.
The model does not consider service-level constraints.	This model considers more sophisticated service-level constraints that ensure the TI's are placed in order and the data flows are processed at each TI mentioned in the service model. Such a model is crucial for deploying futuristic services that are deployed in a distributed manner and require real-time scaling.

defined as resource utilisation weighted by the CCP. The two service placement schemes are compared in Table 4.4.

In Section 4.4, first two applications types are introduced, one with low processing requirements and the other with high processing requirements and both type of applications are placed on the vehicle cluster to promote multi-tenancy in the model. Then, very detailed simulation results are presented for the introduced placement scheme by considering different mobility states of the vehicle cluster including stable and unstable vehicle clusters based on the CCP of the participating vehicles. The experiments also consider different resource profile of vehicle clusters classified as "Small", "Medium" and "Large" vehicle node types. The two types of applications are placed simultaneous to evaluate the performance of the service model in terms of node and link costs. In Section 4.5.1, our approach is compared to a naive solution introduced in [Feng et al., 2017] and the clustering-based approach introduced in [Hu et al., 2021] significantly. Our approach outperforms both the baseline approaches in terms of both node and link cost.

Chapter 5

Placement of Distributed Video Processing Applications on Moving Vehicle Clusters

5.1 Introduction

The increasing amount of data generated from Internet of Things (IoT) devices has resulted in isolated sources of data that are not fused with other data sources and hence are not fully utilised. On the other hand, the emergence of powerful machine learning and deep learning algorithms requires large amounts of data to make accurate inferences. Most of the surveillance carried out on roads or other public places is through static cameras that send most of the collected data to the back-end server [Rathore et al., 2021]. This approach leads to increasing operational costs in deploying dedicated hardware as well as using expensive bandwidth to send this data to the cloud through dedicated links.

The concept of vehicular fog computing (VFC) [Ning et al., 2019] is derived from using the available and under-utilised vehicular resources, like in-built sensors, processors, dashboard cameras, advanced onboard units (OBUs), etc., for both crowdsensing and processing data. The VFC paradigm aims to make computation more efficient by leveraging the processing and communication capacity of the vehicles, instead of offloading computation to the edge servers or the cloud [Lee and Lee, 2020]. VFC extends the intelligence of mobile edge computing [Zhao et al., 2019], [Chen and Hao, 2018] and fog computing [Yousefpour et al., 2019] to the vehicular network, in an attempt to increase the processing capacity to meet the resource requirements for vehicular and infotainment applications. This novel system of distributed service deployment reduces the traffic at the core network, saves the network

bandwidth in sending local and contextual data to the cloud, and also reduces end-to-end latency [Yadav et al., 2020], [Zhu et al., 2018].

VFC also reduces the need for installing infrastructure to facilitate Intelligent Transport Systems, for improving vehicular flows and reducing congestion, for recording data for road condition monitoring to detect potholes, accidents, etc., and increasing commuter safety, which has been theorized for more than a decade but has not been implemented [Thakur and Malekian, 2019]. The use of Roadside Units (RSUs) or edge servers for meeting the demands of vehicular applications has also been explored, but it has limitations due to the mobility of vehicles. These RSUs have limited coverage on the highways and have limited sojourn time with moving cars. Thus, provisioning services on RSUs can increase the control cost of service migration between different RSUs. On the other hand, the OBUs on self-driving cars have evolved in their processing capability and their ability to communicate with neighboring vehicles and keep open connections with edge or cloud servers. These vehicles also have a full System on a chip (SOC), for example, a Tesla car has 4 LPDDR4 RAM chips, with complete redundancy to have failure resistance for any system on board [MEOLA, 2020].

Many recent studies on VFC have focused on latency-sensitive applications that are safety-critical [Ho et al., 2020], [Du et al., 2020]. In this study, we look at a novel use-case of making opportunistic vehicle clusters in urban sections of the city and leveraging the resources on these vehicles to collect and process data. Most of the existing works on VFC either use very simplistic mobility models [Zhou et al., 2020], like considering a straight road segment with constant speed or consider only parked vehicles [Lee and Lee, 2020]. Many works also consider taxis and buses as potential fog nodes as their trajectories are more predictable [Ge et al., 2020]. We, however, want to focus on using the embedded sensors on any vehicle whose owner is willing to contribute the data. Another problem is that many existing works on VFC consider a static and composite service template, which is not suitable for the dynamic vehicular environment. We introduce distributed and flexible services that can be adapted according to the resource requirement, are suitable for the heterogeneous and distributed resources on vehicles, and can be reconfigured easily. Instead of optimizing only computation or communication resources, we jointly optimize both link and processing costs. To select vehicular nodes that are more probable to stay together, we introduce a vehicular node selection scheme and a mobility-aware service placement heuristic.

For this system to operate properly, we first introduce a distributed, graph-based service model where each component or task can be scaled to multiple task instances based on the amount of data collected. We also leverage the mobility pattern of vehicles, stuck in high density/congested traffic to estimate the ongoing availability of these vehicles to perform tasks. The distributed services we propose are scaled in real-time and deployed on the vehicle

cluster such that we get a robust initial placement with less need to reconfigure services. Our model can support many applications using crowdsourced data from opportunistic clusters including sensing applications like pedestrian detection to understand human engagement with coffee shops, gas stations, and other locations. The collected data can also be used to detect congestion and study usage patterns of roads, as part of building *Smarter* cities[Kanaka Sri Shalini et al., 2019].

In this chapter, the following contributions are made to introduce a distributed and scalable service model that can be effectively placed on a group of closely moving vehicles by leveraging their historic mobility patterns:

- A mathematical formulation leveraging the mobility-awareness of infrastructure and a novel and distributed service model is introduced. The scaling and placement of the services are modelled as a bi-objective, constrained optimization problem with the objective being efficient utilization of communication and computation resources.

- Instead of focusing on widely researched latency-sensitive applications, we introduce a novel use case of initiating opportunistic clusters to collect and process data using just macroscopic vehicular density data. We study how traffic reaches congestion levels at different occupancy rates and other traffic patterns. We then calibrate our microscopic mobility model using the real vehicle density data. The mobility model is built from our extensive work on studying the predictability of vehicular flows and estimating the computation and communication capacity of vehicle clusters in our previous work [Sharma et al., 2021].

- We introduce a service model where each application is made of interrelated tasks. Each task can also be scaled to multiple task instances to increase resilience in the service in case of node or link failures. Two such applications are profiled in this chapter to understand the resource usage of such applications.

- We leverage a community-detection-based node selection scheme to study the collective availability of vehicular nodes in a cluster. We then introduce an effective graph-theory-based heuristic that promotes placing task instances optimally within the vehicle clusters. Our approach outperforms an integer linear program solution, and a baseline, first-fit approach. Our approach also results in reduced service latency compared to mobile edge computing-based placement.

The chapter is organised as follows: section §5.2 presents the system model for selecting a well-connected cluster and managing the placement of the services on the vehicular cluster.

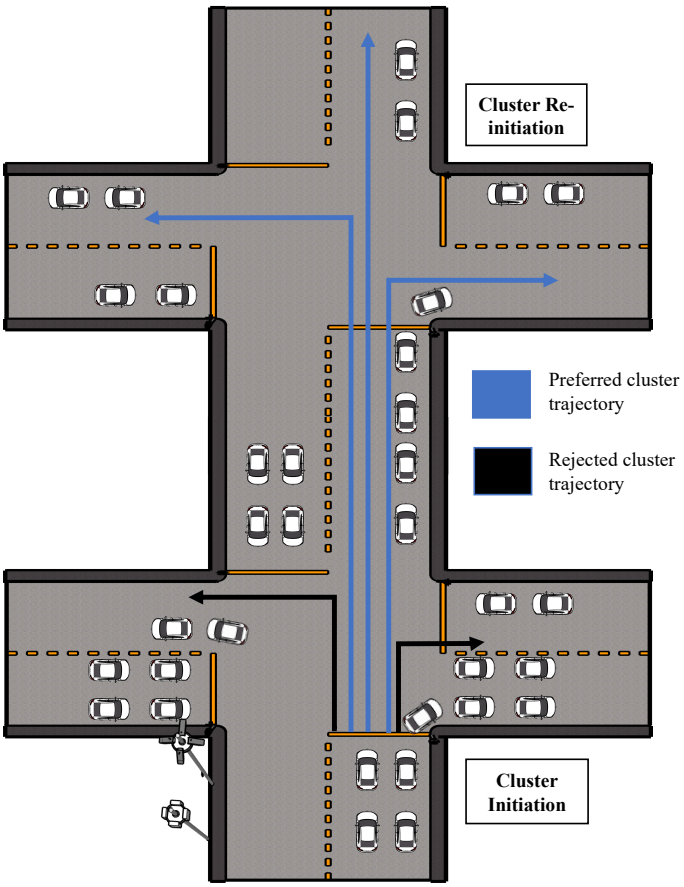


Fig. 5.1 Vehicle cluster's lifecycle. The figure represents the preferred and rejected trajectories of vehicles.

§5.3 introduces two distributed application types considered in this work. We then give details on the network topology and distributed service model along with the notations used for the mathematical modelling. Section §5.4 covers the service scaling and placement constraints and the infrastructure constraints. The section also details the mobility model and the objective function of the constrained optimisation problem. Section §5.5 describes the community-detection-based node selection for cluster formation and the graph-based service placement heuristics. Section §5.6 has a detailed evaluation of the introduced technique compared to the ILP and first-fit solution. We also show the performance of our schemes through service time and the state of the selected nodes in a cluster over time. The chapter is concluded in section §5.7 where future work has also been suggested.

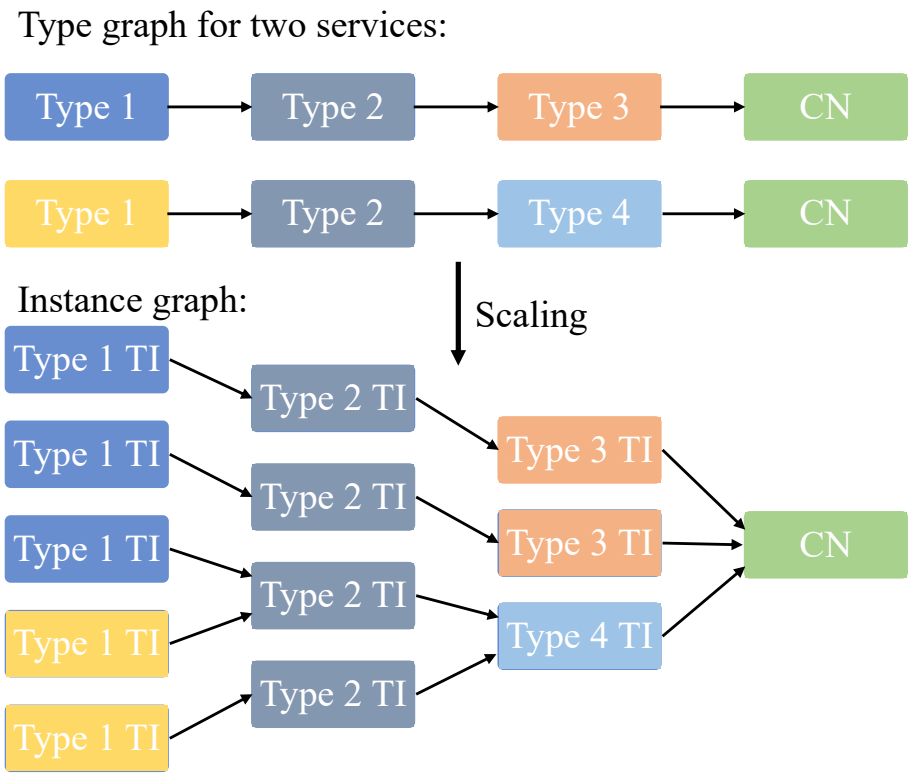


Fig. 5.2 Type graph of two services are presented. The service model depicting tasks and their inter-dependencies. The colour coding represents the scaling of a task to its corresponding TIs.

5.2 System Model

This section details the proposed model for selecting a well-connected cluster and managing the placement of the services on the vehicle cluster. This placement is managed with the coordination of the cluster's CN and the RSUs. The RSU receives requests from clients to deploy services. The client could be a one-off vehicle node moving along with the cluster or a surveillance request from traffic authorities. In this chapter, we assume that the service request is received and decomposed by the RSU in the form of a linear chain of tasks. As depicted in Fig.5.1, the RSU detects the presence of vehicles that have previously subscribed to a brokerage service and hence are prepared to lease their resources for service provisioning. The RSU also stores and updates the database of the mobility patterns of these vehicles. Each vehicle has a probability of taking a certain trajectory based on its historic mobility pattern. Based on the preferred trajectories of the cluster, each vehicle node has a certain probability of following the cluster trajectory. A weighted graph is created where each link is weighted by the probability of two nodes staying together for a duration of time, called the CCP. Based

on this graph, the most well-connected cluster is selected, and the vehicular node with the highest connectivity is elected as the CN, based on a centrality measure.

Once the vehicle cluster and the CN are selected the process of service placement begins. The RSU sends the linear Type graph to the CN in the form of docker images. The CN also collects the updated resource information from the vehicular cluster. Initially, the minimum number of instances of each task is placed on the cluster, to process multiple data flows in parallel. Based on the number of video task instances (TIs) and the amount of processing required, the heuristic scales to more processing TIs by requesting TI images from the CN. If there are no more potential nodes left to place new TIs, the collected data is sent to the nearest RSU with the remaining TIs in the linear chain left to be placed for processing. The RSU can request a re-initiated cluster to place the remaining processing TIs on it.

The system model presented in this work augments the system model introduced in Chapter 3 by adding more detailed application types, instead of assuming black boxes of tasks with certain data processing requirements. In this section, we introduce the resource usage profiles of the two applications. We also present a more concise set of constraints that achieve an optimal placement without increasing the complexity of the ILP.

5.3 Model

5.3.1 Application Type

We consider a linear chain of video collection and processing tasks. We place multiple video collection TIs to increase the scale of video collection which results in better accuracy for object detection applications. To process multiple video streams, we scale all the processing tasks to multiple TIs, to utilize the limited processing capacity of vehicles. The multiple TIs also increases the resilience of the service in a dynamic vehicular network where mobility of vehicles increases node and link failures. We present two distributed services that follow our distributed service model:

Data Collection service : For this kind of service, an initiated cluster acts as "moving sensors" and only pre-processing and compression is carried out at the cluster. Most of the application-specific processing is carried out on more powerful edge computing nodes at RSUs. As an example, we take an application where Type 1 is a video capturing TI, Type 2 is a video pre-processing TI. Type 3 is a video compression TI.

To profile this application, at Type 1 TI, we first capture the video using OpenCV. At Type 2 TI, the video is pre-processed using Gaussian blur and simple thresholding. For

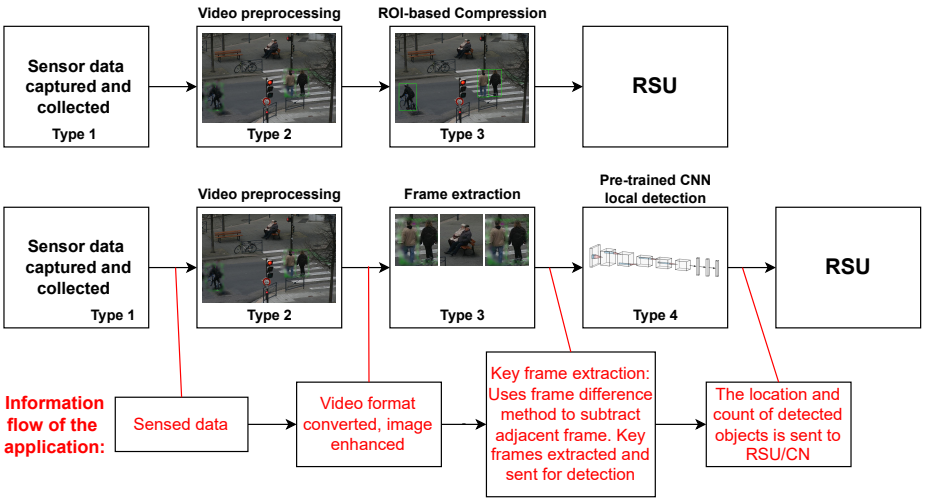


Fig. 5.3 Two types of applications of chain lengths 3 and 4.

Table 5.1 Task function, techniques and commonly-used algorithms for the data collection application's tasks.

Task type	*Task function*	*Task techniques*	*Algorithms*
Type 1	Real-time video capturing.		
Type 2	Video preprocessing.	Image enhancement, Noise reduction	Gaussian blur, Simple thresholding.
Type 3	Video compression.	Non RoI-based compression, RoI-based compression	Huffman encoding.

Type 3 TI, the images are compressed using RoI-based image compression. The task function, techniques and commonly-used algorithms are tabulated in Table 5.1 for this data collection application.

This collected data is sent to the mobile edge for an application that requires specific data over time like traffic monitoring and traffic management. Similarly, a large amount of data can also be collected for applications that require inference from more sophisticated Deep Neural Network (DNN) models that demand powerful cloud computing devices. For example, multiple 3D road maps can be generated from multiple sources of video data. This kind of application requires real-time and local information. Our service model sends pre-processed and compressed data, reducing the overhead of data transmission, for the data-intensive map generation task that can then be executed on mobile-edge computing devices.

Table 5.2 Task function, techniques and commonly-used algorithms for application the object detection application's tasks.

Task type	*Task function*	*Task techniques*	*Algorithms*
Type 1	Video collection		
Type 2	Video pre-processing	Image enhancement/ Noise reduction	PCA tranformation noise removal using Wiener filter.
Type 3	Feature Extraction	Regularization/ Dimensionality reduction.	Independent component analysis.
Type 4	Object detection	Local, pre-trained model for object detection.	Faster R-CNN, YOLO, tinyYOLO.

Object Detection application : The aim of building this distributed object detection application is to deliver a prompt and communication-efficient data processing service leveraging the resources available in moving vehicles. For this kind of application, all the processing of the collected data is executed on the vehicle cluster. Such applications have local context and scope, for example, using object-detection techniques to identify vulnerable pedestrians and alerting drivers in the vicinity. The Type 1 TIs for this application are of video collection type. The Type 2 TIs are the same pre-processing TI as Type 2 in application I. Type 3 TI is a frame extraction type that transforms video stream to images based on an extraction rate. For slow-moving pedestrians, the extraction rate is low which reduces the computation intensity of the task. For Type 4 TI, we use a pre-trained object detector called YOLO [Redmon and Farhadi, 2018] which determines if the object of interest is in the frame, from a detectable object pool. The task function, techniques and commonly-used algorithms are summarised in Table 5.2 for this application type.

If the Type 4 TI finds unknown objects, the images can be transmitted to back-end servers, which can thereby update the inference model of local detectors, but the global knowledge and cloud involvement are not in the scope of the applications we are defining for local detection.

We use the linear model described in [Zhu et al., 2019] to determine the memory usage for video streaming. For the case of video streaming the memory usage ranges between 110-220 MB. The data size for each frame is given in the range of 2.7-33.7 KB for five different video resolutions (1920 * 1080, 1280 * 960, 960 * 720, 640 * 480, 320 * 240). For the object detection application, we ran pre-trained YOLOv3 [Redmon and Farhadi, 2018] model on an Linux OS system with 8GB RAM and i7-6500U running at 2.50 GHz and got

a processing latency of 7.417 seconds. The detection has 16-18% of CPU usage. We also ran Tiny YOLOv3 and got a processing latency of 0.31277 seconds with lower confidence scores. The detection uses 4-5% of CPU and can be used on resource constrained on-board units for non-safety related applications that can afford lower accuracy.

5.3.2 Network Topology

The network topology consists of moving nodes that halt at an intersection and the roadside unit (RSU) that receives application requests from clients. The cluster is initiated by selecting nodes based on their mobility pattern and resource availability. This information of vehicles willing to lease their resources is stored and updated at the RSU. A vehicle cluster is initiated by detecting a density of $\mathbb{I}$ nodes subscribed to provide their onboard computing and camera resources. The RSU represented as $\mathbb{I}_{\text{RSU}}$ collects mobility and resource information from the $\mathbb{I}$ subscribed nodes. The mobility of each node is presented as the CCP, represented as p_I, which is the probability of the node staying at a particular road segment in a time interval from $[t_1, t_2]$. We also derive the communication link probability between two vehicles as the joint CCP for two vehicles to communicate over a period of time $[t_1, t_2]$.

The selected cluster of vehicles is represented as a directed graph $G(V, E)$. Each node of the graph $i \in V$ has K resource types. The available processing capacity, for each resource type k, is represented as $C_k(i)$. Each node i has a probability of staying on a road segment for a time period $[t_1, t_2]$, represented as $P_{(t_1,t_2)}$. The CCP of the RSU is equal to 1 as it is stationary and is always available from the viewpoint of mobility. The available link capacity between two nodes i_1 and i_2 is represented as $B(i_1, i_2)$ Kb/s. The joint probability between two nodes i_1 and i_2 depicts the probability of both nodes to stay together in a road segment for a period of time $[t_1, t_2]$ and is represented as $P_{t_1,t_2}(i_1, i_2)$. This is crucial for placing TIs that depend on other TIs for input data for task completion.

5.3.3 Distributed Service Model

The service model is composed of tasks, denoted as s_p, each with a different processing function or type, represented as p. Due to the limited resource capacity in each vehicle node, a service is composed in a distributed manner as a linear chain of tasks. Due to the dynamic nature of the vehicular network, each task can be scaled to multiple TIs, represented as s_{pj} where p represents the type of each TI and j represents the number of TIs. The number of TIs for each task s_p is N_{s_p} and the maximum number of allowed TIs for each type is given as $N_{s_p}^{\max}$. The objective of scaling tasks to instances is to increase the robustness in the service model, especially because of the link and node failure due to the wireless connectivity and

vehicle mobility. The resource requirement of type k for each task type p is represented as D_{kp}, where $k \in \{1,2,3,4\}$ for CPU, memory, GPU and video camera. The incoming flow from task types s_{p_1} to s_{p_2} is given as $F(s_{p_1}, s_{p_2})$.

The objective of this optimization is to find nodes that have a higher probability of staying together over a period of time. We then place two services of varying chain lengths (3 and 4), as described above. We model the mobile infrastructural resource constraints and the constraints required for placing a flexible and scalable service on the infrastructure. We then optimize resource usage in the service placement on the vehicular cluster through node and link cost, which is the sum of processing resources on vehicles and the communication cost for the data flow between the distributed tasks. The resource utilization is normalized to total available resources and weighted by the CCP to take into account the mobility of vehicle clusters.

5.4 Service Scaling and Placement Constraints

We define the service scaling and placement problem as a constrained optimization problem. We first define the constraints for the distributed service scaling, which are given as:

5.4.1 Flow capacity constraint

The processing requirement for a flow from TI $s_{p_1 j}$ to $s_{p_2 j}$ is given as $C(F(s_{p_1 j}, s_{p_2 j}))$. The constraint 5.1 ensures that each TI has enough processing capacity for the incoming flow. The constraint is given as:

$$\begin{aligned} &\forall i_2 \in \{1,\ldots,\mathbb{I}\}; \\ &\sum_{\forall p_1, j; p_2; p_1 \neq p_2} M(s_{p_2}, j, i_2) C(F(s_{p_1 j}, s_{p_2 j})) \leq C(s_{p_2 j}) \end{aligned} \tag{5.1}$$

where $C(s_{p_2 j})$ is the available processing capacity at TI $s_{p_2 j}$. Here, $M(s_{p_2}, j, i_2)$ is a binary mapping variable which is 1 when the TI $s_{p_2 j}$ is mapped to node i_2 and is 0 otherwise.

5.4.2 In-network processing constraint

Constraint 5.2 ensures that the flow is processed at each TI before being sent to the CN. To ensure that, we calculate the ratio of incoming to outgoing flow which should be equivalent to the data processing factor of each TI. The processing factor is given for each task type p

and is given as α_p. The constraint is presented as:

$$\sum_{\forall p,j;} F(s_{pj}, s_{(p+1)j})\alpha_p \leq F(s_{(p+1)j}, s_{(p+2)j}) \tag{5.2}$$

where $0 \leq \alpha_p \leq 1$. The data processing factor is 1 for forwarding nodes as it does not process the incoming data flow. The incoming flow from s_{pj} to $s_{(p+1)j}$ is given as $F(s_{pj}, s_{(p+1)j})$. The outgoing flow from $s_{(p+1)j}$ to $s_{(p+2)j}$ represents the flow that has to be processed at the TI $s_{(p+2)j}$.

5.4.3 Service Scaling constraint

The constraint 5.3 ensures that the TIs are scaled to the maximum number of TI specified for each task type p. This constraint also ensures that there is at least one TI for for each task type. This constraint is given as:

$$\forall p \;\; N_{sp_{min}} \leq N_{sp} \leq N_{sp_{max}} \tag{5.3}$$

where N_{sp} is the number of TI of task type p. The maximum allowed TIs for the task type p is given as $N_{sp_{max}}$ and the minimum number of TIs for task type p is given as $N_{sp_{min}}$ which is set to 1 for our model.

5.4.4 Infrastructure constraints

The infrastructure constraints ensure the the node and link placement meets the resource constraints for the service placement. The infrastructure constraints are given as:

5.4.4.1 Node Resource constraint

The resource requirement for a TI is represented as D_{pk} where p is the type of task and k is the resource type where $k = 1$ is CPU cycles requirement, $k = 2$ is memory capacity requirement, k = 3 is video camera resource requirement and k = 4 is the GPU availability. A decision variable $M(p, j, i)$ is used if TI s_{pj} is mapped to node i. The node resource capacity ensures that there is enough available capacity at a node to support a TI s_{pj}. The constraint is given as:

$$\forall i \in \{1, \ldots, \mathbb{I}\}, k \in \{1, \ldots, \mathbb{K}\}, \sum_{\forall p,j} M(p, j, i) D_{pj,k} \leq C_k(i) \tag{5.4}$$

where $C_k(i)$ is the available capacity at node i for resource k.

5.4.4.2 Bandwidth constraint

The bandwidth constraint ensures that the bandwidth requirement between two TIs, given as $F(s_{pj}, s_{(p+1)j})$, is less than the available bandwidth capacity over the entire path between two task instances s_{pj} and $s_{(p+1)j}$. The path between two nodes i_1 and i_n is a list of bandwidth of variable length. It stores available capacity over all forwarding nodes between i_1 and i_n, if there is no direct link between the two nodes and the available bandwidth link between the two nodes if they are directly connected. The bandwidth capacity of the path is represented as $path[B(i_1,i_2),\ldots,B(i_{n-1},i_n)]$. We enable this to support multihop clusters for cases where nodes might not be connected through a direct path but are connected over multiple hops. The constraint is given as:

$$\begin{aligned} &\forall i_1 \in \{1,\ldots,\mathbb{I}\}; i_2 \in \{1,\ldots,\mathbb{I}\}; i_1 \neq i_2 \\ \sum_{\forall p_1,j_1;p_2,j_2;p_1\neq p_2} &M(p_1,j_1,i_1)F(s_{pj},s_{(p+1)j})M(p+1,j_2,i_n) \\ &\leq \min(path[B(i_1,i_2),\ldots,B(i_{n-1},i_n)]) \end{aligned} \tag{5.5}$$

5.4.5 Mobility modeling

Each vehicle node has a certain probability of choosing a road segment based on its historic mobility pattern. The mobility history for each vehicle node is stored in the RSU along with the time stamp. We aim to select vehicles with the highest probability of staying at the selected road segment which is RS_j in our case, depicted in Fig. 5.4. A transition probability matrix stores the mobility probability for different road segments for all the vehicles registered to lease their resources and participate in the crowdsourcing service. New vehicles registering for the first time are also added to the table.

For discovering participating vehicles for the service deployment, the RSU broadcasts probe messages for participation requests. If already registered vehicles with known transition probability responds, they are given a priority over newer vehicles that want to participate. The newest participants are given the least confidence score. The confidence score is not issued according to the performance of the deployed task. It is a simple measure of updating confidence score if the vehicles follows its historic mobility trajectory. The confidence score is updated as the number of times the vehicle followed a preferred trajectory, over the total number of trips registered by the vehicle. Once the RSU updates its participants list, the RSU then runs the community detection-based node selection algorithm on a group of participants with a confidence score above a pre-decided threshold value.

We consider each road segment to be a Markov state. The vehicle transitions in the Markov process when moving from one road segment to the next. The vehicles follow a

Markov memory-less property, wherein the node transitions from state n to n+1 and are independent of state n-1. We record the transition of a vehicle from state RS_i to RS_j as the number of times a vehicle transitions to segment RS_j given the vehicle was at RS_i in its previous state. The probability is given as:

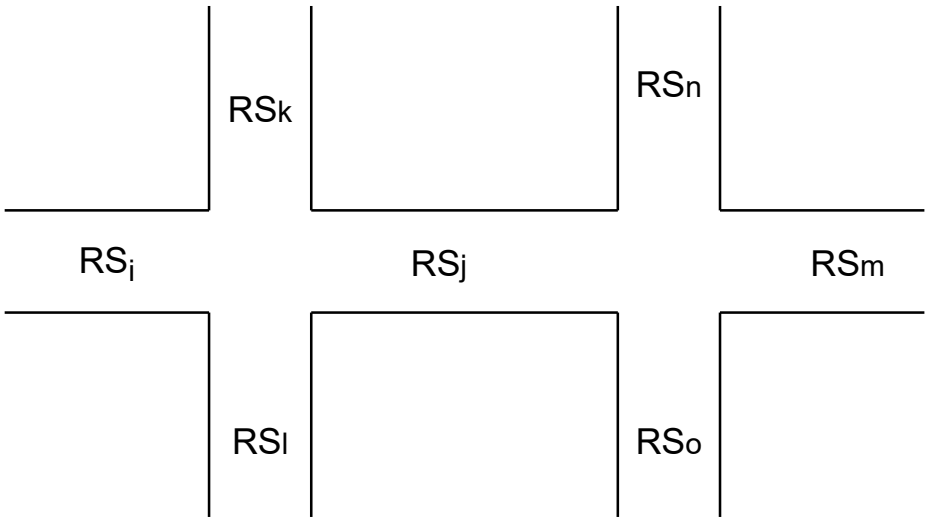


Fig. 5.4 Mobility modelling for selected road segments at intersections.

$$\begin{aligned} P_{t_1,t_2}(i)\{VS_{n+1} = RS_j | VS_n = RS_i\} \\ = \frac{\#(RS_i \; to \; RS_j)}{\#(RS_i)} \end{aligned} \tag{5.6}$$

where $[t_1, t_2]$ is the time interval selected based on the traffic state of the selected road segments. If the traffic is in a free-flow state, the time is chosen to be a 5-minute interval, and if there is queuing and the road is congested, the time interval is taken as 10 minutes. These values are inferred from our detailed experiments based on studying the predictability of vehicular flows at the Dublin intersection in our previous work [Sharma et al., 2021]. To make the microscopic model more realistic, we use the macroscopic data from Transport Infrastructure Ireland Traffic Data[1] to calibrate the microscopic probabilities for each vehicle at the selected Dublin intersections, using the SUMO simulator.

[1] https://trafficdata.tii.ie/publicmultinodemap.asp

5.4.6 Objective Function

We use node and link utilization cost as a measure to analyze the quality of placement of tasks on mobile vehicle nodes from the point of view of resource utilization. We aim to minimize the node and link utilization cost which is defined as:

5.4.6.1 Obj1: Node Utilization Cost

This is defined as the ratio of total used computational capacity to the total available computational capacity for the service placement. The ratio is weighted by the CCP of the node. This considers mobility of nodes rather than placing tasks on nodes with high processing capacity but with very low CCP to stay with the other vehicles in the cluster. The node utilization cost is given as:

$$\begin{aligned} \text{NodeCost}(i) = \sum_{\forall i,j,p;} \\ (1 - P_{(t_1,t_2)}(i)) \cdot (D_{pj,k}/C_k(i)) \cdot M(p,j,i) \end{aligned} \tag{5.7}$$

where $P_{(t_1,t_2)}(i)$ is the CCP of the node with the TI s_{pj} placed on it. As the CCP of a node increases, the cost of placing the TI on that node decreases.

5.4.6.2 Obj2: Link Utilization Cost

The link utilization cost is defined as the ratio of total used link capacity to the total available link capacity. This ratio is weighted by the CCP of the link. The CCP of the link defines the joint probability of two nodes to stay together during the time window $[t_1,t_2]$. The link utilization cost is given as:

$$\begin{aligned} \text{LinkCost}(i_1,i_2) = \sum_{\forall i_1,i_2;i_1 \neq i_2} (1 - P_{(t_1,t_2)}(i_1,i_2)) \cdot \\ (F(p_1,p_2)/\sum path[B(i_1,i_2)]) \Big(M(p,j_1,i_1) \cdot M(p,j_2,i_2) \Big) \end{aligned} \tag{5.8}$$

where $P_{(t_1,t_2)(i_1,i_2)}$ is the joint CCP of two nodes to stay together and $path[B(i_1,i_2)]$ is a list of available bandwidth on the path between two nodes i_1 and i_2 with placed TIs. The total bandwidth is summed over the path between nodes i_1 and i_2.

5.4.6.3 Obj3: Chain length/hop count

The number of hops for the distributed service can have a significant impact on both communication overhead and service reliability. To keep the hops between two placed TIs to the

minimum, we make sure to minimize the length of path between two nodes i_1 and i_2 with placed TIs. The network distance between two placed TIs is given as:

$$\begin{aligned} \forall i_1 \in \{1,\ldots,\mathbb{I}'_{(pj)((p+1)j)}\}; i_2 \in \{1,\ldots,\mathbb{I}'_{(pj)((p+1)j)}\}; i_1 \neq i_2 \\ H(i_1,i_2) = \sum_{\forall p_1,p_2} M(p,j,i_1), M((p+1),j,i_2) \\ len(path[B(i_1,i_2)]), \end{aligned} \tag{5.9}$$

where $H(i_1,i_2)$ is the hop count between two nodes i_1, i_2 with placed TIs.

5.4.6.4 Overall objective function

The multi-objective function aims to minimize Obj1, Obj2, and Obj3. Each objective is weighted by λ_1, λ_2 and λ_3 such that $\lambda_1 + \lambda_2 + \lambda_3$ totals to 1 and each objective is weighted equally (but this can be changed to reflect operational requirements). The objective function is given as:

$$\min \sum_{\forall i_1,i_2;i_1 \neq i_2} \lambda_1 H(i_1,i_2) + \lambda_2 \text{LinkCost}(i_1,i_2) + \lambda_3 \text{NodeCost}(i_1) \tag{5.10}$$

where $H(i_1,i_2)$ is the hop count between two nodes i_1, i_2 with placed TIs.

5.5 Heuristic-based Solution

Due to the nature of vehicular networking, it is required to scale services and find efficient service placement in a very short time, compared to the time required to solve the full integer-linear program (ILP). Consequently, we propose a node selection and service placement *heuristic* solution. We first leverage the historic mobility patterns of vehicles to select a group of vehicles that are more probable to stay together for a period of time using the principles of community detection.

The mobility of vehicle nodes is constrained by the underlying road topology. We model the available vehicle cluster as a graph with their joint CCP as the edge weight for each edge, depicting how probable are two nodes to stay together in the next time segment. The use of community detection using mobility behavior helps in identifying the most connected vehicular nodes that play a crucial role in the successful deployment of our service model with data-dependent TIs. Due to the mobility of nodes, the services can fail because of link and node failures due to vehicles leaving the cluster. The identification of communities helps in reducing service failures and subsequent service reconfiguration, by selecting a group of closely connected vehicles. Even if nodes or links fail within the selected community,

Algorithm 5.1 Service Placement

Input: LG Linear type graph, Vehicle cluster graph
Input: $(UL_{Type_{i+1}}, LL_{Type_{i+1}})$: Upper and lower limit for number of Type i+1 TI
Output: Successful/Unsuccessful service placement

```
1:  procedure SERVICEMAPPING(LG,VC)                                  ▷ The g.c.d. of a and b
2:    while Type_1 do                                                ▷ For all Type 1 instances
3:      for (Type_1,CN) ∈ TI_pairs do
4:        for i ∈ TI_pairs(Type_1,CN) do      ▷ Ensures placement of full chain for each Type_{i+1} instance
5:          TI_placement(i,VC,(UL_{Type_{i+1}},LL_{Type_{i+1}}))
6:          if placement is successful then
7:            Success
8:  procedure TI_PLACEMENT(i,VC,((UL_{Type_{i+1}},LL_{Type_{i+1}})))
9:    while (i) do                                                   ▷ for all available Type_1 TCI
10:     for (Type_i,Type_{i+1}) ∈ i do
11:       node_1 ← location of Type_i
12:       if Type_{i+1} instances exist on VC then                   ▷ To re-use instances
13:         node_2 ← List of location of Type_{i+1}
14:         for j in node_2_location_list do
15:           if resource at j ≥ resource required for Type_{i+1} then
16:             pathtonode_2 ← Get path from Type_i to Type_{i+1}
17:         sortedpath ← sorts path based on path length
18:         for k in sortedpath do
19:           if required_datarate ≤ min(k_path_datarate(i,j)) then
20:             place Type_{i+1} on node_2
21:             return TI Type_{i+1} placed
22:             break
23:       else
24:         if length(node_2_location_list) ≥ UL_{Type_{i+1}} then
25:           return Not enough resources on vehicle cluster
26:         else
27:           CN_location ← location of CN
28:           paths ← weighted path from Type_{i+1} to CN
29:           sorted_paths ← sort paths from highest to lowest path weight
30:           for i in sorted_paths do
31:             if required_datarate ≤ min(i_path_datarate(x,y)) then
32:               while nodes available in i do
33:                 v ← next node on i
34:                 if resource on v ≥ resource required by Type_{i+1} then
35:                   Place Type_{i+1} on v
36:                   break
37:               if no node available on path(Type_i,CN) then
38:                 return Unable to place TI on cluster
```

there will be alternate paths available within the community. The chances of a complete breakdown between any two nodes in the community are low.

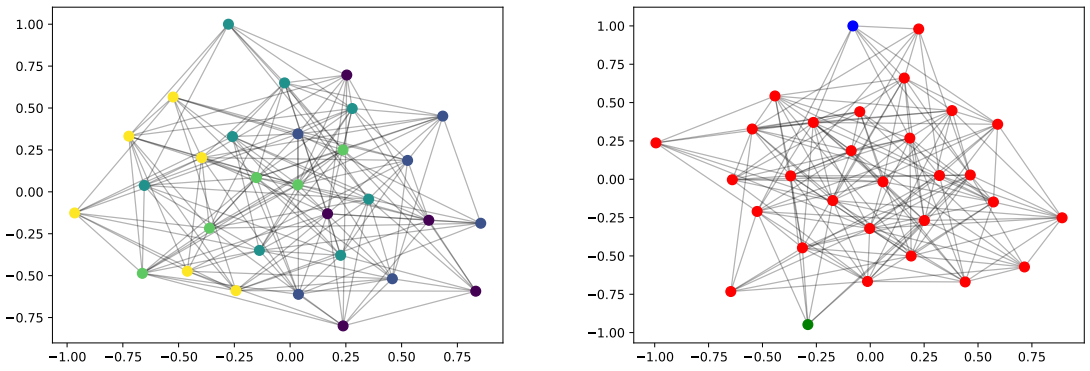
5.5.1 Vehicular Node Selection

The first step of the service placement problem requires selecting vehicular nodes that are more probable to stay together for a certain period of time. This is quantified using the CCP of the vehicles. At this stage, the scheme does not consider the available computation or communication capacity. We aim to find a sub-group of vehicles that have similar mobility patterns and follow a similar trajectory. We use community detection, which is the process of discovering cohesive groups or clusters in a network, to determine vehicular nodes that have better connectivity between them than the rest of the network. Using community detection algorithms, we partition the vehicular network graph into communities and, the biggest community is chosen for service placement. Due to the data-dependency between TIs in the service model, all TI's are promoted to be placed in the same community of nodes. Even if nodes leave or link fails within the selected community, there are alternate paths available in a well-connected community.

We analyze two community detection algorithms for selecting a vehicular cluster for the service placement i.e., Girvan and Newman algorithm and the Louvain algorithm. We use vehicular nodes and nodes interchangeably in the text. We aim to compare a modularity-score-based community detection algorithm with a method that focuses on calculating edge betweenness at each iteration of the algorithm. We first use the modularity score-based Louvain algorithm [Blondel et al., 2008] that initially starts with $|V|$ communities where each vehicular node is considered to be a community in the first iteration. Modularity is defined as the density of edges inside the community with respect to edges outside the community. In each iteration, every node is moved to its neighbouring community and the gain in modularity is calculated. If the gain is positive, the node does not return to its previous community. The iterations of the heuristic stop when the modularity gain between any two iterations, does not exceed a specified threshold value. The algorithm has the complexity of $O(VlogV)$ where V is the number of nodes in the graph. Thus, the algorithm does not depend on the number of edges in the graph. This is especially useful for the vehicular network where a node may be connected to multiple nodes in the cluster via wireless links. We simulate microscopic vehicular trajectories in a selected intersection using the SUMO simulator to compute the edge weights of the vehicular cluster, which is the CCP in our case. In our experiment, the modularity obtained in a cluster of 30 vehicles was 0.1428.

We also considered the hierarchical clustering-based Girvan and Newman algorithm [Newman, 2004] which derives a community tree or a dendrogram with a specified depth

[Lera et al., 2019]. The connectivity of a community increases as the depth of the derived dendrogram increases. The method first removes the edge with the highest edge betweenness centrality. The edge betweenness centrality is the sum of the fraction of the shortest paths that cross the edge. Each iteration splits every existing community into two new communities. The disconnected sub-graphs undergo the same procedure until the entire graph is split into isolated nodes. The complexity of the algorithm is $O(E^2V)$, where E is the edges of the graph and V represents the nodes. The modularity score for the same graph using this method is 0.00186. We, therefore, prefer the Louvain method as it results in a higher modularity score, which is more useful in this context, and Louvain's computational complexity is also lower and the complexity does not depend on the number of edges in the graph. Thus, the strongest selected community is the vehicle cluster that is used for the service placement problem.



(a) Louvain method

(b) Girvan and Newman method

Fig. 5.5 Using two different methods for community/cluster detection.

5.5.2 Service placement heuristic

After the vehicle cluster is selected based on the CCP of nodes using the community detection method, we then place the TIs by utilizing a graph-based heuristic. We first give as input, 1) the selected vehicular cluster which is the strongest detected community, 2) the linear type graph to be placed and 3) the upper $(\mathrm{UL}_{\mathrm{Type}_i})$ and lower limit $(\mathrm{LL}_{\mathrm{Type}_i})$ for the number of TIs of each type to be placed. The LL for all the tasks is 1 as we want to make sure at least one TI of each type is placed. The UL for each TI is equal to the number of video sources or Type 1 TIs, ensuring each stream gets one processing TI, in case the available processing capacity at individual nodes is very low.

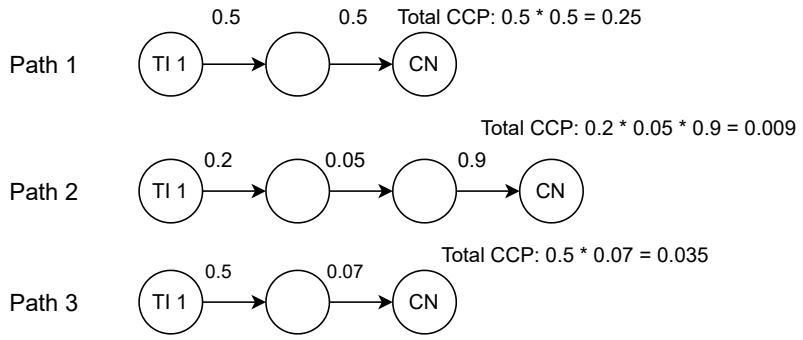


Fig. 5.6 Joint CCP-based path selection for service placement.

We modify a heuristic algorithm inspired by the work of [Dräxler and Karl, 2017], where VNFs are placed along the shortest path with the smallest bottleneck value. Instead of placing TIs on the shortest path, we consider the joint CCP of the path from a source TI (of Type 1) to the CN, as depicted in Fig.5.6. We then place TIs along the path with the highest joint CCP. As we intend to place a long chain of TIs along this path, choosing a longer chain increases the possibility of placing most TIs on the path to the CN. The heuristic may also randomly choose the shortest path, in terms of hop count, if the combined CCP of the path is the highest. From the three example paths shown in Fig.5.6, the heuristic will choose path 1 as it has the highest combined CCP, even though it is shorter than path 2. Choosing path 2 will result in placing TI on two nodes linked by very low CCP, 0.05 in this case. Path 3 is the same length as Path 1 but has lower joint CCP in comparison.

The heuristic is described in Algorithm 5.1. Similar to [Dräxler and Karl, 2017], we place TIs in a pairwise way, as the service model has dependent TIs. In our model, every TI of any type has a common endpoint as the CN. In line 3, we iterate over all TI pairs from Type 1 to the CN. In line 5, each TI pair is sent to the TI_PLACEMENT function along with the UL and LL for the TI. In line 11, the location of the Type 1 instance is detected. On line 12 it is checked if the next TI, of type $Type_{i+1}$, exists on the vehicle cluster. If it exists, say at node j, and the resource capacity at node j meets the capacity constraint for TI $Type_{i+1}$, all the paths from $Type_i$ to $Type_{i+1}$ are stored in the list $sortednode_2$. In line 18, all the available paths are iterated over and the bottleneck edge capacity for each path is compared to the required available capacity between the two TIs. If the constraint is met, $Type_{i+1}$ TI is reused for the flow. If $Type_{i+1}$ does not exist on the cluster, it is checked if the upper limit for the TI type is met (on line 24).

If the upper limit is not reached, all the paths are explored from Type_i to CN of the cluster. All the paths are sorted based on the path weight, which in our case is the total CCP of the path. In line 31, all the paths are iterated over, and the bandwidth capacity requirement

Table 5.3 Optimality gap percentage for the link utilisation cost for different number of Type 1 TIs.

Number of Type 1 instances	*Optimality gap (%)*
1	0
2	0
3	11.03
4	13.05
5	10.997
6	17.83

is checked for the path. If the bandwidth requirement is met and the resource capacity requirement for the node is met, then Type_{i+1} is placed on the node v. If there are no more available nodes on the path to the CN, a failed placement is registered. Thus, this approach aims to send the collected data to the CN and tries to place processing TIs in-network when possible.

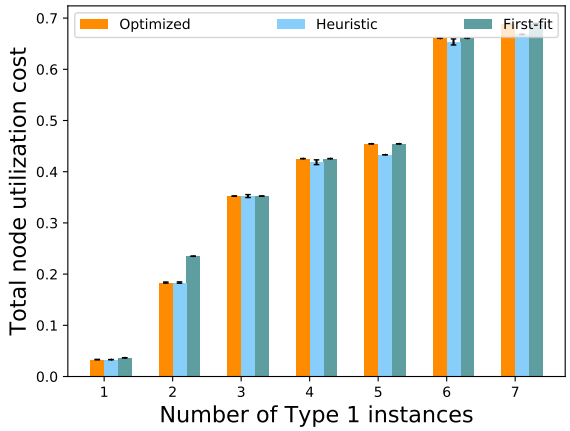
5.6 Evaluation

In this section, we evaluate the performance of the Mixed Integer-linear Program (MIP), the proposed heuristic, and the first-fit approach through resource utilization metrics for bandwidth (link) and processing (node). We also compare the MIP, the first-fit, and the heuristic for the total optimal value. We then evaluate the average chain length and the total instances that the heuristic scales to analyze the performance of the heuristic in terms of not overprovisioning resources. We then use total response time as a Quality-of-Service measure and compare our placement approach to a static mobile-edge computing approach, which does not use task replication in the form of multiple instances of the same task. We also evaluate the performance of the selected vehicular cluster using centrality measures. We use vehicular mobility on a Fog-computing-based simulator, to study the evolution of the selected cluster through its lifetime.

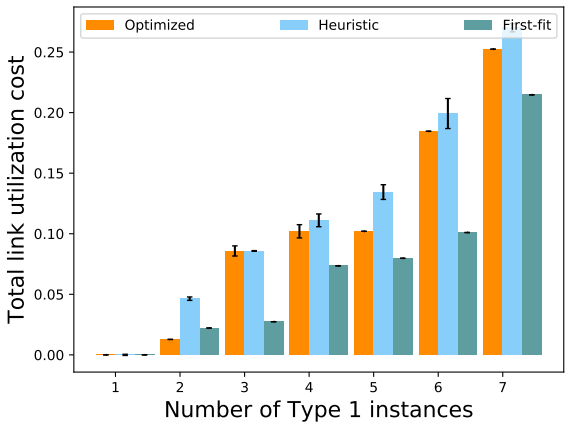
We first use Gurobi, a standard MIP solver to solve the multiple objectives, constrained optimisation problem. We find an optimal solution for placing both service types of different chain lengths on a selected vehicle cluster. We evaluate the solution for the node utilisation cost, link utilisation cost, and the total objective value for placing multiple applications of chain lengths 3 and 4. We vary the number of video instances from 1 to 7 to evaluate the scalability of the experiment. The applications are defined in Section 5.2, where the ‘data collection application’ type is rich in data flow and has low processing requirements, whereas

the ‘object detection’ application type is more compute-intensive and has less bandwidth requirement. The applications are of variable chain length. We use the first fit approach as the baseline approach. It sorts all the available paths from the data collecting TI to the CN and sorts them based on the highest available resource capacity. It then places TIs on all the available vehicle nodes on that path.

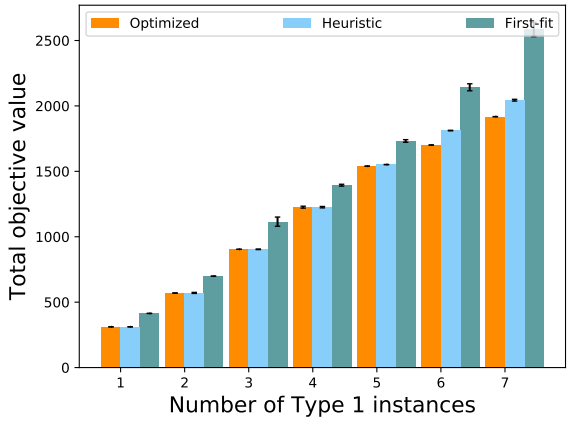
We first place two applications of the first type on the selected vehicular cluster. For the case of the 3 task chain, our heuristic gives better node utilization cost as compared to both optimal and first-fit solutions, as shown in Fig.5.7a. Our heuristic gives comparable link utilization cost in comparison to the optimal solution for 1-3 Type 1 TIs, but it becomes less efficient for a higher number of Type 1 TIs, as shown in Fig.5.7b. This is due to prioritizing paths with higher CCP which may result in selecting longer routes between dependable TIs.



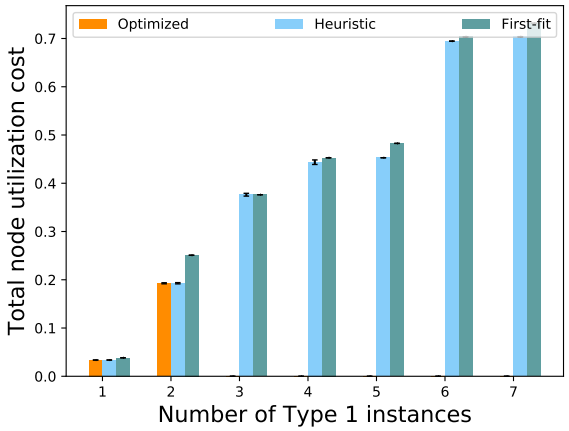
(a) **3 task chain: Total node utilization cost.**



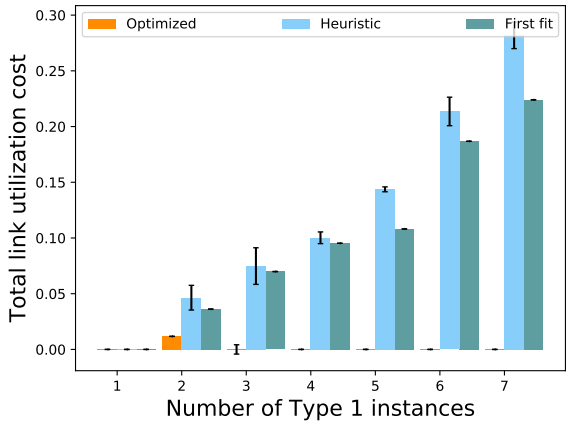
(b) **3 task chain: Total link utilization cost.**



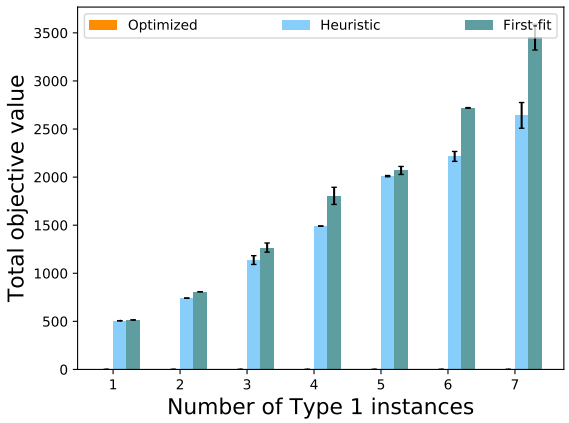
(c) **3 task chain: Comparison of the total objective value.**



(d) **4 task chain: Total node utilization cost.**



(e) **4 task chain: Total link utilization cost.**



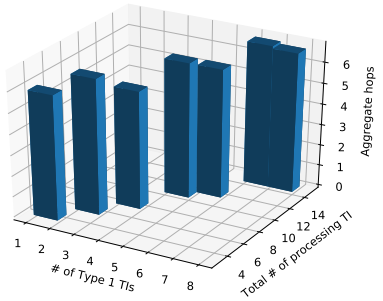
(f) **4 task chain: Comparison of the total objective value.**

Fig. 5.7 Total node utilisation cost, total link utilisation cost and total objective value for task chain of lengths 3 and 4.

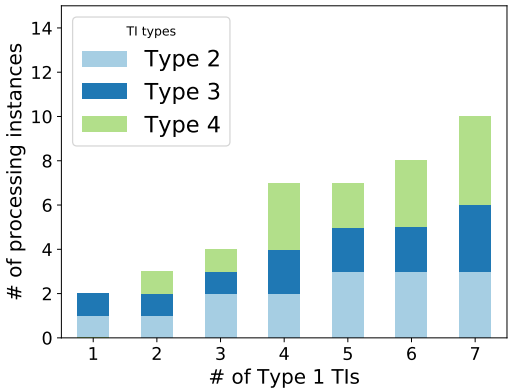
For the second case, we place both applications of chain length 3 and 4 TIs for both services. The ILP solver fails to give a solution in a reasonable time for applications with a longer chain length. We get a solution from the solver in seconds for 1-2 Type 1 TIs. But as the number of Type 1 TIs increases, the solver does not converge to a solution even after hours. We, therefore, compare our heuristic solution to the baseline approach. Our heuristic performs better than the baseline approach for any number of Type 1 TIs (from 1 to 7), as shown in Fig.5.7d. The baseline approach outperforms the heuristics solution for the link utilization cost for the second case, as shown in Fig.5.7e. This is due to choosing paths that have higher joint CCP, to increase the robustness of the service placement. This results in more bandwidth utilization as a tradeoff to selecting more robust paths. Our approach outperforms in minimizing the total objective value as compared to the first-fit approach, as depicted in Fig.5.7f. The baseline approach fails in minimizing the total objective for the higher number of Type 1 TIs.

As our heuristic solution does not select the shortest path but the most reliable path, it might select very long paths with multiple hops between the Type 1 and the CN TIs. We present both the aggregate hop count and the total number of scaled processing TIs (of Type 2, 3 and 4) corresponding to the number of Type 1 TI, presented in Fig.5.8a. For any number of TIs, the hop count of all the paths between Type 1 TI and CN ranges between 5 and 6. The total number of processing TIs for each placement is also plotted in the same figure and it ranges from 4 to 15.

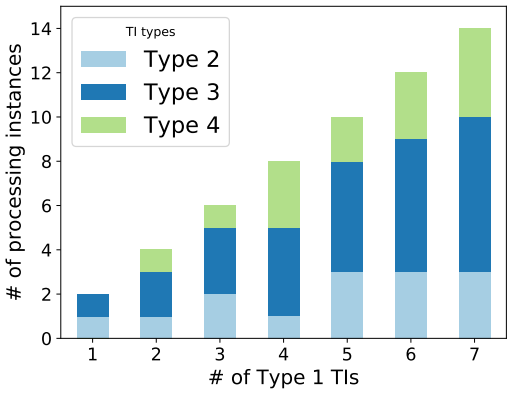
To compare the number of scaled processing TI's, we run the optimization problem with the objective of minimizing the number of processing TIs, to analyze the least number of processing TIs required for meeting the application demands. We calculate the minimum number of Type 2, 3, and 4 TIs required for a successful service placement without any TI being rejected a placement on the vehicular cluster in Fig.5.8b. We compare this to the number of TIs scaled by our heuristic, plotted in Fig.5.8c. As shown, our heuristic scales are close to the minimum number of required TIs. It places 2-4 more TIs in comparison to the least number of required TIs. But the heuristic chooses more reliably connected nodes to place the TIs.



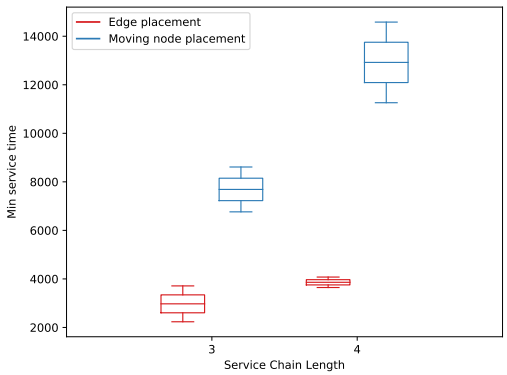
(a) **Comparison of total number of scaled, processing TIs and aggregate hops for the 3 TI chain placement corresponding to the number of Type 1 TIs.**



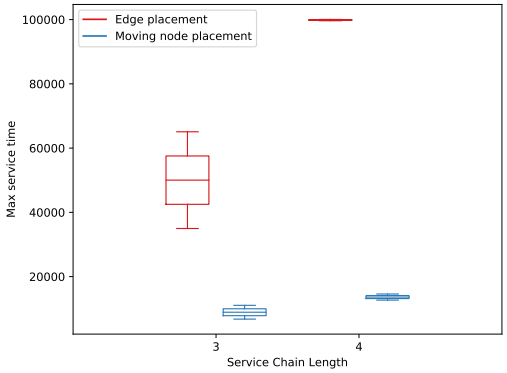
(b) **Minimum number of required processing TIs to meet the service demand corresponding to the number of Type 1 TIs.**



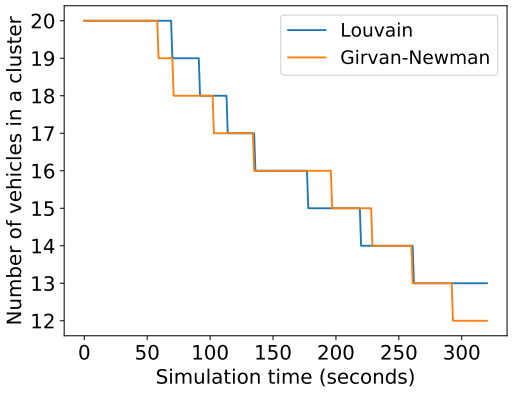
(c) **Total number of processing TIs scaled by our heuristic corresponding to the number of Type 1 TIs.**



(d) **Minimum service time for different chain length.**



(e) **Maximum service time for different chain lengths.**



(f) **Cluster state through the simulation.**

Fig. 5.8 Different performance metrics to evaluate the performance of the proposed heuristics, service time comparison for the proposed approach with edge placement and evaluation of the cluster state throughout the states of the simulation.

5.6.1 Comparison of placement techniques in terms of service time

We have evaluated our placement approach from a resource utilization point of view. We now look at a QoS-based metric to compare our approach to a mobile-edge computing-based solution. The service demands are still generated by moving vehicles, but the mobile-edge computing approach places all the TIs on static edge servers. For the real-time performance of the vehicle cluster, we use a fog computing simulator called Yet Another Fog Simulator (YAFS) [Lera et al., 2019] for modeling the mobility and estimating the real-time performance of the selected cluster. YAFS is a python-based discrete event simulator that supports resource allocation policies in fog, edge, and cloud computing. The simulator has a distributed data flow application model that could be easily adapted to our use case. The simulation provides dynamic service selection, placement, and replacement of services that we have customized for our requirements. The support of mobility of nodes, which can also be treated as processing nodes makes the simulator a good fit for our case.

We consider the service time as the total time it takes for a service to execute, including both processing and link latency. We observe the minimum and maximum service time for the two services of different chain lengths for the mobile-edge placement versus our approach of placing multiple TIs on a moving vehicle cluster. We observe that the minimum service time is significantly less for the cloud placement, in comparison to our approach, in Fig.5.8d. Our approach places multiple TIs on different vehicles, thus the delay in the execution of one TI can result in a significant delay in service execution time. For the case of maximum service time, as can be seen in Fig.5.8e, the mobile-edge placement is significantly high. This is because of the delay in sending all the collected data from moving vehicles to the edge server or RSU. The service time is also higher for the chain length of 4 for the mobile edge placement approach. In comparison, our approach approximately takes the same time for a chain length of 3 or 4 in the worst-case scenario as the minimum service time in Fig.5.8d. Thus, even if an optimal placement is not achieved, on average our approach performs better in terms of service time, in comparison to the mobile-edge placement approach.

5.6.2 Evaluation of the selected cluster over time

We also analyse the performance of the selected cluster by evaluating the number of nodes in the cluster that stay together over a period of time. For the two community, detection-based node selection approaches, out of the 20 selected nodes, 12 to 14 nodes make it till the end of the simulation, as depicted in Fig.5.8f.

We then evaluate the quality of the selected cluster in terms of the nodes that stay till the end of the simulation by using a resilience score. We use the betweenness centrality as

a measure to check the importance of a node, in terms of connectivity in the graph. The *betweenness centrality* calculates the shortest weighted path between every pair of nodes in a graph. Each node gets a betweenness centrality score (BCS) based on the number of shortest paths that pass through the node. The resilience score is calculated as the total BCS for all the nodes that made it till the end of the service time upon the total BCS of all the nodes in the selected cluster. The higher resilience score shows that nodes with higher BCS stayed with the cluster, thereby reducing the need for rerouting flows or re-configuring service chains due to the absence of a forwarding node or a path between two data-dependent TIs. We evaluate the communities detected for two community sizes, of 15 and 30, using the Girvan-Newman and the Louvain approaches. We observe a higher resilience score for the Louvain approach for communities of either size, as depicted in Fig.5.9.

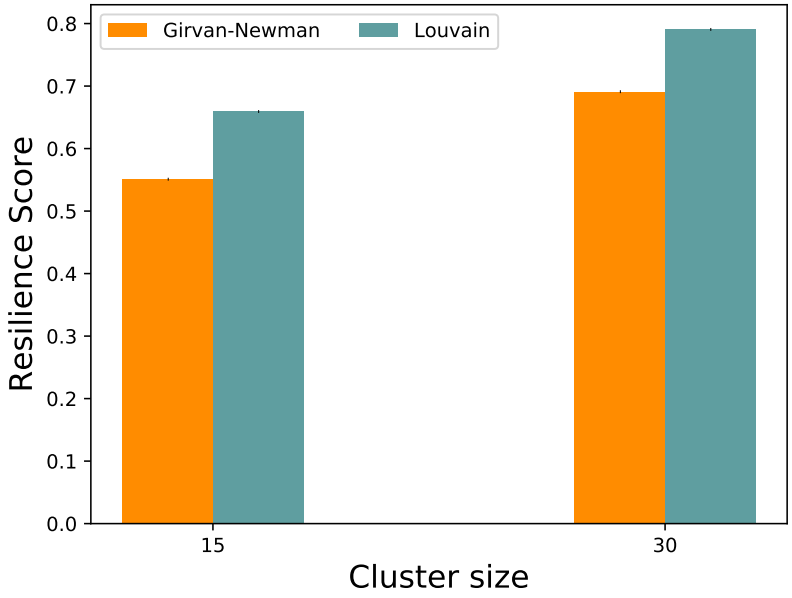


Fig. 5.9 Calculated betweenness centrality score for the two community-detection approaches.

We observe a higher resilience score for the bigger cluster, of 30 nodes, as we observe more number of nodes with higher BCS stay with the cluster till the end of the simulation time. However, this might not always be the case. A bigger cluster may not always end up being more resilient than a smaller cluster. The resilience score depends on the BCS, which is based on the importance or the role of a node in the graph in terms of the flow of communication. For example, the BCS of any vertex in a complete graph is zero since no vertex lies on the shortest path, as every node is connected to the other by a unique edge.

5.7 Conclusion

The chapter aims to solve the problem of placing video collection and object-detection applications on moving vehicle clusters. The first application executes pre-processing tasks on the vehicle cluster whereas the second application executes a pre-trained local object-detection service, which is computation-intensive. We then model the problem as a multi-objective, constrained optimization problem. We introduce a vehicular node selection and service placement problem with the novelty of placing scalable and distributed services on mobile infrastructure.

We also evaluate the performance of the service placement heuristic using other resource utilization measures like the number of scaled TIs and the average hop-count for placing the distributed services. We then consider a QoS-level parameter called service time to analyze how our approach performs compare to a mobile-edge placement approach. We also emulate the service placement using a Fog simulator. We analyze how the node selection approach performs in terms of the life of the selected cluster. We introduce a betweenness centrality-based resilience score to evaluate the performance of the chosen cluster, in terms of the quality of nodes that make it to the end of the execution time.

Our approach outperforms the integer linear program because it generates placement plans more quickly but with similar resource costs, and outperforms the baseline first-fit solution, because the mobility-aware strategy ensures that the cluster cohesion is higher, increasing the resilience of the system. We also compared our vehicular fog computing approach to edge computing, where the vehicles are not used for processing and just forward the data to the RSU and cloud for processing. Our placement technique results in better worst-case performance, with a much lower maximum service time which is a measure of the time taken in service execution, including both processing and link latency.

Chapter 6

Video Data from the MLK Corridor

This chapter aims to use data from an intersection-based testbed to implement a realistic application in the template introduced in this work. So far, the thesis focuses on the efficient placement of services that are spread out to TIs to increase resilience in the service. At the same time, the number of TIs is bounded to an upper limit, which ensures that the computational and communication resources available on vehicles are not over-provisioned thereby not utilising the resources efficiently. Thus, the introduced optimisation tries to strike a balance between a widely spread versus an effectively narrow service placement. This chapter focuses on the practical implementation of the distributed data collection and processing application. We specifically deploy an object detection application that utilises video data collected from multiple sources at a road intersection.

To make a realistic estimate of the resource usage and performance of the video collection and processing application, we use video data from an intersection-based testbed in the Centre for Urban Informatics and Progress (CUIP), Chattanooga, Tennessee. The testbed, presented in Fig. 6.1 is a 1.2-mile, real-world sandbox that comprises 11 poles fitted at intersections. The poles are equipped with air quality sensors, cameras, LIDAR, RADAR, audio recording and networking capabilities. To use the concept of collecting video data from multiple video data sources, we consider video from three video cameras projected on poles at an intersection. The use of video cameras provides more information about the environment with high resolution and textual information in comparison to other sensors such as LIDARs, radars and ultrasonic devices etc. The data collected from the video cameras can not only help in assessing the usage requirements of a video processing application, it can also help in the realistic implementation of the applications suggested in this work.

We specifically focus on an object detection application used to first classify cyclists, pedestrians and vehicles. The application is further used to detect vulnerable pedestrians and cyclists that are jaywalking or are in a vulnerable position. This kind of application is

challenging to deploy as video data collected from moving cameras results in reidentification of objects. We also aim to collect video data from multiple cameras, which will result in more pre-processing of data to avoid redundancy in frames. Our service model also has the advantage of using vehicles as ad-hoc sensors based on the requirement of the application. The collection of video data from multiple sources can be extended to collect multi-modal data for more accurate scene understanding application. Our model can also be used to study the traffic flows at an intersection more thoroughly. This can help in detecting usage patterns and especially the traffic conditions at different times of the day to improve traffic efficiency. Different from traffic-related applications, an estimate of vehicular traffic can help in initiating data collection for applications related to surveying, recording accidents or crimes in a neighbourhood, and studying the usage pattern of roadside gas stations, libraries and cafes.

This chapter first highlights the object detection techniques that we have used on our custom dataset in §6.0.1. The section describes two state-of-the art techniques used commonly for object detection including YOLOv2 and Faster R-CNN. We then introduce a Federated Learning [Li et al., 2020b] based object detection scheme that can be used to collect and process data at the edge of the network in §6.1.1. This section introduced both centralised and decentralised federated learning model [Saha et al., 2021] that can be used where the aggregator is either centralised or deployed as many different local aggregators. §6.1.2 presents the system architecture of the standard FL framework and the proposed distributed FL framework. We then present the preliminary results and the future direction of the research in §6.2. The conclusion is then presented in § 6.3.

6.0.1 Object detection techniques

We first test the standard object detection algorithms like YOLO and Faster R-CNN Resnet-50 FPN. We then introduce a distributed federated learning scheme that has an objective of reducing the number of communication rounds in the model, making it more resource efficient.

1. YOLO: You Only Look Once is a state-of-the-art, real-time, deep learning-based object detection system consisting of a Convolutional Neural Network (CNN) and an algorithm for post processing outputs from neural network. A CNN is a deep learning algorithm which can take in an input image and assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. Instead of training data to learn the filters and the characteristics of the input data, CNNs have the ability to learn these filters. Unlike other region

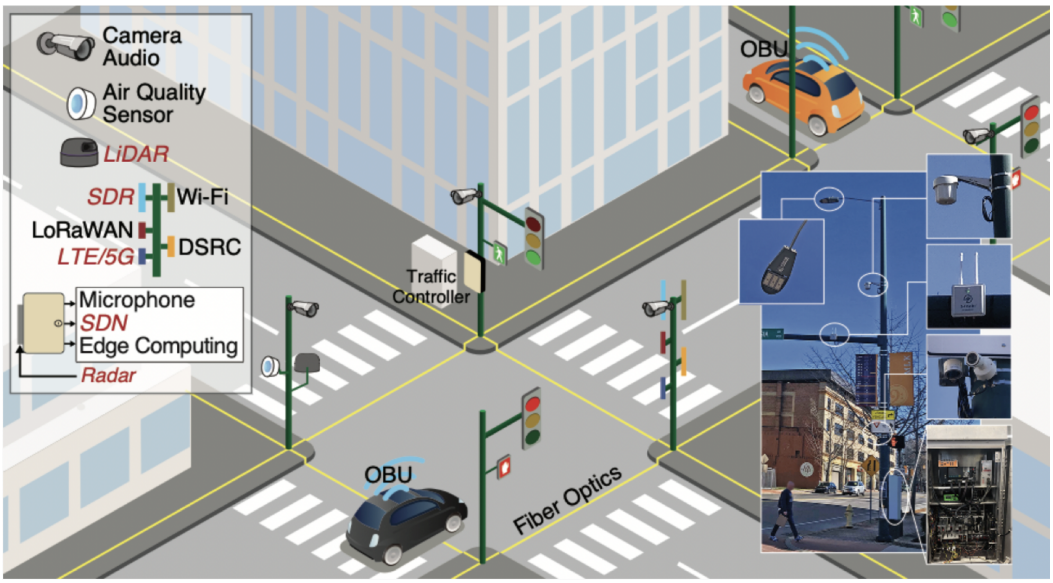


Fig. 6.1 The CUIP testbed corridor.

proposal classification networks, like Fast RCNN, that perform detection on various regions in an image, YOLO architecture passes the image once through the CNN and outputs the prediction. In the YOLO architecture, a single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. This results in YOLO being a faster system compared to competing system that implement region-based CNN.

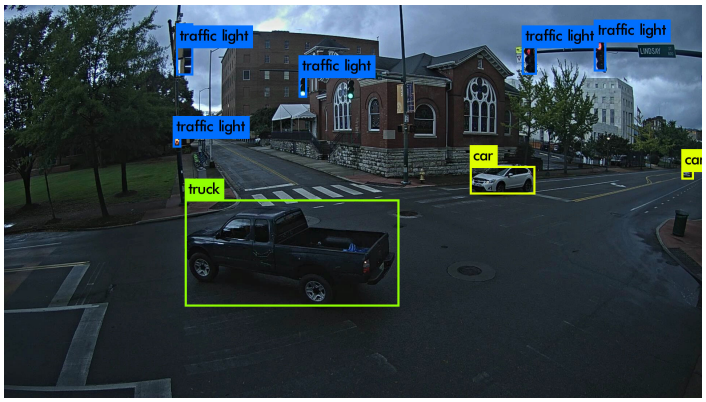


Fig. 6.2 Detected objects using YOLO.

YOLO is an ideal candidate for distributed object detection used in the case of Fog computing. In our case, the edge or fog node is used to both collect and resize or pre-process the original images. This results in effectively reducing the amount of data

that is transferred to the cloud. The distributed application can also result in reduced communication resource usage and latencies by using pre-trained models received from the cloud at the edge node. We use YOLO algorithm on our dataset to detect classes from three videos captured in overcast condition at the same intersection from the testbed. The prediction on one of the frame is depicted in Fig. 6.2. The confidence scores for the image are: traffic light: 98%, traffic light: 97%, traffic light: 91%, traffic light: 65%, traffic light: 55%, car: 97%, truck: 99%, car: 99%.

2. Faster R-CNN Resnet: is an object detetion model that improves the Fast R-CNN model by using the region proposal network (RPN) along with the CNN model. The Faster R-CNN model introduces a RPN that shares full-image convolutional features with the detection network, thus providing nearly cost-free region proposals [Ren et al., 2015]. The RPN and Fast R-CNN are merged into a single network by sharing their convolutional features: the RPN component tells the unified network where to look. As a whole, Faster R-CNN consists of two modules. The first module is a deep fully convolutional network that proposes regions, and the second module is the Fast R-CNN detector that uses the proposed regions. Fig. 6.3 represents the prediction in three different frames using the Faster R-CNN Resnet model. Fig. 6.4 represents the training and validation loss for the model.

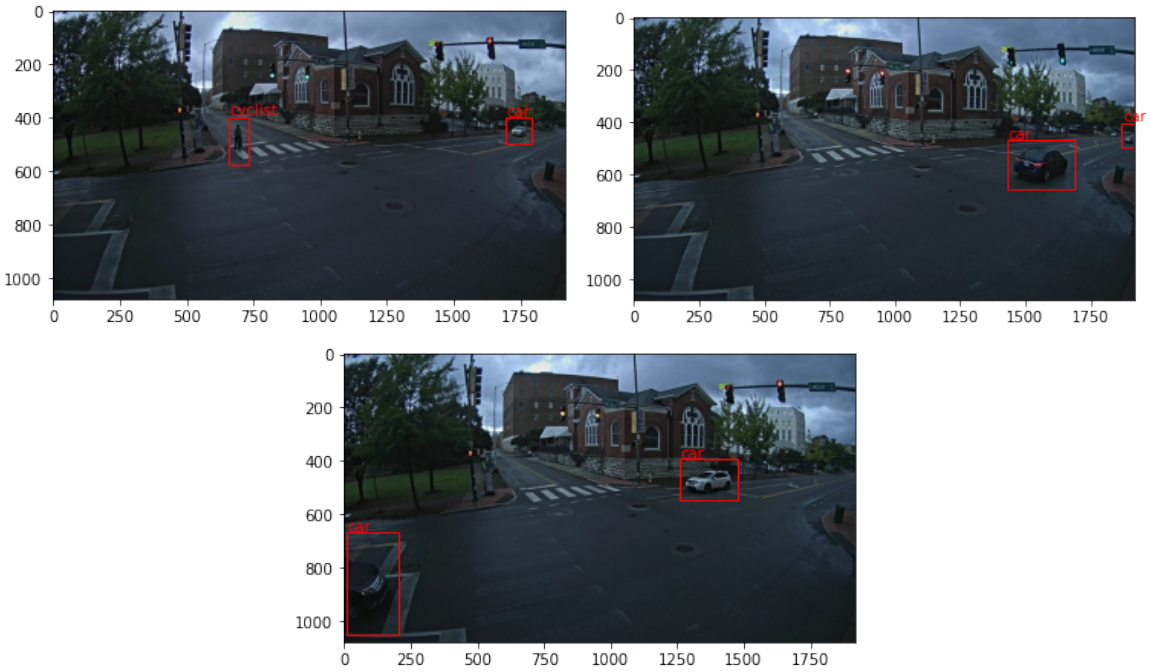


Fig. 6.3 Detected objects using Faster R-CNN Resnet.

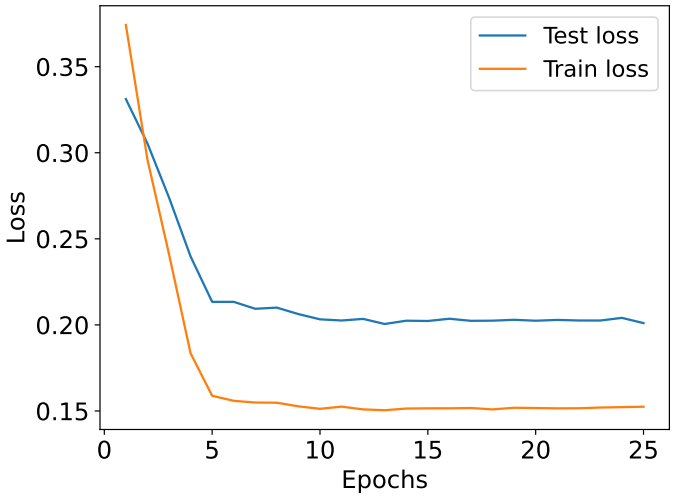


Fig. 6.4 Training and validation loss for Faster R-CNN Resnet.

6.1 Federated Learning based Object Detection

We devise a method to use the massive amount of informative training data collected at the network's edge. Sending this data to the cloud can be very expensive and slow. Most of this data is location and context-specific, with very low latency requirements, specifically for safety-related, vehicular applications. We use Federated Learning, where each client performs a few local Stochastic Gradient Descent (SGD) updates using its own data, and the resulting models are aggregated at the central server. One of the critical features of FL is that the data collected by client devices are heterogeneous in size and can also be collected asynchronously. The computational capacity of the clients is also heterogeneous, which varies the computational speed of training data.

6.1.1 Federated Learning

Federated Learning is a form of distributed machine learning, where the data is trained on different devices or clients. After training different data on different devices with the same model their weights are sent to the central server, where weights are aggregated using different techniques. After aggregation, the global weights are again sent to the clients and the training is carried out on the client devices. This approach is beneficial from the point of view of data privacy as well as removes the need to send collected data to a central server. The approach also helps in identifying client-specific features and results in improving the global model weights. The research in FL has led to different architectures based on the requirement

of the use case. For resource-constrained IoT devices, the focus of the architecture is on communication efficiency. Similarly, most of the client devices are not available for training at a given time, which results in asynchronous training of the FL model. We first list the different ways in which FL architectures are classified.

- Centralized federated learning: The process of federated learning is coordinated by a global aggregator. The global aggregator selects, coordinates, and configures all the participating nodes. This can become a single point of failure or a bottleneck that can affect the quality of the global model.
- Decentralized federated learning [Saha et al., 2021]: In distributed federated learning, many local aggregators participate in the learning process. The distributed federated learning approach can leverage from the Fog computing paradigm where different nodes with different computation capacities are assigned the role of the client with local training of the model, and for the role of local and global aggregators. The global model is only updated in this model after several local aggregations and communication rounds, which reduces the communication cost in the proposed framework.

6.1.2 System architecture

We consider I video cameras, generating video streams that will be used for object detection for pedestrian and cyclist safety. We consider a cluster of n nodes that are willing to participate as local processors and aggregators. The RSU acts as an edge device for global aggregation. The clients locally update the model parameter x. Each training dataset is split into the input and desired output, represented as (X_k, Y_k). The loss function for each data sample is given as $l(X_k, Y_k, W_k)$. We assume that each participating node has D_1 to D_N local datasets. The local loss function at the i-th client is defined as:

$$F_i(x) = \frac{1}{|D_i|} \sum_{\forall \eta \in D_i} f(x; \eta) \tag{6.1}$$

where $|D_i|$ is the local dataset for the i-th client. The local aggregation parameter x_l after t epochs is given as:

$$x_l = \frac{\sum_i^{n_s} n_i x_i(t)}{\sum_i^{n_s} |D_i|} \tag{6.2}$$

where n_i is the number of local clients under the local aggregator and n_s is the total number of local aggregators. Typically the global objective is the weighted average of local objective in proportion to the size of the data at the i-th client and is given as:

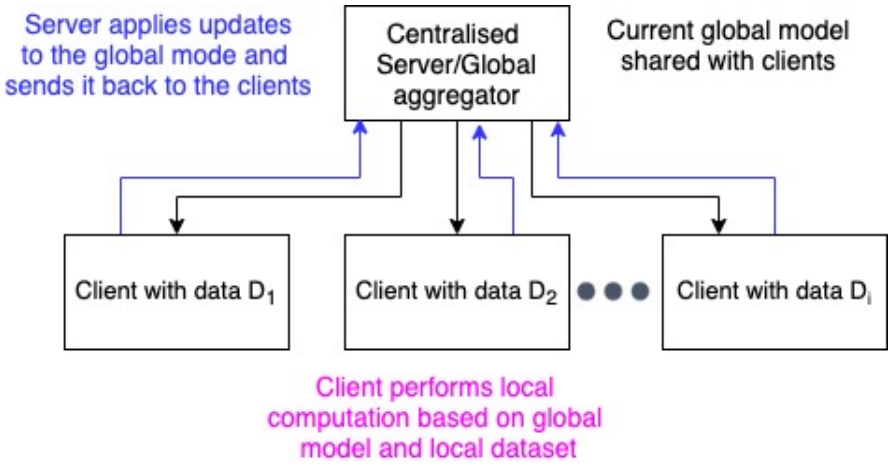


Fig. 6.5 Standard FL framework.

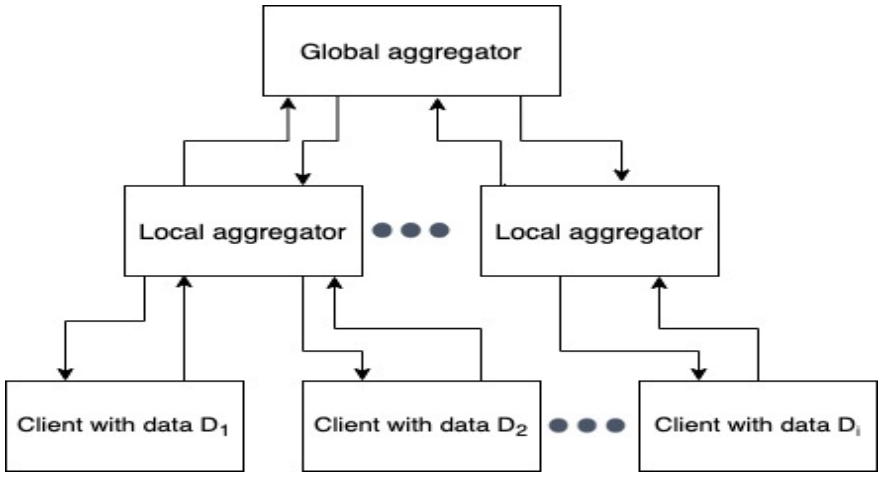


Fig. 6.6 Proposed, distributed FL framework.

$$F(x) = \sum_{i=1}^{m} P_i F_i(x) \text{ where } P_i = \frac{|D_i|}{\sum_{i=1}^{m} |D_i|} \tag{6.3}$$

The goal of the model is to find x such that F(x)is minimized. The local loss parameter for each client i is represented as x_i and is calculated as:

$$x_i(t) = x_i(t) - \eta \nabla F(x_i(t, B_i)) \tag{6.4}$$

where $\eta > 0$ is the learning rate, where $t = 1...N$ and represents the number of iterations and B_i represents the mini-batch size. Thus, each client locally takes one step of gradient descent on the current model using the local data whereas the global model takes a weighted average of the client models. The number of local updates is given as $u_i = E_i \frac{n_i}{B}$, where E_i is the number of training passes the i-th client makes on its local dataset on each round and B is the local mini-batch used for client updates, and $n_i = |D_i|$.

6.2 Preliminary Results and Future Direction

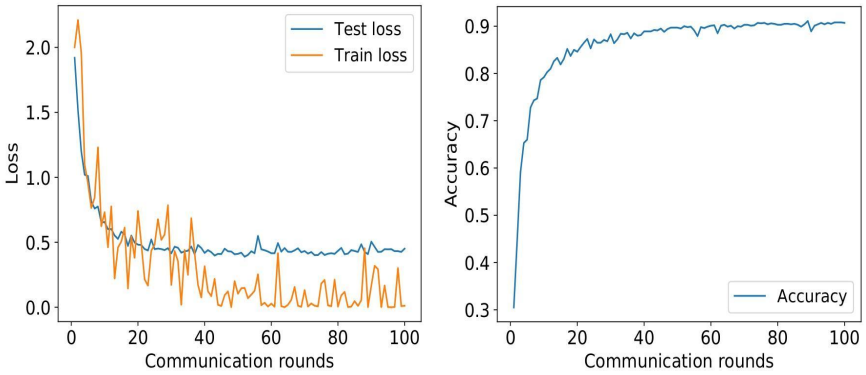


Fig. 6.7 Training and test loss and accuracy of the centralised FL model.

We first used a centralised FL framework trained on our custom dataset to study the model's performance. In Fig. 6.7, the training and test loss of the model is presented against the number of communication rounds. The classification output's loss is considered on the test and train datasets. The figure also represents the test accuracy of the model, which reaches 0.903 by the end of 100 communication rounds in the model. The accuracy is calculated as the ratio of correct predictions to the total number of predictions for all classes. We also used the distributed FL framework on our custom dataset to study the model's

performance. We observed lower global test accuracy in the model of 0.614. We aim to now make the distributed model more accurate by optimising the number of local aggregators used in the distributed model. We also aim to study the total network usage in deploying the distributed model compared to the centralised FL model. We can estimate the trade-off between a less accurate model that is more resource efficient compared to the centralised FL model.

6.3 Conclusion

This chapter aims to address RQ4. introduced in § 1.1. The question focused on developing a FL-based scheme to deploy a distributed object detection application using video data from a testbed. The aim of this chapter is to introduce a realistic application as an extension of our proposed service model, network model and the service placement problem introduced in this dissertation. The FL-scheme can be deployed in the service template introduced in this work. FL is a popular distributed learning approach that focuses on enhancing data privacy while collaboratively training the model, learning a shared prediction model. The centralised FL model suffers from communication overhead and high computational requirement. To make the model adaptable for the resource-constrained Fog devices, we prosed a decentralised FL model with local aggregators.

We were able to gather video data from a 1.2 mile long, road-intersection testbed in Chattanooga, Tennessee. Using the data we explored the performance of different standard object detection techniques introduced in the literature. We used YOLO and R-CNN Resnet on our custom data to study the accuracy of the predictions. We then introduced both centralised and decentralised federated learning architectures. We then introduce the system architecture for the FL architecture and present the preliminary results for both centralised and decentralised models on our custom dataset. We also highlight the future directions that we will take to improve the accuracy of the proposed decentralised FL model.

Chapter 7

Conclusions and Future Work

The aim of the dissertation is to introduce methods to utilise the sensing, processing and communication resources in moving networks. The utilisation of moving networks requires novel service models and the knowledge of mobility patterns of vehicles. To understand the historic mobility patterns of vehicles, we identified a gap in existing vehicle mobility-related research where the relative mobility of a group of moving vehicles has not been studied. In our work, we find a means to bridge this gap using available macroscopic mobility models and simulating calibrated microscopic mobility models. We also introduce distributed and flexible service models that are adaptable to the dynamics of the moving networks. Similarly, we introduced mobility as an intrinsic part of the infrastructure as it directly affects the availability of the service. Our work aims at increasing the probability of successful placement and execution of services without over-utilising resources. Our work has presented a way to extend the mobile edge computing resources further to VFC, thereby providing more sensing, processing and communication resources to the service providers.

This chapter focuses on first summarising the research work and technical results achieved in this dissertation. In Section 7.1, a short summary of each chapter is presented. In Section 7.2, we present the short-term and long-term future research directions of this work.

7.1 Summary

Chapter 1 of the dissertation introduces the core research idea of VFC where vehicles are proposed to be used as infrastructure. It highlights the motivation behind using the under-utilised sensing, processing and communication capacity of vehicles in urban traffic. The chapter also briefly highlights the tools required to leverage the moving vehicular network for service provisioning. The chapter outlines the research hypothesis and the breakdown of

the research objective into four research questions. The chapter maps each research question to a contribution and describes the organisation of the dissertation.

In the next chapter, Chapter 2, we first present in-depth background of technologies like vehicular computing, fog and MEC. The chapter highlights the evolution of VANETs as a network for disseminating safety information to the IoV ecosystem today that is envisioned as a platform for service deployment. The chapter presents the VFC architecture and urban scenarios where services can be deployed on moving infrastructure. We also present key challenges involved in application deployment on the IoV. The chapter also presents the state-of-the-art in fog computing, VCC, VFC, programming models for future internet applications and task allocation and scheduling schemes in vehicular fog. The chapter then presents the key challenges and limitations identified in leveraging vehicles as sensing and processing infrastructure, after a detailed review of the recent works. We identified gaps in the literature and mapped each research gap to a corresponding research question.

Chapter 3, focuses on the mobility behaviour of vehicles in urban traffic conditions to support the service placement decision on these moving vehicular clusters. The chapter aims to address RQ1 and focuses on novel methods to leverage the mobility patterns of vehicles to make the service placement more robust. This study is crucial to utilise moving vehicles for efficient resource utilisation. We first highlight the predictability of vehicular flow using an MVLR model that accurately predicts vehicular flow at intersections. We compared the accuracy of our model to competing schemes like random forest and the ARIMA model for time-series forecasting. The chapter highlights the motivation behind using moving vehicles as infrastructure with a special study on the predictability and density of vehicles using real vehicular traffic data. We also introduced methods to calculate the aggregate communication and computation capacity of such clusters. To the best of our knowledge, our work is one of the first few attempts to study the relative mobility states of vehicles with the objective of distributed service placement. In the next part of the chapter, the terminology of the system model, the network topology and the distributed service model used in the rest of the research work are introduced. Instead of placing static services, our work focuses on scaling tasks to multiple task instances placed on nearby vehicle nodes. This model thus not only handles the processing of multiple video streams but also provides a reliable service placement in case links and nodes fail due to the dynamic vehicular network.

The fourth chapter aims to answer RQ2 by introducing the methodology to scale and place distributed services on vehicle clusters, using the knowledge of mobility patterns of vehicles. To use moving vehicles as a potential infrastructure, we first introduce the distributed service model where services are broken down into multiple data-dependent tasks. Each task is modelled to be scaled to multiple TIs to increase the resilience of the service and to deploy

more processing resources to process the collected video from multiple sources. We also modelled the service placement problem as a bi-objective, constrained optimisation problem to optimise resource usage in service placement. We intrinsically formulated mobility as part of the model by using CCP as a parameter to weigh the vehicle node's resources while making the service placement plan. This results in selecting those vehicles for service deployment that have a historic mobility pattern to follow the chosen trajectory. This selected cluster of vehicles is chosen to stay together for the duration of service execution till completion. Selecting those vehicular nodes that are less probable to stay with other vehicles for the duration of service execution will result in service failure and reconfiguration, resulting in more overhead costs. The chapter then presents detailed simulation results of placing two applications with different resource profiles on the vehicle cluster. We present our results for resource-rich and resource-poor vehicle clusters with stable and unstable mobility states. Our approach outperforms a naive solution and a clustering-based solution in the literature.

The fifth chapter, addresses RQ3 by providing a more detailed service placement model along with heuristic based solution for node selection, service scaling and placement. We first introduce two distributed applications using the distributed service template introduced in our work. The first application is a data collection service where vehicles act primarily as 'moving sensors' and carry out pre-processing and compressing the collected data. The processed data is then sent to edge or cloud servers for more complex processing. These novel applications can be used to collect data regarding traffic conditions and to analyse the usage pattern of roadside cafes, libraries and gas stations by different commuters. The second application is an object detection application focusing on detecting vulnerable pedestrians. This kind of application carries most of the data processing on the vehicle cluster as the application requires near real-time decision-making. As such applications have local context and scope, we suggest that such applications should not offload the processing of data to edge or cloud servers.

The chapter then introduces a more detailed service placement optimisation problem, with constraints related to flow capacity and flow order for the distributed applications. We then introduce a community-detection-based vehicular node selection and graph-based service placement heuristics for vehicular cluster selection and service placement. We evaluate the performance of the service placement heuristic using other resource utilisation measures like the number of scaled TIs and the average hop count for placing the distributed services. We then consider a QoS-level parameter called service time to analyse how our approach performs compared to a mobile-edge placement approach. We also emulate the service placement using a Fog simulator. We analyse how the node selection approach performs in terms of the life of the selected cluster. We introduced a betweenness centrality-based

resilience score to evaluate the performance of the chosen cluster, in terms of the quality of nodes that make it to the end of the service execution time.

Our approach outperforms the ILP because it generates placement plans more quickly but with similar resource costs, and outperforms the baseline first-fit solution, because the mobility-aware strategy ensures that the cluster cohesion is higher, increasing the system's resilience. We also compared our VFC approach to edge computing, where the vehicles are not used for processing and just forward the data to the RSU and cloud for processing. Our placement technique results in better worst-case performance, with a much lower maximum service time which is a measure of the time taken in service execution, including both processing and link latency.

In the last chapter, we use video data collected from an intersection-based testbed in Chattanooga, Tennessee to apply object-detection applications similar to the ones proposed in the study. This chapter aims to solve RQ4. We test two different object detection schemes including YOLO and Faster R-CNN Resnet to detect different classes of objects including cars, trucks, pedestrians and cyclists. We then introduce a distributed FL-based scheme, analogous to the service model presented in our work. The distributed FL scheme focuses on reducing the communication rounds during the model's training by introducing local aggregators, thus making it more resource-efficient.

7.2 Future Works

To conclude the dissertation, we present some future directions of the research work. We first highlight some immediate future research direction that is a direct and effective extension of the current research. We then highlight some long-term goals that we aim to achieve in the future.

7.2.1 Next steps of research

Service Reconfiguration We first aim to add reconfiguration of services when tasks or services fail in the service placement model. Instead of re-configuring the entire service, only those tasks that fail are replaced in the linear service instance graph. The re-configuration cost can be minimised using our mobility modelling for service selection wherein new vehicular nodes can be selected for service replacement. We aim to use deep learning-based approaches where an agent can learn the best resource allocation schemes that can migrate failed tasks to new nodes and find data communication paths to these new nodes.

Decentralised control As we consider moving vehicle clusters for service provisioning, instead of coordinating the lifecycle of services using a central controller, we aim for leaving vehicle nodes to themselves offload tasks to the nearest nodes in the cluster. This will reduce the cost of managing and re-configuring services. As an extension of this work, service reconfiguration and service-state take-back scheme can be added to Algorithm 5.1.

Relevant QoS metrics for dynamic networks We have aimed at introducing measures to detect the quality of vehicular nodes that made it till the end of service execution using measures like resilience score. The resilience score checks for how many well-connected vehicular nodes are still part of the selected cluster using a betweenness centrality of nodes in the graph. Other measures can be introduced to assess the performance of such moving networks and the services deployed on these moving networks. For example, Quality of Information (QoI) can be used to analyse the quality of data collected from such services. Similarly, from a business point of view, the historical performance of vehicles can be studied using measures like accepting and successfully completing services. Such measures can add a confidence score to each participating vehicular node. The confidence score will help in making better node selection decisions. The node selection will not be based on just the historical mobility patterns of vehicles but also on the number of times the vehicles delivered the service successfully. The successful service completion is based on both the available communication and processing resources as well as if the vehicles takes the predicted trajectory each time, similar to its historic mobility pattern. This score can be used by clients/agents requesting the vehicles to collect and process data to make more efficient offloading decisions.

Improved Federated Learning-based application for pedestrian counting : We aim to make the distributed federated learning-based framework for pedestrian detection, introduced in Section 6.1, more accurate. The distributed federated learning-based framework follows the template of the distributed service model introduced in Section 3.7.3. The FL-based frameworks often require high communication and computation capacity. We thus introduced the concept of distributed FL where several local aggregators are used instead of a centralised server for global aggregation. While the distributed FL framework will be resource efficient, it is not as accurate as the classic FL schemes and other object detection schemes mentioned in our work. We aim to improve the training efficiency and model accuracy by optimising the number of local aggregators and communication rounds required for local and global updates. We

also aim to improve the object detection application by collecting more data from the intersection-based test-bed mentioned in Section 6. Collecting more data on vulnerable pedestrians, cyclists and jaywalkers, the object detection application can be tested for detection accuracy.

7.2.2 Longer-term research direction

Multiple RSUs to manage the lifecycle of the CN and the vehicle cluster In this work, we consider a single RSU that receives requests from clients and manages the state of the cluster. This work can be extended to multiple RSUs that can manage the lifecycle of both the CN and the vehicle cluster as the cluster moves along a freeway or road segment. This would require the handover of the vehicle cluster and service states between different RSUs, but will also help in making the service deployments more robust.

Multiple clusters for developing a vehicular fog marketplace This research can be extended to initiate and manage multiple clusters for executing multiple services. This will give clients and agencies that want to offload service deployment on the vehicle clusters an option to choose from the multiple clusters based on the cost, the confidence scores of participating vehicles and the quality of the data collected. This will lead to the formation of a vehicular fog marketplace where services can be advertised to users and service providers. With the requirement of using big data in AI and ML algorithms, there are not many open datasets available for vehicular sensor data. The collection of such data will lead to new cross-sectorial services, to actualise the goal of smart cities including safer roads and energy-efficient means of transport. There have been some works in literature, introducing the concept of vehicular big data marketplace [Pillmann et al., 2017]. This concept can be extended to include both data collection and processing service's marketplace.

More applications related to autonomous driving This work can be extended to other novel applications like the management of self-driven cars that can share information with neighbouring cars to make driving safer. For example, the generation of a 3D road map for autonomous cars, can be shared and made more accurate by collecting data from a fleet of autonomous vehicles moving together. These vehicular clusters can also be used for surveillance purposes with different sense-process-actuate cycles, according to the use case.

Bibliography

Aazam, M. and Huh, E. (2015). Fog computing micro datacenter based dynamic resource estimation and pricing model for iot. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, pages 687–694.

Aazam, M. and Huh, E. N. (2016). Fog computing: The cloud-iot/ioe middleware paradigm. *IEEE Potentials*, 35(3):40–44.

Aazam, M., Zeadally, S., and Harras, K. A. (2018). Fog computing architecture, evaluation, and future research directions. *IEEE Communications Magazine*, 56(5):46–52.

Abbas, N., Zhang, Y., Taherkordi, A., and Skeie, T. (2018). Mobile edge computing: A survey. *IEEE Internet of Things Journal*, 5(1):450–465.

Abboud, K., Omar, H. A., and Zhuang, W. (2016). Interworking of dsrc and cellular network technologies for v2x communications: A survey. *IEEE Transactions on Vehicular Technology*, 65(12):9457–9470.

Abuelela, M. and Olariu, S. (2010). Taking vanet to the clouds. In *Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia*, MoMM '10, page 6–13, New York, NY, USA. Association for Computing Machinery.

Al-Heety, O. S., Zakaria, Z., Ismail, M., Shakir, M. M., Alani, S., and Alsariera, H. (2020). A comprehensive survey: Benefits, services, recent works, challenges, security, and use cases for sdn-vanet. *IEEE Access*, 8:91028–91047.

Alghamdi, T., Elgazzar, K., Bayoumi, M., Sharaf, T., and Shah, S. (2019). Forecasting traffic congestion using arima modeling. In *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, pages 1227–1232.

Ang, L., Seng, K. P., Ijemaru, G. K., and Zungeru, A. M. (2019). Deployment of iov for smart cities: Applications, architecture, and challenges. *IEEE Access*, 7:6473–6492.

Author, U. (2020). An Introduction to the Calfornia Department of Transportation Performance Measurement System (PeMS). https://pems.dot.ca.gov/. [Online; accessed 05-April-2022].

Barcelo, M., Correa, A., Llorca, J., Tulino, A. M., Vicario, J. L., and Morell, A. (2016). Iot-cloud service optimization in next generation smart environments. *IEEE Journal on Selected Areas in Communications*, 34(12):4077–4090.

Benkerdagh, S. and Duvallet, C. (2019). Cluster-based emergency message dissemination strategy for vanet using v2v communication. *International Journal of Communication Systems*, 32(5):e3897. e3897 dac.3897.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008.

Bonomi, F., Milito, R., Natarajan, P., and Zhu, J. (2014). *Fog Computing: A Platform for Internet of Things and Analytics*, pages 169–186. Springer International Publishing, Cham.

Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012). Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, page 13–16, New York, NY, USA. Association for Computing Machinery.

Boukerche, A. and Grande, R. E. D. (2018). Vehicular cloud computing: Architectures, applications, and mobility. *Computer Networks*, 135:171 – 189.

Chen, J., Mao, G., Li, C., Liang, W., and Zhang, D. (2018). Capacity of cooperative vehicular networks with infrastructure support: Multiuser case. *IEEE Transactions on Vehicular Technology*, 67(2):1546–1560.

Chen, M. and Hao, Y. (2018). Task offloading for mobile edge computing in software defined ultra-dense network. *IEEE Journal on Selected Areas in Communications*, 36(3):587–597.

Chen, X., Jiao, L., Li, W., and Fu, X. (2016). Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, 24(5):2795–2808.

Dai, Y., Xu, D., Maharjan, S., Qiao, G., and Zhang, Y. (2019). Artificial intelligence empowered edge computing and caching for internet of vehicles. *IEEE Wireless Communications*, 26(3):12–18.

de Mendonça, F. F., Kokkinogenis, Z., Dias, K. L., d'Orey, P. M., and Rossetti, R. J. (2022). The trade-offs between fog processing and communications in latency-sensitive vehicular fog computing. *Pervasive Mob. Comput.*, 84(C).

Dos Reis Fontes, R., Campolo, C., Esteve Rothenberg, C., and Molinaro, A. (2017). From theory to experimental evaluation: Resource management in software-defined vehicular networks. *IEEE Access*, 5.

Dräxler, S. and Karl, H. (2017). Specification, composition, and placement of network services with flexible structures. *International Journal of Network Management*, 27(2):e1963. e1963 nem.1963.

Dräxler, S., Schneider, S., and Karl, H. (2018). Scaling and placing bidirectional services with stateful virtual and physical network functions. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 123–131.

Du, H., Leng, S., Wu, F., Chen, X., and Mao, S. (2020). A new vehicular fog computing architecture for cooperative sensing of autonomous driving. *IEEE Access*, 8:10997–11006.

Feng, J., Liu, Z., Wu, C., and Ji, Y. (2017). Ave: Autonomous vehicular edge computing framework with aco-based scheduling. *IEEE Transactions on Vehicular Technology*, 66(12):10660–10675.

Ge, S., Cheng, M., and Zhou, X. (2020). Interference aware service migration in vehicular fog computing. *IEEE Access*, 8:84272–84281.

Gerla, M. (2012). Vehicular cloud computing. In *2012 The 11th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, pages 152–155.

Giang, N. K., Blackstock, M., Lea, R., and Leung, V. C. (2015). Developing iot applications in the fog: A distributed dataflow approach. In *2015 5th International Conference on the Internet of Things (IOT)*, pages 155–162.

Goudarzi, M., Wu, H., Palaniswami, M., and Buyya, R. (2021). An application placement technique for concurrent iot applications in edge and fog computing environments. *IEEE Transactions on Mobile Computing*, 20(4):1298–1311.

Grossglauser, M. and Tse, D. (2001). 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)*, volume 3, pages 1360–1369 vol.3.

Gu, L., Zeng, D., and Guo, S. (2013). Vehicular cloud computing: A survey. In *2013 IEEE Globecom Workshops (GC Wkshps)*, pages 403–407.

Gu, Y., Chang, Z., Pan, M., Song, L., and Han, Z. (2018). Joint radio and computational resource allocation in iot fog computing. *IEEE Transactions on Vehicular Technology*, 67(8):7475–7484.

Hagenauer, F., Sommer, C., Higuchi, T., Altintas, O., and Dressler, F. (2017). 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*, CarSys '17, page 31–35, New York, NY, USA. Association for Computing Machinery.

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

Ho, I. W., Chau, S. C., Magsino, E. R., and Jia, K. (2020). Efficient 3d road map data exchange for intelligent vehicles in vehicular fog networks. *IEEE Transactions on Vehicular Technology*, 69(3):3151–3165.

Hong, K., Lillethun, D., Ramachandran, U., Ottenwälder, B., and Koldehofe, B. (2013a). Mobile fog: A programming model for large-scale applications on the internet of things. In *Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing*, MCC '13, page 15–20, New York, NY, USA. Association for Computing Machinery.

Hong, K., Lillethun, D., Ramachandran, U., Ottenwälder, B., and Koldehofe, B. (2013b). Mobile fog: A programming model for large-scale applications on the internet of things. In *Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing*, MCC '13, pages 15–20, New York, NY, USA. ACM.

Hou, X., Li, Y., Chen, M., Wu, D., Jin, D., and Chen, S. (2016). Vehicular fog computing: A viewpoint of vehicles as the infrastructures. *IEEE Transactions on Vehicular Technology*, 65(6):3860–3873.

Hu, P., Dhelim, S., Ning, H., and Qiu, T. (2017). Survey on fog computing: architecture, key technologies, applications and open issues. *Journal of Network and Computer Applications*, 98:27–42.

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

Huang, C., Lu, R., and Choo, K.-K. R. (2017). Vehicular fog computing: Architecture, use case, and security and forensic challenges. *IEEE Communications Magazine*, 55(11):105–111.

Hussein, M. K. and Mousa, M. H. (2020). Efficient task offloading for iot-based applications in fog computing using ant colony optimization. *IEEE Access*, 8:37191–37201.

Intharawijitr, K., Iida, K., and Koga, H. (2016). Analysis of fog model considering computing and communication latency in 5g cellular networks. In *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 1–4.

Jang, I., Choo, S., Kim, M., Pack, S., and Dan, G. (2017). The software-defined vehicular cloud: A new level of sharing the road. *IEEE Vehicular Technology Magazine*, 12(2):78–88.

Kanaka Sri Shalini, C. M., Roopa, Y. M., and Devi, J. S. (2019). Fog computing for smart cities. In *2019 International Conference on Communication and Electronics Systems (ICCES)*, pages 912–916.

Keshari, N., Singh, D., and Maurya, A. K. (2022). A survey on vehicular fog computing: Current state-of-the-art and future directions. *Vehicular Communications*, page 100512.

Khan, A. u. R., Othman, M., Madani, S. A., and Khan, S. U. (2014). A survey of mobile cloud computing application models. *IEEE Communications Surveys Tutorials*, 16(1):393–413.

Lee, S.-S. and Lee, S. (2020). Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information. *IEEE Internet of Things Journal*, 7(10):10450–10464.

Lera, I., Guerrero, C., and Juiz, C. (2019). Availability-aware service placement policy in fog computing based on graph partitions. *IEEE Internet of Things Journal*, 6(2):3641–3651.

Lera, I., Guerrero, C., and Juiz, C. (2019). Yafs: A simulator for iot scenarios in fog computing. *IEEE Access*, 7:91745–91758.

Li, L., Fan, Y., Tse, M., and Lin, K.-Y. (2020a). A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854.

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. (2020b). Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60.

Li, X., Huang, X., Li, C., Yu, R., and Shu, L. (2019). Edgecare: Leveraging edge computing for collaborative data management in mobile healthcare systems. *IEEE Access*, 7:22011–22025.

Liang, J., Zhang, J., Leung, V. C., and Wu, X. (2021). Distributed information exchange with low latency for decision making in vehicular fog computing. *IEEE Internet of Things Journal*, pages 1–1.

Lienert, P. (2015). 12 million driverless cars to be on the road by 2035 -study. https://www.reuters.com/article/autos-bcg-autonomous-idUSL1N0UN2GQ20150108. Accessed: 2021-09-30.

Lin, D., Kang, J., Squicciarini, A., Wu, Y., Gurung, S., and Tonguz, O. (2017). Mozo: A moving zone based routing protocol using pure v2v communication in vanets. *IEEE Transactions on Mobile Computing*, 16(5):1357–1370.

Liu, Y. and Wu, H. (2017). Prediction of road traffic congestion based on random forest. In *2017 10th International Symposium on Computational Intelligence and Design (ISCID)*, volume 2, pages 361–364.

Liu, Z., Dai, P., Xing, H., Yu, Z., and Zhang, W. (2021). A distributed algorithm for task offloading in vehicular networks with hybrid fog/cloud computing. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–14.

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

Maglaras, L. A. and Katsaros, D. (2016). Social clustering of vehicles based on semi-markov processes. *IEEE Transactions on Vehicular Technology*, 65(1):318–332.

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

MEOLA, A. (2020). How 5g & iot technologies are driving the connected smart vehicle industry. https://www.businessinsider.com/iot-connected-smart-cars?r=US&IR=T. Accessed: 2021-09-30.

Misra, S. and Saha, N. (2019). Detour: Dynamic task offloading in software-defined fog for iot applications. *IEEE Journal on Selected Areas in Communications*, 37(5):1159–1166.

Mohd Zaini, K., Mohd Shariff, A. R., and Shi, Z. (2016). A calculation of wlan dwell time model for wireless network selection. *JOURNAL OF TELECOMMUNICATION, ELECTRONIC AND COMPUTER ENGINEERING*, 8:73–76.

Naha, R. K., Garg, S., Georgakopoulos, D., Jayaraman, P. P., Gao, L., Xiang, Y., and Ranjan, R. (2018). Fog computing: Survey of trends, architectures, requirements, and research directions. *IEEE Access*, 6:47980–48009.

Nazmudeen, M. S. H., Wan, A. T., and Buhari, S. M. (2016). Improved throughput for power line communication (plc) for smart meters using fog computing based data aggregation approach. In *2016 IEEE International Smart Cities Conference (ISC2)*, pages 1–4.

Newman, M. E. J. (2004). Analysis of weighted networks. *Phys. Rev. E*, 70:056131.

Ni, J., Zhang, A., Lin, X., and Shen, X. S. (2017). Security, privacy, and fairness in fog-based vehicular crowdsensing. *IEEE Communications Magazine*, 55(6):146–152.

Ni, J., Zhang, K., Yu, Y., Lin, X., and Shen, X. S. (2018). Providing task allocation and secure deduplication for mobile crowdsensing via fog computing. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1.

Ni, L., Zhang, J., Jiang, C., Yan, C., and Yu, K. (2017). Resource allocation strategy in fog computing based on priced timed petri nets. *IEEE Internet of Things Journal*, 4(5):1216–1228.

Ning, Z., Huang, J., and Wang, X. (2019). Vehicular fog computing: Enabling real-time traffic management for smart cities. *IEEE Wireless Communications*, 26(1):87–93.

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

Noorani, N. and Seno, S. A. H. (2018). Routing in vanets based on intersection using sdn and fog computing. In *2018 8th International Conference on Computer and Knowledge Engineering (ICCKE)*, pages 339–344.

Oueis, J., Strinati, E. C., and Barbarossa, S. (2015a). The fog balancing: Load distribution for small cell cloud computing. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pages 1–6.

Oueis, J., Strinati, E. C., Sardellitti, S., and Barbarossa, S. (2015b). Small cell clustering for efficient distributed fog computing: A multi-user case. In *2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*, pages 1–5.

Phan, L.-A., Nguyen, D.-T., Lee, M., Park, D.-H., and Kim, T. (2021). Dynamic fog-to-fog offloading in sdn-based fog computing systems. *Future Generation Computer Systems*, 117:486–497.

Pillmann, J., Wietfeld, C., Zarcula, A., Raugust, T., and Alonso, D. C. (2017). Novel common vehicle information model (cvim) for future automotive vehicle big data marketplaces. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1910–1915.

Qiao, G., Leng, S., Zhang, K., and He, Y. (2018). Collaborative task offloading in vehicular edge multi-access networks. *IEEE Communications Magazine*, 56(8):48–54.

Rathore, M. M., Paul, A., Rho, S., Khan, M., Vimal, S., and Shah, S. A. (2021). Smart traffic control: Identifying driving-violations using fog devices with vehicular cameras in smart cities. *Sustainable Cities and Society*, 71:102986.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks.

Report, T. (2009). Intelligent transport systems (its);vehicular communications;basic set of applications;definitions. https://www.etsi.org/deliver/etsi_tr/102600_102699/102638/01.01.01_60/tr_102638v010101p.pdf. Accessed: 2021-09-30.

Saha, R., Misra, S., and Deb, P. K. (2021). Fogfl: Fog-assisted federated learning for resource-constrained iot devices. *IEEE Internet of Things Journal*, 8(10):8456–8463.

Salahuddin, M. A., Al-Fuqaha, A., and Guizani, M. (2015). Software-defined networking for rsu clouds in support of the internet of vehicles. *IEEE Internet of Things Journal*, 2(2):133–144.

Sarkar, S., Chatterjee, S., and Misra, S. (2018). Assessment of the suitability of fog computing in the context of internet of things. *IEEE Transactions on Cloud Computing*, 6(1):46–59.

Sarkar, S. and Misra, S. (2016). Theoretical modelling of fog computing: a green computing paradigm to support iot applications. *Iet Networks*, 5(2):23–29.

Shah, S. A. A., Ahmed, E., Imran, M., and Zeadally, S. (2018). 5g for vehicular communications. *IEEE Communications Magazine*, 56(1):111–117.

Sharma, K., Butler, B., and Jennings, B. (2021). Scaling and placing distributed services on vehicle clusters in urban environments. *arXiv*.

Sharma, K., Butler, B., and Jennings, B. (2022a). Graph-based heuristic solution for placing distributed video processing applications on moving vehicle clusters. *IEEE Transactions on Network and Service Management*, pages 1–1.

Sharma, K., Butler, B., and Jennings, B. (2022b). Scaling and placing distributed services on vehicle clusters in urban environments. *IEEE Transactions on Services Computing*, pages 1–1.

Sharma, K., Butler, B., Jennings, B., Kennedy, J., and Loomba, R. (2018). 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)*, pages 1800–1806.

Shi, H., Chen, N., and Deters, R. (2015). Combining mobile and fog computing: Using coap to link mobile device clouds with fog computing. In *2015 IEEE International Conference on Data Science and Data Intensive Systems*, pages 564–571.

Shi, L., Butler, B., Botvich, D., and Jennings, B. (2013). Provisioning of requests for virtual machine sets with placement constraints in iaas clouds. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 499–505.

Singh, P. K., Nandi, S. K., and Nandi, S. (2019). A tutorial survey on vehicular communication state of the art, and future research directions. *Vehicular Communications*, 18:100164.

Skarlat, O., Nardelli, M., Schulte, S., Borkowski, M., and Leitner, P. (2017). Optimized iot service placement in the fog. *Service Oriented Computing and Applications*, 11(4):427–443.

Skarlat, O., Schulte, S., Borkowski, M., and Leitner, P. (2016). Resource provisioning for iot services in the fog. In *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, pages 32–39.

Sookhak, M., Yu, F. R., He, Y., Talebian, H., Sohrabi Safa, N., Zhao, N., Khan, M. K., and Kumar, N. (2017). Fog vehicular computing: Augmentation of fog computing using vehicular cloud computing. *IEEE Vehicular Technology Magazine*, 12(3):55–64.

Suh, J., Chae, H., and Yi, K. (2018). Stochastic model-predictive control for lane change decision of automated driving vehicles. *IEEE Transactions on Vehicular Technology*, 67(6):4771–4782.

Tan, K., Feng, L., Dán, G., and Törngren, M. (2022). Decentralized convex optimization for joint task offloading and resource allocation of vehicular edge computing systems. *IEEE Transactions on Vehicular Technology*, pages 1–15.

Tang, C., Xia, S., Li, Q., Chen, W., and Fang, W. (2021). Resource pooling in vehicular fog computing. *Journal of Cloud Computing*, 10(1):1–14.

Teo, H. M. and Kadir, W. M. W. (2006). A comparative study of interface design approaches for service-oriented software. In *2006 13th Asia Pacific Software Engineering Conference (APSEC'06)*, pages 147–156.

Thakur, A. and Malekian, R. (2019). Fog computing for detecting vehicular congestion, an internet of vehicles based approach: A review. *IEEE Intelligent Transportation Systems Magazine*, 11(2):8–16.

Ucar, S., Ergen, S. C., and Ozkasap, O. (2016). Multihop-cluster-based ieee 802.11p and lte hybrid architecture for vanet safety message dissemination. *IEEE Transactions on Vehicular Technology*, 65(4):2621–2636.

Wang, D., Liu, Z., Wang, X., and Lan, Y. (2019). Mobility-aware task offloading and migration schemes in fog computing networks. *IEEE Access*, 7:43356–43368.

Wang, D., Zhang, Q., Wu, S., Li, X., and Wang, R. (2016). Traffic flow forecast with urban transport network. In *2016 IEEE International Conference on Intelligent Transportation Engineering (ICITE)*, pages 139–143.

Wang, P., Liu, S., Ye, F., and Chen, X. (2018). A Fog-based Architecture and Programming Model for IoT Applications in the Smart Grid. *arXiv e-prints*, page arXiv:1804.01239.

Xiao, X., Hou, X., Chen, X., Liu, C., and Li, Y. (2019). Quantitative analysis for capabilities of vehicular fog computing. *Information Sciences*, 501:742–760.

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

Xu, J., Palanisamy, B., Ludwig, H., and Wang, Q. (2017). Zenith: Utility-aware resource allocation for edge computing. In *2017 IEEE International Conference on Edge Computing (EDGE)*, pages 47–54.

Yadav, R., Zhang, W., Kaiwartya, O., Song, H., and Yu, S. (2020). Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing. *IEEE Transactions on Vehicular Technology*, 69(12):14198–14211.

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

Yerabolu, S., Kim, S., Gomena, S., Li, X., Patel, R., Bhise, S., and Aryafar, E. (2019). Deepmarket: An edge computing marketplace with distributed tensorflow execution capability. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 32–37.

Yi, S., Hao, Z., Qin, Z., and Li, Q. (2015). Fog computing: Platform and applications. In *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, pages 73–78.

Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J., and Jue, J. P. (2019). All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, 98:289 – 330.

Yu, R., Xue, G., and Zhang, X. (2018). Application provisioning in fog computing-enabled internet-of-things: A network perspective. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 783–791.

Zeng, D., Gu, L., Guo, S., Cheng, Z., and Yu, S. (2016). Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. *IEEE Transactions on Computers*, 65(12):3702–3712.

Zhang, J., Huang, X., and Yu, R. (2020). Optimal task assignment with delay constraint for parked vehicle assisted edge computing: A stackelberg game approach. *IEEE Communications Letters*, 24(3):598–602.

Zhang, W., Zhang, Z., and Chao, H.-C. (2017). Cooperative fog computing for dealing with big data in the internet of vehicles: Architecture and hierarchical resource management. *IEEE Communications Magazine*, 55(12):60–67.

Zhang, Y., Li, C., Luan, T. H., Fu, Y., Shi, W., and Zhu, L. (2019a). A mobility-aware vehicular caching scheme in content centric networks: Model and optimization. *IEEE Transactions on Vehicular Technology*, 68(4):3100–3112.

Zhang, Y., Wang, C.-Y., and Wei, H.-Y. (2019b). Parking reservation auction for parked vehicle assistance in vehicular fog computing. *IEEE Transactions on Vehicular Technology*, 68(4):3126–3139.

Zhao, J., Li, Q., Gong, Y., and Zhang, K. (2019). Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks. *IEEE Transactions on Vehicular Technology*, 68(8):7944–7956.

Zhou, Z., Liao, H., Wang, X., Mumtaz, S., and Rodriguez, J. (2020). When vehicular fog computing meets autonomous driving: Computational resource management and task offloading. *IEEE Network*, 34(6):70–76.

Zhu, C., Pastor, G., Xiao, Y., Li, Y., and Ylae-Jaeaeski, A. (2018). Fog following me: Latency and quality balanced task allocation in vehicular fog computing. In *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9.

Zhu, C., Pastor, G., Xiao, Y., and Ylajaaski, A. (2018). Vehicular fog computing for video crowdsourcing: Applications, feasibility, and challenges. *IEEE Communications Magazine*, 56(10):58–63.

Zhu, C., Tao, J., Pastor, G., Xiao, Y., Ji, Y., Zhou, Q., Li, Y., and Ylä-Jääski, A. (2019). Folo: Latency and quality optimized task allocation in vehicular fog computing. *IEEE Internet of Things Journal*, 6(3):4150–4161.