Adaptive Q-learning-supported Resource Allocation Model in Vehicular Fogs

Md Tahmid Hossain

Submitted in partial fulfilment
of the requirements for the degree of

Master of Science

Department of Computer Science
Brock University
St. Catharines, Ontario

©*Md Tahmid Hossain,* 2021

Abstract

Urban computing has become a significant driver in supporting the delivery and sharing of services, being a strong ally to intelligent transportation. Smart vehicles present computing and communication capabilities that allow them to enable many autonomous vehicular safety and infotainment applications. Vehicular Cloud Computing (VCC) has already proven to be a technology shifting paradigm harnessing the computation resources from on board units from vehicles to form clustered computing units to solve real world computing problems. However, with the rise of vehicular application use and intermittent network conditions, VCC exhibits many drawbacks. Vehicular Fog computing appears as a new paradigm in enabling and facilitating efficient service and resource sharing in urban environments. Several vehicular resource management works have attempted to deal with the highly dynamic vehicular environment following diverse approaches, e.g. MDP, SMDP, and policy-based greedy techniques. However, the high vehicular mobility causes several challenges compromising consistency, efficiency, and quality of service. RL-enabled adaptive vehicular Fogs can deal with the mobility for properly distributing load and resources over Fogs. Thus, we propose a mobility-based cloudlet dwell time estimation method for accurately estimating vehicular resources in a Fog. Leveraging the CDT estimation model, we devise an adaptive and highly dynamic resource allocation model using mathematical formula for Fog selection, and reinforcement learning for iterative review and feedback mechanism for generating optimal resource allocation policy.

Acknowledgements

I would like to thank my family for supporting me along my studies, my supervisor, Dr. De Grande, for his guidance, and friends and colleagues for helping and accompanying me along my MSc thesis studies. I am grateful to Brock University for providing me with Dr. Raymond and Mrs. Sachi Moriyama Graduate Fellowship and DGS Spring Research Fellowship.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Vehicular networking has gained significant interest as vehicles are being equipped with computerized modules and wireless devices, carrying and distributing information among other vehicles and stationary roadside units (RSU), and exploiting advanced wireless technologies. Vehicular ad-hoc networks (VANET) [7] allow exchanging sensitive information among nearby vehicles and roadside control nodes, e.g. weather conditions and accidents, to improve vehicle traffic efficiency building up towards intelligent transportation systems (ITS).

With the recent surge in the number of smart vehicles, traditional VANETs face several challenges due to poor connectivity, scalability, and its nature of inflexibility [31]. Vehicular Cloud computing (VCC) is developing fast, as it can overcome the drawbacks of VANETs by providing low-cost services to vehicles, and it is capable of managing road traffic efficiently by instantly using onboard units. VCC can be considered as a group of autonomous vehicles collaboratively sharing resources for computing, sensing, and communication [24]. Collaboration in VCC can be implemented following a service-oriented fashion/paradigm, e.g. network as a service [23], storage as a service [3], cooperation as a service [23], and computing as a service [10]. Such services enable a series of vehicular and ITS applications, e.g. traffic route optimization [15], parking lot data centres, and safety messaging [24].

In vehicular Cloud computing, the mobility of the nodes is very high, and the network topology frequently changes, in addition to the intermittent wireless network connection. As a result, issues appear, for example, authentication problems, service delays, and poor system integrity, which cause low quality of service (Qos) or denial of service (Dos). With the widespread use of vehicular networks and the rapid increase in vehicles with onboard computational units, the amount of data that vehicles must process, e.g. entertainment, traffic, and driving safety details, is surging rapidly. As

most vehicles are equipped with limited computation power, with increased data- and computation-heavy applications, vehicles need to transmit to and receive back data from Cloud computing systems for processing. As a result, effectively improving the vehicular Cloud computing resource utilization to improve the overall processing efficiency has gained much research interest.

The shortcomings of VCC in high mobility and intermittent network conditions have ushered new technologies, e.g. vehicular edge computing (VEC) [11], vehicular Fog computing (VFC) [13], and mobile vehicular cloudlet (MVC) [37], which enable unlocking the full potential of VCC. With high-level mobility support, device heterogeneity, flexible location, and low latency, a few VCC drawbacks can be overlooked [26]. VFC follows the same paradigm of Fog computing [8] in the vehicular environment. Vehicles are grouped according to proximity to form cloudlets [30] which serves as Fogs, and AP Edge computing nodes that deal with the connectivity with the Cloud. A mobile cloudlet follows a peer-to-peer communication model [16] termed as a group of mobile devices in close proximity connected by short-range communications. Advantages of Fog/edge computing include improved response time [29], scalability [29], security and privacy [2], and Cloud outage control.

Resource management has been dealt with in several recent works with different approaches. Some works have focused on architectural aspects, e.g. P2P resource scheduling [21,22]. Other works have directly targeted resource allocation. For example, game theory approaches [41] have attempted to aid in the resource allocation process. Markov decision process (MDP) has been utilized to formulate a reinforcement learning-based resource allocation problem [28]. Similarly, Semi-Markov decision process (SMDP) model has been used to solve the resource allocation problem [18]. Through new VEC network architectures [19], SMDP and Q-learning-based reinforcement learning methods were utilized for optimal offloading and resource allocation process. Optimization models have been proposed to deal with the allocation problem where some works explored resource allocation policies [25] based on influence factors and assigned weights in vehicular Fog environment. However, the influence factors and their weights are pre-defined and fixed. They also may not perform well in heterogeneous application environments. In addition, much of the works mainly assumed ideal scenarios in the vehicular environment and proved their results on pre-recorded vehicle data.

1.1 Motivation

One of the most important issues a vehicular cloud system faces is the latency caused by limited computation power and bandwidth. While technologies e.g. Fog computing helps reduce the latency and brings the power of cloud computing near the mobile nodes, the necessity of instantaneous decision making is still a major issue to solve in vehicular environment. Whenever an application request arises within a fog computing node, the node then has to process that request within minimal delay to improve the quality of service. On the other hand, the delay in decision making causes service denial for latency prone applications.

1.2 Objective

We propose a reinforcement learning based dynamic resource allocation model based on an estimation method of cloudlet dwell time and corresponding available resources for vehicular Fog computing. To accomplish this, we design a precise estimation model for predicting the trajectory of vehicles relative to Fogs/RSU/destination. Based on the possible trajectory, the model estimates vehicles' future location and determines the possibility of the presence of the vehicle within a Fog nearby. From this estimation the model can create a possible available resource pool for applications to serve in the future. The Fog selection policy follows a simple analytic hierarchy process (AHP) calculation to make an initial decision. We then design an adaptation method for adjusting AHP to better accommodate the changes in the highly dynamic vehicular environment. The model constantly updates itself based on the decision made on the previous iterations, therefore reducing latency when an application request arises. In this work, we focus on designing and developing a resource allocation model in VFC paradigm to improve the application QoS and reduce DoS.

1.3 Contribution

The contributions of this work are split in two major parts: a light-weight cloudlet dwell time estimation model and a RL-based dynamic resource allocation model/system.

1.3.1 Dwell time and Resource Availability Estimation model

The cloudlet dwell time estimation model helps to estimate the number of vehicles that might be connected to the Fog and that can provide sufficient resources to sustain

vehicular applications. The model relies on following parts:

- Mobility-based trajectory estimation using simple and light weight linear regression model.
- Application of multiple linear regression (MLR) for estimating vehicles location with time.
- Based on vehicles predicted location and resource availability, prediction of possible available resources that a Fog might hold to sustain a vehicular application.

Predicting resource availability in advance helps the fog computing node to allocate resources instantaneously to application requests, thus improving the QoS and reducing DoS.

The results of the performance of the CDT and resource availability estimation model have been presented on the 25th International Symposium on Distributed Simulation and Real Time Applications (IEEE/ACM DS-RT 2021) [12].

1.3.2 RL based dynamic resource allocation model

We extensively test and measure the error rates of the CDT estimation model and found that the model tends to overestimate for resource availability. Hence, to mitigate the effects of overestimation rate of the CDT estimation model, we employ a Q-learning based dynamic resource allocation policy. Each resource allocation decision taken by the model is reviewed on each episode and appropriate feedback is given by a RL agent, ultimately converging to an optimal resource allocation policy. The Q-learning based method relies on following parts:

- Analytic hierarchy process (AHP) for finding a possible set to Fogs to allocate resources from.
- Estimated resource availability data for choosing optimal Fogs.
- Realtime accurate data on application status to evaluate the outcome of the decisions taken and receive appropriate feedback to update the R and Q values.

1.4 Outline

The remainder of this work is organized as follows. Chapter 2 presents the background of this work. Various approaches and methods applied to works related to this research, e.g. resource allocation and CDT estimation, are reviewed and discussed in

Chapter 3. Chapter 4 describes the problems that persist and can be solved in the field of vehicular environment. We also formulate the problems we deal with in our model in this chapter. A novel cloudlet dwell time estimation model is introduced in Chapter 5, and we elaborat on the methodology, mathematics and algorithms used for the CDT estimation. In chapter 6 we discuss the fog selection method and the implementation of reinforcement learning for resource allocation. We present the performance analysis of the CDT estimation model and RL based resource allocation model in chapter 7. Finally, Chapter 8 concludes the thesis work and provides future research directions.

Chapter 2

Background

Although VCC has introduced novel strategies and methods in organizing vehicles resources in highly dynamic environments, many shortcomings of this paradigm have already been identified. Many works have investigated the feasibility of various subsets of technologies leveraging VCC and/or connected vehicles, supporting and improving the management of the infrastructure. It is important to study and investigate the core architecture of such technologies like VEC, VFC and cloudlets. As a result, it is worth observing the main aspects and technological advancements in support of the architecture, their similarities and their advantages, and applications in VCC, VEC, VFC and MVC.

2.1 Vehicular Cloud Computing (VCC)

The vehicular Cloud computing can be defined as follows [24]:

> A group of largely autonomous vehicles whose corporate computing, sensing, communication and physical resources can be coordinated and dynamically allocated to authorized users.

Vehicular networks are a prominent feature of a smart city environment, as most of today's vehicles are equipped with on board computation units, most of which remain unexploited throughout the day. This makes them the perfect resources to form a cloud computing network acting as nodes leveraging the onboard resources. In the Cloud computing paradigm, local resources can be offloaded to a shared pool, which can become and ideal solution for problems that are compute and memory intensive.

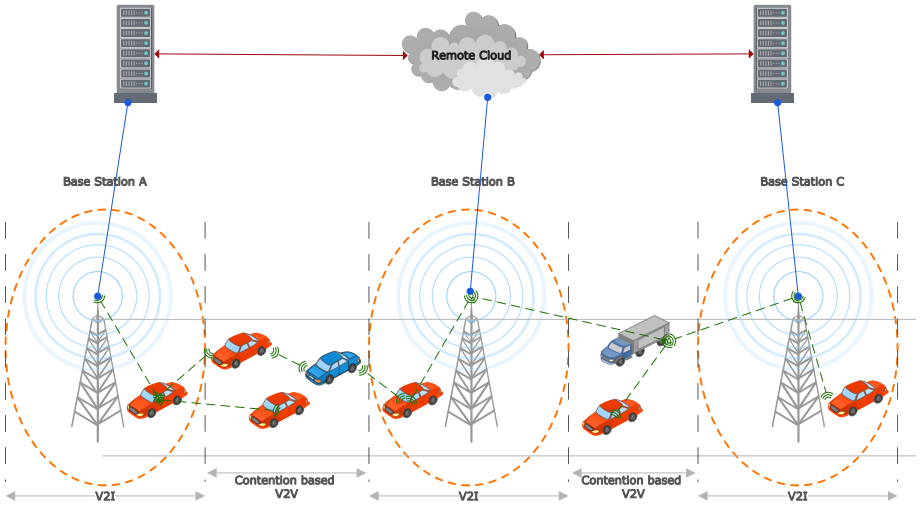

Figure 2.1: Vehicular Cloud Computing Hypothetical Scenario

2.2 VCC Architecture

VCC architecture generally comprises of three layers: inside vehicle, communication infrastructure, and back-end Cloud. The inside-vehicle layer consists of various mechanical and environmental sensors, navigational instruments, cameras, smart applications, wireless communication apparatus and more. The data generated and collected from the inside-vehicle layer then can be transmitted using the communication infrastructure to the back-end cloud system for storage and performing computation heavy tasks within the cloud. The communication layer is responsible for exchanging the operational data among vehicles, infrastructures and the cloud over advanced wireless networks, e.g. 4G, satellite or internet. Figure 2.1 shows a high level architecture of VCC where vehicles are connected with each other through V2V communication. By using road side infrastructures through V2I communication, vehicles are connected to the cloud server.

One advantage of vehicular Cloud computing is that the data aggregated over various scenarios within the network can be stored in the cloud. Then it can be used to perform various studies, analysis and decision making processes, as the cloud can use massive data sets to perform complex computation in minimal time.

There are various components in the cloud primary application services layer, which are called vehicular cloud computing services, and are described as follows:

- Network as a service (NaaS) [23]
- Storage as a service (STaaS) [3]

- Cooperation as a service (CaaS) [23]
- Computing as a service [10]
- Pictures on a wheel as a service
- Information as a service (INaaS) and entertainment as a service (ENaaS)

Several VC formation scenarios are:

- **Stationary VC formation:**
  A VC formed by stationary parked vehicles e.g. in a parking lot. Mostly acts as a conventional cloud in static environment.
- **Linked with a fixed infrastructure:**
  A VC formed in an area instrumented and deployed by some form of a static infrastructure.
- **Dynamic formation:**
  Using a unique vehicle as a broker tasked with authorization to accumulate computing resources to form a large computing entity similar to a supercomputer.

2.3 Vehicular Cloud Computing Applications

Various driving applications and traffic management services have emerged due to recent technological advances, e.g. flow control of traffic, monitoring and hazard detection, and infotainment services. All these applications can greatly benefit by utilizing the vehicular cloud computing infrastructures, which can cooperatively propagate and process information through connected vehicles and utilize highly capable and scalable cloud servers for resource intensive computations.

2.3.1 Traffic signal and route optimization

Traffic signals use pre-defined cycle lengths for red and green phases. Although the signal cycle can be changed depending on the time of the day or the week, it is not always practical, as the behaviour and congestion of traffic on the road cannot always be predicted. Using the aggregated data from a vehicular network regarding the volume of traffic in a direction, traffic signals can become more efficient by changing

the signal cycle dynamic [15]. Also, by using the congestion data over the cloud, any on-board routing and navigation application can make decisions to a better route for the vehicle to reach the destination.

2.3.2 Shopping mall data center

A recent study performed on teens shopping at malls showed that 95 percent of shoppers spent more than 1 hour at the mall, while 68 percent of them spent more than 2 hours [1]. Thus, thousands of cars and their on board computing resources remain idle for hours, which can be utilized for computing resources through the internet [3]. A business model can be setup by the mall management to use these resources in return of discounts, free parking for example. However, it has to be highly dynamic, as the arrival and departure of cars happen quickly and the time to use their resource is limited.

2.3.3 Road safety messages

Now-a-days most cars are embedded with sensing devices for proficient and safe operation. The on-board high definition cameras tracks the lane marks and help the vehicles to stay in the lane. Thus, cars have a sensor node, and a vehicular cloud can form dynamically with a large wireless sensor network. Other vehicles in close proximity can obtain data from sensors of another car and obtain valuable insights about road conditions and hazards that lay ahead. They can then act accordingly, which would otherwise be impossible [24].

2.3.4 Evacuation management

Transportation agencies need to strategize potential evacuation scenarios in urban areas. They can achieve this by intensive traffic modeling in metropolitan areas. A VC can help the authorities for evacuation by providing data in regards to travel time, bottlenecks for example. Exploiting intelligent transportation system, vehicular network, and mobile and cloud computing technology, an intelligent disaster management system can be built.

2.4 Vehicular Edge Computing (VEC)

A prominent portion of future intelligent transportation systems will comprise of vehicular networks. Already in 2021, various mobility aware services are in use in some forms. Cloud computing provides centralized computation and storage services using either a cloud server or a number of remote servers, and vehicular ad-hoc networks (VANETs) depend predominantly on cloud computing. Cloud computing provides the users with virtual servers, storage and computing capacity without much user intervention. Using cloud computing, the stored data can be accessed from anywhere from any vehicle, without large storage and computational power. With rising demand of computationally intensive and latency prone applications, the delay in transferring data from vehicles to cloud with ever increasing numbers of vehicles, and hight mobility is a challenge in VCC paradigm. Today's real world applications require low latency and uninterrupted service. Whereas, the communication and computation overhead is increasing day by day and consumption of energy on wireless devices are also increasing. This in turn increases the bandwidth cost massively. To assure the quality of service (QoS) on vehicular applications, vehicular edge computing (VEC) comes into play [11].

In the edge computing (EC) paradigm, data processing, computation and analysis is taken place at the edge of the network, in close proximity to the end devices acting as an agent in between the cloud and end users. With mobility, latency and communication overhead being the main issues of VCC and VANETs, edge computing can be a game changing solution in vehicular environments [11]. While traditional cloud computing is a centralized system, vehicular edge computing deploys its application in a distributed environment. Edge nodes having high computational power, and storage are deployed in close proximity to the vehicular networks. Hence, better QoS can be offered to the user. Wireless sensors on board the vehicles can directly communicate with the edge servers to collect, process and store data seamlessly, ensuring low latency at the application level and better context awareness. As opposed to VCC where most of the application level decision making might take place remotely, in VEC it can take place locally. Safety applications can benefit from this low latency. Although the development cost of VEC is much lower than VCC systems, the storage capacity and computational capabilities are expected to be lower than that of VCC. With high level of mobility support, device heterogeneity, flexible location and low latency, a few drawbacks can be overlooked [26]. Figure 2.2 shows a high level architecture and key components of VEC leveraging roadside cloudlets and AP edge

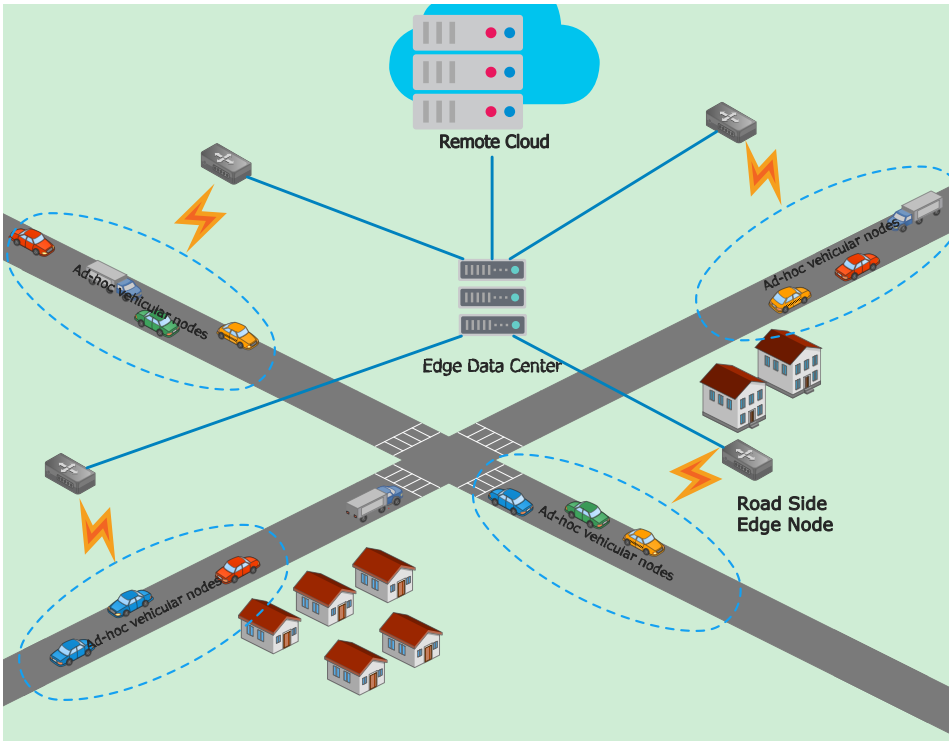

Figure 2.2: A hypothetical Vehicular Edge Computing environment

computing nodes at the edge of the network.

2.5 Vehicular Fog Computing (VFC)

The term Fog Computing was first coined by Bonomi *et al.* [8] from Cisco referring the extension of Cloud Computing to the edge of a network. A Fog exists in-between the cloud data center and IoT devices. More and more vehicles are now having on-board computational units. By utilizing these vehicles as the infrastructures for communication and computation, Vehicular Fog Computing (VFC) shows great promise. A VFC architecture is formed [13] by utilizing a collaborative multitude of vehicles' on-board units connected wirelessly or near-user edge devices to carry out communication and computation. The cumulative computation and storage resources of individual vehicles helps enhance the quality of services and applications. Figure 2.3 shows a high level architecture of VFC where vehicles are grouped according to proximity to form cloudlets which serve as Fogs, and AP edge computing nodes that deal with the connectivity with the cloud.

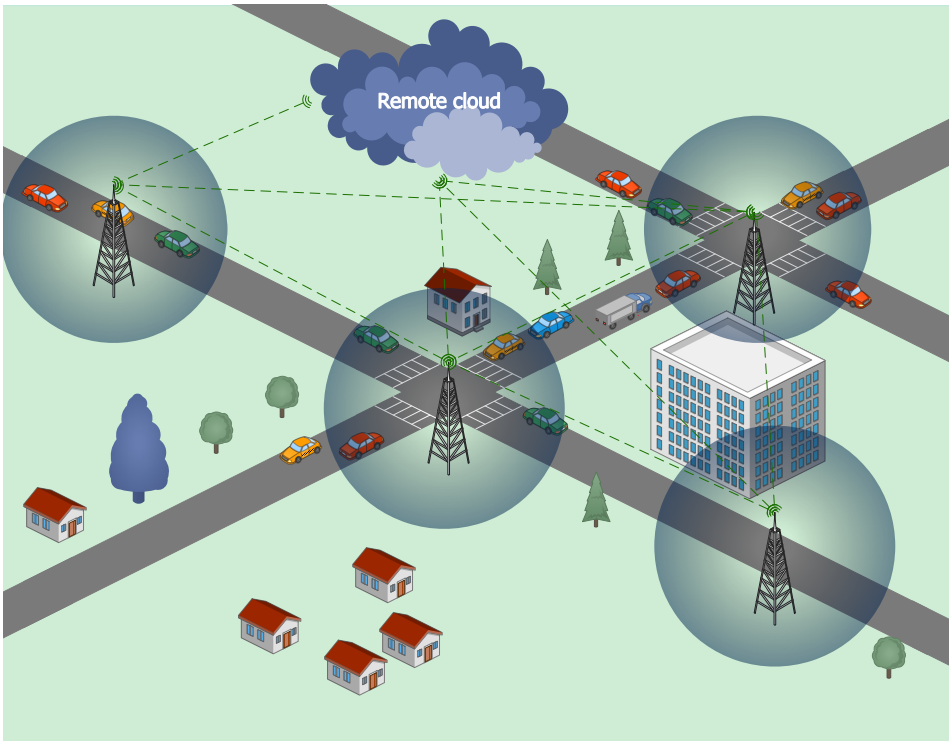

Figure 2.3: Vehicular Fog Computing.

2.6 Mobile Vehicular Cloudlet (MVC)

The term cloudlet was first proposed by Satyanarayanan *et. al* [30]. A mobile cloudlet is a peer-to-peer communication model [16] termed as a group of mobile devices in close proximity connected by short range communications. The particular model of cloudlet can be applied to a vehicular environment, where a collection of smart vehicles can connect to form a mobile vehicular cloudlet (MVC) to provide cloud computing service locally. Each smart vehicle in a MVC [37], termed as a node, can be a computing service provider and also act as a client. As computation service is provided locally, the communication latency and the cost can be greatly reduced, and the computation can be sped up, conserving energy by dividing the tasks among MVC Nodes. Figure 2.4 shows a high level architecture and key components of MVC, where vehicles are grouped as individual cloudlets within the range of RSUs, and RSUs are linked to the remote cloud through wired links.

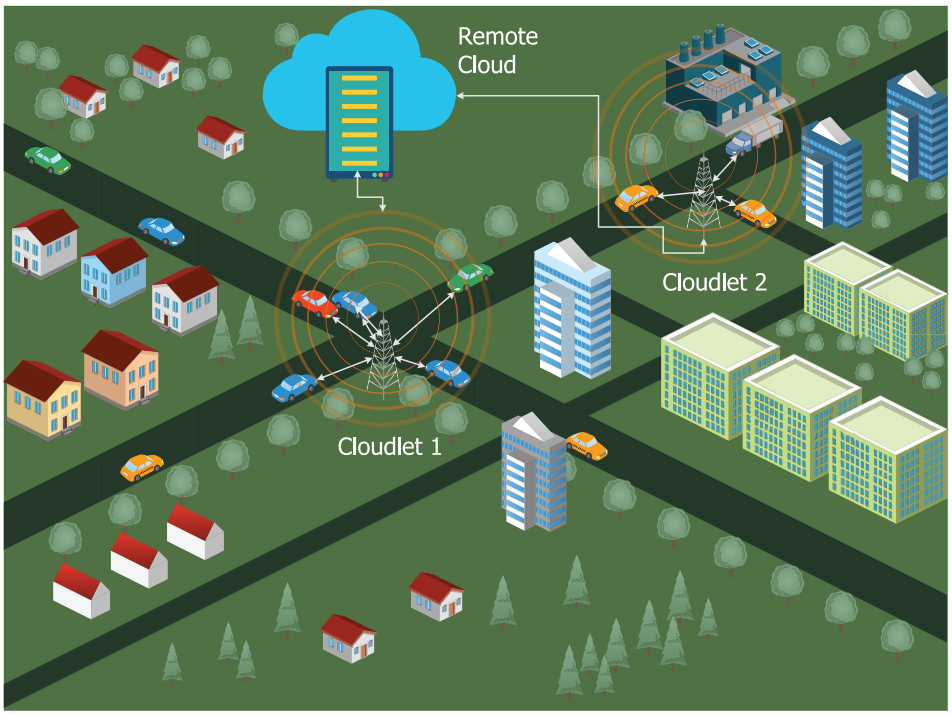

Figure 2.4: Mobile vehicular cloudlet

2.7 Advantages of Fog/Edge Computing

2.7.1 Reduced response time

In Fog/edge computing, the cloudlets or the fog nodes are closer to the IoT devices than the centralized Cloud is. In the traditional Cloud computing architecture, data originated to and from either of end user device or the cloud server have to go through multiple wireless hops to reach one another, which causes latency and execution delay. Whereas, usually in Fog computing, there exists only one wireless hop to reach fog nodes. It can be processed with very low latency and produce fast response time [29].

2.7.2 Scalability

In Fog/edge Computing, not every segment of data needs to be sent to the centralized cloud, as many of the tasks are accomplished by individual cloudlets in different regions. As in Fog computing setup, the centralized cloud receives less data and requires less computation capacity for each request, it can cater to a large number of users without requiring a large amount of bandwidths. If more users are being connected and require services, the system can create additional cloudlets instead of pressuring the central cloud server. It can reduce the amount of cloudlets in an

opposite scenario, giving the system more scalability [29] than a traditional cloud setup.

2.7.3 Security and Privacy

The cloudlet based Fog computing solution provides accurate control to end users, to control the access to their sensitive data [2]. As the data is not stored in a single centralized cloud, the security and privacy of user data is much more reliable in Fog systems.

2.7.4 Controlling Cloud Outages

In a traditional centralized cloud system, any catastrophic failure on the central server immediately requires the system to seize service and stop all operations. If the user data is stored in decentralized cloudlets, failure in one cloudlet or network outage does not cause a denial of service for end users, as they can be redirected to another nearby cloudlet to receive service.

2.8 Similarities between VEC, VFC and MVC

- Although vehicular Cloud vomputing is a paradigm shifting idea, researchers quickly found the drawbacks of centralized cloud systems in vehicular environments. All of these models (VEC, VFC and MVC) were proposed to remedy the shortcomings of VCC.
- All of these models perform computing tasks at close proximity to the end user devices as opposed to VCC.

2.9 Differences between VEC, VFC and MVC

- Although all of these models have close similarities, they were proposed to solve different sorts of problems and have different methodologies.
- MVC is formed incorporating vehicle on-board devices to act as micro data centers, whereas VFC extends cloud computing and emphasizes on device proximity. VEC emphasizes processing the computation at the proximity of the source of data.

2.10 Resource Management and Allocation

Due to the rapid increase in vehicles and number of applications running on a vehicular network, the information processing requirements e.g. entertainment, traffic and driving safety information is increasing, which requires large computing and storage resources. Due to the lack of ability to process large scale data on vehicle on board computational unit, vehicles need to transmit most of the information to the cloud for processing. Hence, improving the processing efficiency of vehicular cloud systems by effectively utilizing the cloud computing resources has become an interesting research topic in the field of vehicular networks. Recent years saw a large increase in studies and experiments for optimal resource allocation schemes for vehicular cloud computing. For example Lin *et. al.* [18] proposed an SMDP model for resource allocation in the VCC framework that takes into account heterogeneous vehicles and incorporates V2V and V2I.

In their work [21] Meneguette *et. al.* proposed a Cooperative and Adaptive REsource Scheduling Scheme (CARESS) for managing and scheduling resources without the necessity for roadside infrastructure. The technique takes into consideration the high dynamic topology and high mobility pattern of the vehicular cloud and uses a caching mechanism for resource registration to achieve high QoS. Meneguette *et.al.* [22] proposed a scheme for resource search and management in vehicular cloud connected network, named as SMART. The work leverages peer to peer protocol based on Gnutella [35] concepts to create cooperation among vehicles and establish connections to provide resource parameters in the vehicular network.

2.11 Reinforcement Learning

Reinforcement learning is described as follows:

> Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment in order to maximize the notion of cumulative reward. Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning. Reinforcement learning differs from supervised learning in not needing labelled input/output pairs be presented, and in not needing sub-optimal actions to be explicitly corrected. Instead the focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge). [14]

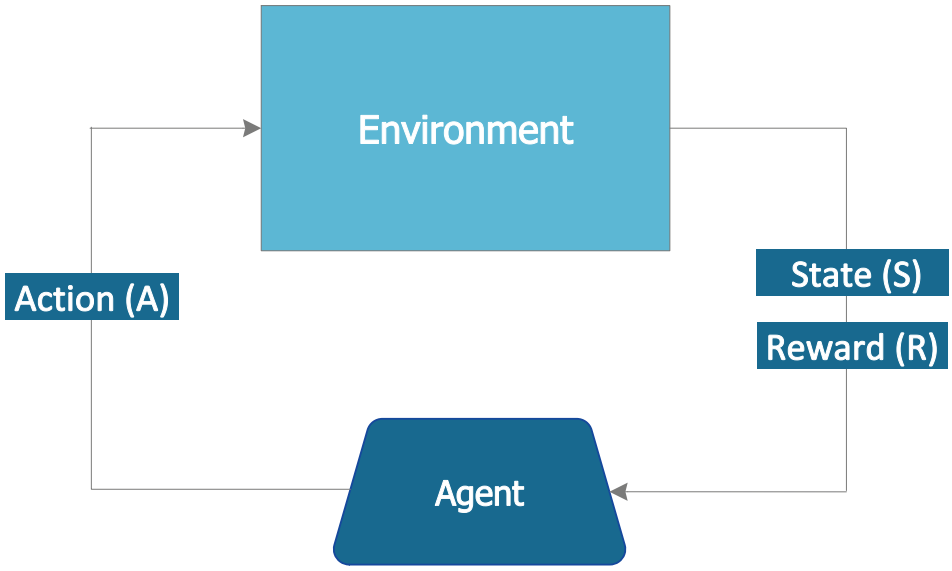

Figure 2.5: Reinforcement Learning.

In the standard reinforcement learning model [34], an agent is connected to its environment through perception and its corresponding action. On each step of interaction, the agent receives an input from the environment as a form of indication of the state S of the environment, therefore deciding on an action a to get as output. The action changes the state of the environment depending on the action. The environment then produces a reward signal r, which the agent receives as an input on the next iteration along with the new state. The overall goal of the system is to maximize the total sum of rewards after all the iterations. Guided by various reinforcement algorithms, the system can learn to produce maximum rewards over time.

There have been many research works utilizing machine learning with the vehicular networks, and recently reinforcement learning had some breakthroughs in vehicular environments regarding computation offloading and resource allocation problems and is promising more good results. Distributed machine learning has also been used in solving many complex problems.

Chapter 3

Related Works

Reinforcement learning can serve as a vital tool to improve service offloading and resource allocation solutions in mobile vehicular environments. Prominent recent ideas have been proposed to deal with resource allocation and service/computation offloading using reinforcement learning techniques. These works have a focus on improving the application feasibility and quality of service in Vehicular Edge, Cloudlets and Fog vehicular computing paradigms. Table 3.1 summarizes the approaches related to this work.

3.1 Cloudlet Dwell Time Estimation

Cloudlet dwell time (CDT) is defined as the longest period of time a certain amount of vehicles remains connected to a computation node e.g. a Fog/RSU while sharing their resources to accumulate a resource pool to sustain a vehicular application.

Peng *et. al.* [33] have proposed a cloudlet dwell time estimation method by mathematically calculating the vehicular cloudlet existence probability according to different traffic situations, using historical traffic data to assist vehicular edge tasks. Relying on these derived results, based on the specified requirements on computational capacity, the RSU can determine the feasibility of a given VEC-related application within the given road segment in advance.This allows the system to reduce latency for implementing the VEC application caused by the reactive-based feasible determination procedure. The inter-arrival of external vehicles to the test road follows the Poisson distribution with average arrival rate λ. It is assumed that all the on road vehicles have identical communication range R_c. By using Burke's theorem [9], they have calculated the average number of on-road vehicles and the average time consumed by each vehicle travelling through the road segment. Using the average of number

of vehicles and speed information, they have calculated the external vehicle arrival rate and vehicle density within the road segment. As the road segment consists of multiple fictitious queues, if there exists an empty slot of road where no vehicle is connected to another, it divides the vehicles into two isolated groups. Mathematically they have presented the formula to calculate the probability of the existence of a cloudlet containing NV_c vehicles on a given road, and demonstrated the performance of the proposed work with a simulation.

3.2 Computation offloading

Computation offloading is the task of sending computation intensive application components for remote execution on a remote server using wireless a network. The first step of computation offloading is application partitioning, which involves granulating each application into offloadable and non offloadable components. The preparation step takes care of the processes required for offloaded components to be used in the mobile application, including remote server selection, data and code transfer. The final step in the offloading process is the offloading decision, which is to decide whether to offload and execute a component remotely or not. Works described in this section tackled the computation offloading problem in the vehicular environment using multiple techniques, e.g. empirical, SMDP, Q-learning etc.

Ashok *et. al.* proposed dynamic computation offloading [4] by selectively offloading parts of the applications instead of the whole application. A dynamic computation offloading system should ensure that the response time of embedded vehicular applications are within their deadlines, where, response time is the total execution time of the application from initiation to completion, and deadline is defined as the maximum time within which the execution must be completed. The proposed framework is based on a service-based offloading technique. Each application is divided into various modules based on computation requirements. Each module is a specific set of function, the model definition of which is already available in the cloud. The system has to identify which modules should be offloaded and which should be executed on the on-board CPU. Once identified, the data corresponding to the module is then sent to the cloud. The cloud already having the model definition can then complete the computation and return back the output. Software controllers are used both in client and cloud machine to handle the offloading process. The client includes an application profiler that records input, output data size of modules, and execution time. A network profiler, interfaced with the TCP socket module, keeps track of the

network bandwidth represented as the round trip time measurement. To do this, the network profiler sends probes of packets to the server and measures the average RTT. It is done on a parallel thread so that the actual offloading is not hampered. The client-server communication is done using a wireless network. The model definitions of the offloadable modules are dynamically shared with the cloud server at the beginning, and updated whenever necessary at run time.

Liu *et. al.* [19] proposed a vehicle-assisted offloading mechanism for user devices by considering the task execution delay and limited computation capabilities. They have formulated the problem as a SMDP process and used Q-learning-based reinforcement learning methods to determine the optimal offloading and resource allocation policies. The states of the reinforcement learning algorithm is defined by the communication and computation model between the user equipment (UE) and the Vehicular Edge Server (VES) and fixed edge server (FES). For FES computation, the UE offloads its computation tasks via small-cell BS to its associated FES. For VES computation, UEs can offload the computation task to the VES through wireless connection between users and vehicles using one-hop ad-hoc network. For VES and FES offloading, the system measures the spectrum efficiency for the communication link between VES/FES and UE. For VES communication, there is interference from the vehicles reusing the same channel with the transmitting vehicle. As the UEs are considered to be using an orthogonal spectrum in each small cell, for FES communication there are no interference caused among UEs in one small cell. Rather, the interference is from the neighboring small cells. The action space of the model is to decide the computation task offloading strategy for selecting VES, FES or local computing methods. For both VES and FES computation, the time costs for executing the task is the sum of communication and computation time. Whereas, if local computation is chosen, there are no communication time involved, however the computation is much higher. They have formulated the transition of the number of available VESs for one UE as a Markov chain. The gain of the reward function is determined by the action of the system. In the proposed network, vehicles can provide computation services to user equipment as well as the traditional edge server.

In many cases, traditional computation offloading methods fail to incorporate the high mobility of dynamic environments and variable network conditions. Any offloading policies pre-set by any models may not perform well in a variety of environments. Reinforcement learning based models leveraging techniques like MDP, SMDP or Q-learning tends perform better in comparison with traditional works.

3.3 Resource Allocation

This section provides a high level overview on several works related to resource allocation and management in vehicular environment. We discuss various approaches to solve the resource allocation problem using MDP, deep reinforcement learning, analytic hierarchy process etc in highly mobile and dynamic scenarios.

Salahuddin *et. al.* [28] investigated resource management in vehicular cloud, and demonstrated the benefits of reinforcement-learning-based techniques for resource provisioning by integrating various proposed vehicular cloud models. The resource allocation problem in this paper is designed as a Markov decision process (MDP), where all possible configurations of the allocated resources are stored as a set of states and the transition from one state to another is defined as actions. They have dealt the management of virtual resources (communication, computation, and storage resources) as a software as a service model. The ultimate goal of the work is to tackle the resource allocation problem in VC by reinforcement learning, by dynamically provisioning resources to maximize long term reward and avoid near sighted decision making. MDP is a discrete time stochastic process denoted by a quad-tuple $< S, A, P, R >$ with S representing the set of states and A representing the set of acts. The transition from state m to state n is based on the action $a \in A$, defined by the probability P(m, n,a), with corresponding reward R(m, n, a). Q-learning, policy iteration, value iteration, and linear programming are some of the methods that can be used to solve the MDP. The authors used the policy iteration method to achieve the optimal policy. The MDP converges to the optimum policy when the overall incentive cannot be improved any further on the following iterations. MDP always chooses the best settings to optimise long-term reward by allocating resources in such a way that long-term allocations are reduced.

Ye *et. al.* [40] proposed a deep reinforcement learning based decentralized multi-agent resource allocation scheme for vehicle-to-vehicle communication. For latency prone applications, every link from a V2V environment is considered as an agent that learns to make own optimal decisions minimizing the interference by refining its resource sharing strategy through interacting with the unknown vehicular environment. The effects of other agents' behaviour on the environment is observable throughout the system and a co-ordination system is implemented by encouraging agents to share their activities with their neighbours and to make decisions in turns. By selecting the actions jointly, rather than independently, the agents in turn bring improvements to the system. The main issue of using DRL in vehicular environment

is the computational complexity. Although in the proposed system the time required for each iteration is very minimal, it was achieved using GPU processing. In larger urban environments, the time delay for DRL system may increase even further. The model requires an initial training, and then it adapts in real time.

Rickson *et. al.* [25] proposed a resource allocation and management policy for vehicular cloud, formulated based on mathematical methods in which it allowed for optimal decision making to based on various influence factors derived using analytic hierarchy process (AHP) to maximize resource utilization in the cloud. They have used processing power, storage, service time and bandwidth as influence factors by using AHP and mathematically derived decisions of which Fogs to choose to perform a specific service request or whether to drop or block the service. The AHP provides a structured technique for the decision making of problems with multiple criteria involved, by using pairwise comparison between the numerical values of each parameter and their relative degrees of importance, in order to adjust their weights at runtime. When a vehicle connects to the Fog, it adds its resources to the pool and when it leaves, the resource amount must be deducted from the pool. Moreover, the system has to reallocate any resources to other vehicles that was allocated to the leaving vehicle. When vehicles connect to the fog, it can also do service requests. The fog in turn have to allocate resources to serve that request, either by providing resources from its own pool, or allocate resources from another selected fog. If the selected fog does not have enough resources, the algorithm keeps the application on hold and pick another Fog. If the requested service cannot be served by any of the fogs in the system, it denies the service. The number of service requested and number of service denied is one of the important performance metrics used by the authors.

3.4 Remarks

Various methods and techniques, for example Markov decision process (MDP), semi-Markov decision process (SMDP), deep reinforcement learning (DRL), and Q-Learning, have proven to be useful in vehicular scenarios in recent years. While many of the works we have reviewed produce good results in ideal scenarios, many other fail to incorporate variable change in vehicular environment. Due to this fact, under-utilization of resources is a common problem. Although, resource allocation models for vehicular environment show promising results, after reviewing several approaches, it is evident that there still persists many drawbacks and problems that need to be addressed. This includes issues e.g. adaptability to intermittent network conditions and high

mobility of vehicular nodes, support of application heterogeneity, and reduction of under-utilization of vehicular resources. Improving these issues helps unlock the full potential of vehicular clouds.

Table 3.1 summarizes a few works closely related to the scope of this thesis.

Table 3.1: Summary of Related Works

Work	**Environment**	**Context**	**Approach**
[4]	VCC	Computation Offloading	Empirical
[28]	VCC	Resource provisioning	MDP
[17]	VCC	Resource allocation	SMDP
[40]	V2V	Resource allocation	DRL
[19]	VEC	Resource allocation and offloading	Q-learning
[33]	VEC, MVC	Dwell time estimation	Probabilistic Model
[25]	VCC, VFC	Resource allocation	AHP

Chapter 4

Problem Formulation

Most of the works that we have reviewed so far assume that there already exists a Fog or a cloudlet, using which, service offloading and resource allocation decisions can be taken. In almost all cases, the location and capacity of the Fogs or cloudlets are known. Peng *et. al.* [33] formulated a mathematical way to calculate the probability of forming a Cloudlet within a fixed segment of road, which in turn helps the RSU to determine the feasibility of a VEC related application. They did not address what the system does if the framework decides that its not feasible to run a VEC application. Also, they calculated the probability on a historic data which might not always provide the best results. In this work, they have used only one road segment and their framework finds out if a VEC application is feasible on that segment or not. However, in real life, it is possible that their pre-defined segment of road does not meet the threshold to form a cloudlet, but there is a probability of a cloudlet formation nearby, which this system fails to incorporate.

Rickson *et. al.* [25] used various influence factors and assigned weights to them to formulate a method for finding optimal policies for resource allocation on Fog based vehicular environment. The influence factors and their weights are pre defined and fixed and may not perform very well in heterogeneous application environments.

As depicted in Figure 4.1, we assume that an urban centre is composed of a set of moving vehicles. Each vehicle contains a set of resources that can be shared/exchanged among other vehicles or the Cloud. Our goal is to provide flexible and adaptable resource management solutions in a vehicular Fog environment.

For a vehicular network system, determining the feasibility of a related application within a given road segment in advance can reduce latency significantly. By knowing the probability of a cloudlet formation within a region by CDT estimation in advance, the system can initiate the application procedure faster than a reactive-based

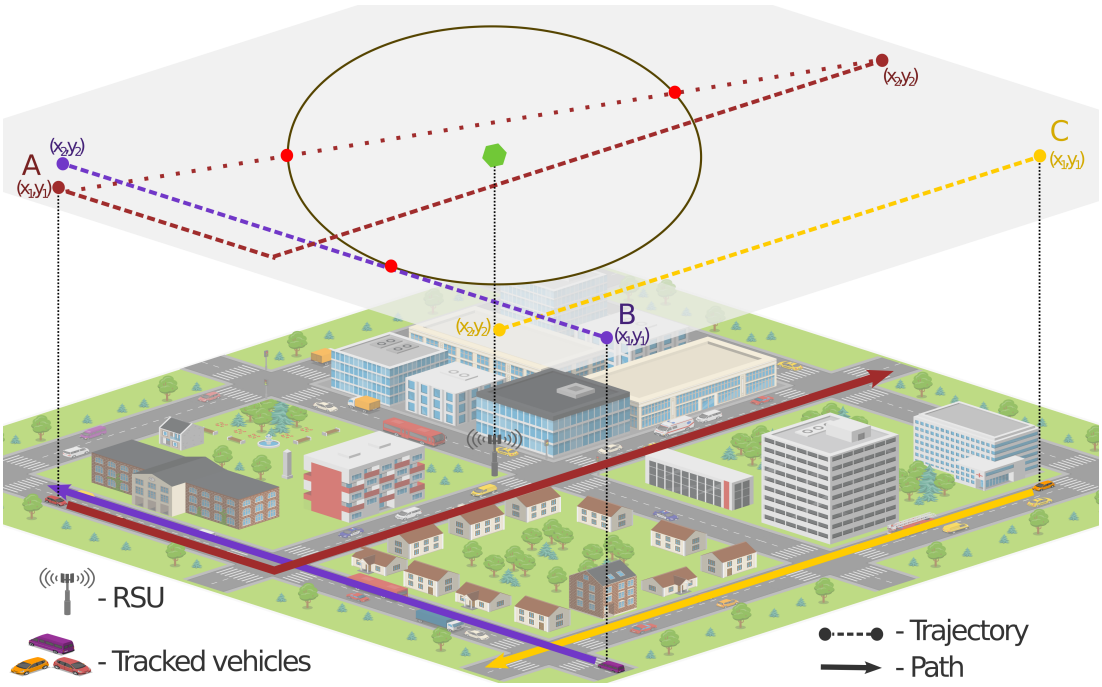

Figure 4.1: Path trajectories crossing RSU range in urban centre.

feasibility estimation system. Existing CDT estimation systems can be extended to take into account multiple segments of road at the same time reacting to updated traffic flow data in real time. Coupled with an optimal resource allocation policy system which evolves its policies automatically based on available computing and communication resources, the system can outperform any other resource allocation framework currently in existence. Hence, we propose a mobility-based dwell time estimation method for accurately estimating vehicular resources in a Fog, leveraging which we can design an adaptive and highly dynamic resource allocation model using reinforcement learning techniques.

4.1 Dwell Time Estimation

The first part of the proposed system is to be a dynamic Cloudlet dwell time estimation method. In this part, we first define the regions and divide them into multiple segments. A dynamic approach is applied for identifying the cloudlet existence probability for each of these segments by feeding the system with real-time traffic data. As flowing traffic moves from one segment to another, the probability of cloudlet formation for each segment is expected to be changing rapidly. The goal of this segment of the system is to keep the estimations for each segment updated at all time so that the

VEC system can take the values and do proper calculations for the following steps.

4.2 Resource allocation policy

As we have multiple segments and the vehicular applications have multiple options to receive resources, the network and computational parameters for each of those segments also differ. To this end, we want to implement a resource allocation model based on influence factors. Based on the calculations and CDT estimations from the first segment, the system decides how to allocate resources efficiently. For making decisions, the system needs to know the resource details from each of the cloudlets and the requested service overheads e.g., bandwidth, storage, computation power, information regarding the produced data. At the start of run time, each resource has an influence factor attached to them. We measure the performance of the influence factors' weights over time and make them reactive to the service requested. For example, for a critical security related application, the execution time delay should be of utmost importance, whereas for an entertainment application the storage resource could be vital. The goal of the system is to learn from the application request type and update itself according to necessity.

4.2.1 RL-based Cloudlet/Fog selection

As this system could be potentially elastic in nature and we want to ensure heterogeneity in terms of devices and applications, we don't want our system to be static. Rather, we want it to evolve over time. We employ reinforcement learning algorithms to learn the outcomes of the decisions made in first two segments of the system and try to improve the result in the next iteration. The RL agent has knowledge about which Fog was chosen with what probability, and which influence factors had what weights. Based on the output reward function, it takes proper actions to ultimately maximize the overall system reward. As the influence factors' weights may largely vary depending on the type of application service requested, the RL agent also knows the characteristics of the application. In turn, the system learns to correlate between applications requested, and choices made, and the reward of those choices, which helps make optimal educated decisions on heterogeneous applications.

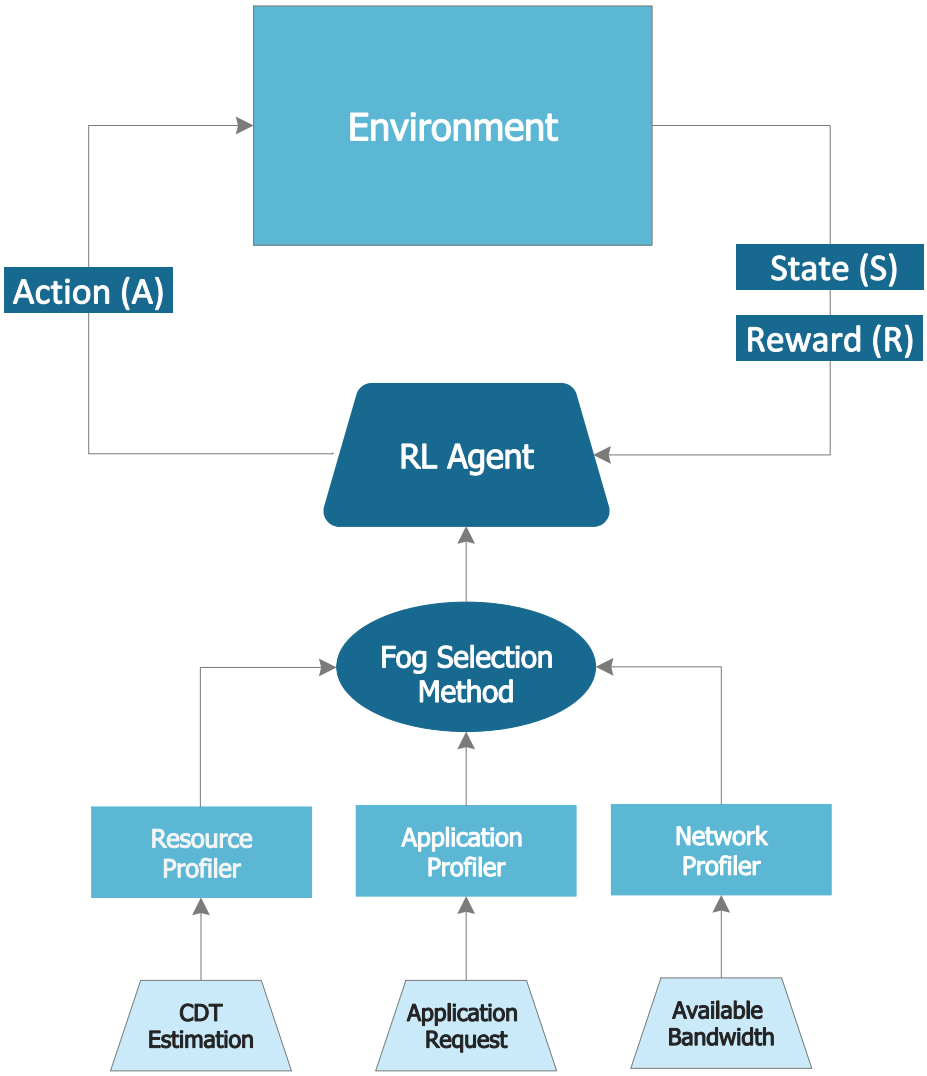

Figure 4.2: Reinforcement Learning based resource allocation model

4.2.2 Parameters

- **CDT estimation:** We consider the CDT estimation as a parameter of the RL algorithm as it is one of the vital influence factor in our system. Machine learning algorithms often act like black boxes and we cannot always predict the outcome, i.e the future reward for a decision. Any linear decision algorithm would choose the Fog with highest amount of available resources to allocate resources from. However, it might be the case that choosing the Fog that barely meets the resource requirement could result in maximizing overall system reward. Analysing these outcomes, we are be able to find correlations between CDT estimation and optimal resource allocation decision, which in turn could contribute to come up with a better CDT estimation method that incorporates

other fewer factors into account.

- **Computation power:** Computation power required for an application and computation power available from a Fog is important to take into account to make a decision about resource allocation. If a fog does not have enough computation resources to spare, it is an easy choice not to choose that fog.
- **Storage Space:** Each application request might come with some data that needs to be stored while processing the request. The application profiler needs to know all necessary information regarding the application and communicate to the fog selection method with the requirements.
- **Bandwidth:** A network profiler keeps track of available bandwidth, to and from the application source Fog and the selected Fog to allocate resource from. Even if the selected fog has abundant resources, it is vital for the network profiler to decide whether the application can be executed and receive the returned result in time, keeping the bandwidth constraint in consideration.

Chapter 5

Cloudlet Dwell Time Estimation Model

Cloudlet dwell time (CDT) is defined as the longest time that the number of interconnected vehicles in a cloudlet remains greater than the VEC-task required value while preserving the connection with an arbitrary RSU [33]. In our CDT estimation model, we assume that the position of arbitrary RSUs can be anywhere in the given scenario and the number of available RSUs and their corresponding co-ordinates are unknown to all vehicles at the beginning. In the beginning, the RSUs start broadcasting their position and the information is disseminated throughout the urban region; thus, all the vehicles become aware of all the RSUs' locations. Moreover, the movement direction of the vehicles in the scenario is arbitrary, and vehicles move independently, not sharing a single direction. We devise a vehicle trajectory prediction method that takes place as a distributed computing process to generate the required parameters for the CDT estimation.

In Section 5.1 we discuss a cloudlet dwell time estimation method leveraging Linear Regression for traffic flow characterization. We also discuss the protocols used for V2V and V2I communication in an urban scenario.

5.1 Traffic Flow Characterization

We assume that the communication range of RSUs and vehicles are r_r and r_v respectively, and the current timestamp is t_i. Although the RSUs are already deployed in the urban environment and are static, we do not assume that all the vehicles in the environment are aware of their coordinates. The RSUs transmit their coordinates at predefined intervals, and their coordinates are disseminated through the vehicles

using multiple hops. The intersection of each vehicle's trajectory and the range of the RSUs determines if a vehicle will be in range of that RSU. Each vehicle in the scenario acts as a computing node and keeps track of the vehicle's past coordinates, speed, and timestamp independently. Vehicles also keep track of all the RSUs' locations that they receive. As the dissemination of location information happens with only 2 hop neighbors, we assume that the information the vehicles receive is from a nearby RSU.

We incorporate a linear regression model for each vehicle based on its past position coordinates - x and y to predict the direction towards which the vehicle is moving. This estimated direction allows us to project it against the urban road segment topology. Linear regression suits this direction estimation because it is a simple, lightweight method to model the relationship between two variables. After training the regression model with historic data points from the vehicle's movement, we identify a linear equation of the regression line and two points on the line. The trained model is periodically updated for each vehicle after a predefined random amount of time. The update period can be simply set to 20-30 seconds. However, in real scenario, vehicles can remain idle for 30 seconds or travel a prominent distance by that time. Therefore, we resorted to a random selection method of the simulation time where the vehicles' timestamp is divisible by a number between 13 and 19, generated randomly. Hence, the location reading intervals are completely random and represent the real world movement pattern more accurately. With each training instance, it is expected to project the vehicles' trajectory more accurately as the data points used while training increase with each iteration. The equation has the form $Y = a + bX$, where Y is the dependent variable, X is the independent variable, b is the slope of the line, and a is the y-intercept. We find the values of a and b from n data points using Equations 5.1 and 5.2.

$$a = \frac{(\sum y)(\sum x^2) - (\sum x)(\sum xy)}{n(\sum x^2) - (\sum x)^2} \tag{5.1}$$

$$b = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2} \tag{5.2}$$

The vehicle's trajectory is a straight line given by the regression model. Using the oldest and latest known x-coordinates for the vehicle, we identify two points (x_1, y_1) and (x_2, y_2) on the regression line. The Y values are not the actual recorded y-coordinates. Instead, the pair is on the regression line which more precisely mimics the trajectory of the vehicle. Following the assumed r_r communication range of

ALGORITHM 1: Estimate of RSU contact

```
Data: R; pos_i
Result: rsu_c; rsu_n
rsu_c = ∅; rsu_n = ∅;
while t_int AND (pos_i − pos_{i−1} > Δs) do
    for r ∈ R do
        if d_r < r_r then
            rsu_c = r;
            Calculate p_{x,i};
            Calculate Δt_{d,i};
            wsm = (t, Δt_{d,i});
        else
            if Vehicle trajectory crosses any RSU range then
                rsu_n = r;
                Calculate p_{e,i}, p_{x,i};
                Calculate Δt_{r,i}, Δt_{d,i};
                wsm = (t, Δt_{r,i}, Δt_{d,i})
```

an RSU, we identify the possible case scenarios where the estimated trajectory of the vehicle might intersect the RSU's range, allowing a possible communication to happen. We thus simply determine the possible intersections of the circle and a straight line containing points (x_1, y_1) and (x_2, y_2). We have three possible scenarios to consider in regards to the vehicular path and the RSU range circle depicted in Fig 4.1:

- The estimated trajectory of vehicle misses the RSU range.
- The estimated path of vehicle is a tangent to RSU range.
- The estimated path intersects the RSU range circle in two points, where the closer intersection is the entry point and the further intersection is the exit point.

There are also possibilities that the path of a moving vehicle crosses multiple RSU ranges, but we consider the closest RSU for our calculation as the traffic direction is determined as a probability and the vehicle can change its direction at any time before reaching RSUs further away. The projected path may also denote that a certain vehicle is already in the range of an RSU and moving out towards another or none. Algorithm 1 summarizes the estimation each vehicle conducts to determine if its trajectory goes through the range of any known RSU.

5.1.1 Entry and exit point calculation

The trajectory of the vehicle is represented as an (infinite) line determined by two points (x_1, y_1) and (x_2, y_2). This trajectory might pass across the communication range r_r of an RSU in two points or meet the range at only one point. The intersection thus can be simply a tangent to the range or a secant to the range, or may not intersect at all. Mathematically, we calculate two co-ordinates at which the trajectory intersects the range circle. We assume the nearest and farthest point as the entry point $p_{e,i}$ and exit point $p_{x,i}$, respectively. We can then calculate the linear distance between $p_{e,i}$ and $p_{x,i}$ and get the probable dwell distance $d_{d,i}$ of the vehicle under the range of the next RSU. The linear distance between current location and $p_{e,i}$ gives distance to entry point $d_{e,i}$.

The vehicle uses a multiple linear regression (MLR) model to predict the time required for it to reach the RSU range and corresponding dwell time within its range. For this, we use at least 5 data points from the vehicle's time series location and speed data. However, simply dividing the linear distance by the average speed of the vehicle does not provide an accurate estimation of the time required to travel that distance in urban topology. For two data points $p_{n,i}$ and $p_{n+1,i}$, we calculate the average speed $S_{a,i} = (s_{n,i} + s_{n+1,i})/2$ and linear distance $d_{n,n+1}$, which we use as the explanatory variables in the MLR model. We use the time difference between two points as the response variable. From a total of n data points, we get $n-1$ training data. After the training is done, we plug the MLR model with the average speed from n data points and $d_{e,i}$, which returns the time required to reach the entry point $\Delta t_{r,i}$. Also, by receiving the average speed and $d_{d,i}$, the model returns the dwell time for the vehicle $\Delta t_{d,i}$. The calculated data is then sent to the corresponding RSU using a WAVE short message (WSM). We are using WSM because of the underlying communication system being based on WAVE protocols. The vehicles store the RSU information once, but they transmit vehicle information periodically through WSMs, and the messages are disseminated using other vehicles to reach the range of the RSU. We use WSM for data dissemination as these messages are targeted to an RSU, having their destination address indicating an RSU. Each RSU has an unique ID in the environment which the vehicle use to target the WSM towards that particular RSU. We can use h amount of hops for the WSM to reach the RSU. We use a pure flooding dissemination protocol, in which every node of the network relays the information once they receive it from its one-hop neighbors. To keep the flooding in the system to a minimum, vehicles drop any messages that they already disseminated in the past. To accomplish this, we keep track of vehicle ids which already disseminated this message in the message header.

ALGORITHM 2: Classification of vehicles in RSU

```
Data: R; v_i; wsm;
if wsm[target] = r then
    if d_(v_i,r_c) < r_r then
        v_i → v_o;
        t_x = t + Δt_d;
    else
        if (d_(v_i,r_c) > r_r) AND (d_(v_i,r_c) < r_r*3) then
            v_i → v_f;
            t_e = t + Δt_r;
            t_x = t_e + Δt_d;
        end
    end
    v_i → db_i;
end
```

If the distance between a vehicle and the destination is greater than the distance between the source and destination, we also discard the message for dissemination.

Here, we adopt to 4 hops for limiting the amount of flooding in the environment and mitigate a broadcast storm problem. Please note that we adopt an extremely simple, but effective, dissemination protocol since dissemination is not the scope of the work, but just a tool.

5.2 Traffic Analysis in the RSU

For our proposed approach to properly determine the load of arriving, residing, and departing vehicles, it requires RSUs to receive, filter, process, and store relevant information sent by nearby vehicles.

5.2.1 Filtered Dissemination

In our scenario where multiple RSUs are placed, the computation process is not centralized and can run in parallel. We consider a vehicle and an RSU can communicate bi-directionally only when the vehicle reaches the range of the RSU. Each vehicle in the scenario keeps a timestamp t to denote the state it is currently in. The message sent from vehicle i of its state $vs_{i_{t_j}}$ can hop multiple times through other vehicles to reach the RSU. For this reason, the particular RSU can receive the same message originated from the same vehicle, and multiple times through multiple vehicles. We discard the message of the same state for a vehicle in the RSU. If a new message

comes in for the same vehicle for a new state $vs_{i_{t_{j+1}}}$ we update the previous state information with the new one in the RSU database.

5.2.2 Collecting and Classifying Vehicles

With the information sent from each vehicle, the RSUs can calculate necessary values for CDT estimation. Algorithm 2 describes the classification process of vehicles in the RSU and necessary calculations to store vehicle data in the database. We discard the vehicle information that is too far away from the range of the RSU. The vehicles that are already inside the range of RSUs are classified as outgoing vehicles v_{i_o} and incoming vehicles as v_{i_f}. Along with all necessary information for a vehicle, the RSU receives the timestamp t when the message was sent by a vehicle, the amount of time to reach the entry point of the range of the RSU (Δt_r), the amount of time the vehicle can be in the range (dwell time) (Δt_d). Therefore, for an incoming vehicle v_{i_f}, we calculate the entry time $t_e = t + \Delta t_r$ and exit time $t_x = t_e + \Delta t_d$ from the range of the RSU. For an outgoing vehicle v_{i_o}, we calculate the exit time $t_x = t + \Delta t_d$.

Upon receiving the information from vehicles and calculating necessary values, the system stores them in a known vehicle database. For an incoming vehicle the system needs to know the time required to reach the RSU and the corresponding dwell time to calculate the entry and exit time. For a vehicle already inside the range of the RSU, we simply calculate the exit time by using the current timestamp and predicted dwell time.

5.2.3 Real Time Flow Estimation

Whenever a vehicle is in range of an RSU, it sends a message to the RSU with the relevant information, e.g. position and speed. These WSMs are sent every second and the RSUs receive the messages without the necessity of any dissemination as the vehicles are already in range. Upon receiving the message, the RSU stores the information in the database with the timestamp as key values. As a result, the database has all the vehicles' records for each particular timestamp. From this data, we can calculate the average traffic density ρ for a given period of time and the maximum traffic density ρ_{max}. The RSU also stores the vehicle's speed information, from which we can calculate the average vehicle speed v. In the database for each second, the RSU stores all the vehicle IDs that are in range. When we have sufficient data points built up in the database, we can compare the vehicle IDs of a certain timestamp with the vehicle record of previous timestamp. We find out which vehicle

ALGORITHM 3: Maximum CDT Calculation

Data: R; N_{vc}; t; $totalSimTime$; r_r; $\Delta t_d = 0$

Result: $\Delta T_{d_{max}}$;

```
while Δt_d <= (totalSimTime − t) do
    N_{t+Δt_d} = Σ_{i=1}^{n} V_{i,t+Δt_d(d_{v_i,r_c} > r_r)}
    if N_{t+Δt_d} >= N_vc then
        Δt_d + + ;
    else
        break;
    end
end
ΔT_{d_max} = Δt_d
```

IDs are new to the RSU and which vehicle IDs are missing from the previous data point. We denote the new vehicles as the newly arrived and the missing vehicles as the departed vehicles. From this approach, we then calculate the number of vehicles that arrived and departed between two particular points in time. After doing the calculation for a certain number of times, we can then find out the average external vehicle arrival rate (λ) and the average vehicles departure rate (μ) in terms of (veh/s).

5.3 Vehicular Cloudlet Dwell Time Estimation

Algorithm 3 shows the calculation process inside an RSU for maximum dwell time estimation. When the estimation method is triggered at t, it is given a value of N_{vc}, the minimum amount of vehicles that needs to be within the range of the RSU. We assume the maximum dwell time of N_{vc} vehicles cannot exceed the amount of time between current time and total simulation time. Starting from t and for each dwelling second Δt_d, we proceed to calculate the number of vehicles ($N_{t+\Delta t_d}$) that are within the range of the RSU. If $N_{t+\Delta t_d} \geq N_{vc}$ we decide that the cloudlet persists at $t + \Delta t_d$. We continue this process for each additional Δt_d until $N_{t+\Delta t_d} \leq N_{vc}$ and conclude as $\Delta T_{d_{max}} = \Delta t_d$.

We assume that an application request arrives in the system periodically in the future. We want to be able to calculate the feasibility of the application by determining the cloudlet dwell time and resource availability for that application through a certain amount of time. From the known vehicles database, we are able to calculate the number of vehicles within the cloudlet and the time the cloudlet formation persists. Application cases and requirements to consider are the following:

1. The application needs certain amount of resources for certain amount of time and needs continuous connection with the same vehicles (continuous connectivity).

2. The application needs certain amount of resources for certain amount of time but continuous connection with same vehicle is not needed (disrupted connectivity).

To reduce the latency in a VEC application, we want to proactively determine the feasibility of the application beforehand by predicting the possibility of a cloudlet formation with sufficient vehicles with available resources connected within the range of the RSU. In this work, we determine the possibility of a cloudlet formation by identifying when $n \geq N_{vc}$. Here, N_{vc} is the total number of vehicles required to be connected to the RSU with available resources necessary to complete an application request, and n is the total number of vehicles that are actually connected.

For an application starting at t_a with execution time Δt_a, we determine the cloudlet formation possibility for the two application cases mentioned above. We mathematically define the possible amount of vehicles $\phi_c(n)$ that will be within the range of the RSU with continuous connectivity from an application's start time and throughout its execution time from Equation 5.3. Equation 5.4 denotes the calculation of possible amount of vehicles $\phi_d(n)$ that can be present within the range throughout the application's execution time. In this case, we also consider disruptive connectivity, which means connection with the same vehicle is not necessary as there are be other vehicles that may come in.

$$\phi_c(n) = \sum_{i=1}^{n} V_i \left[(t_e \leq t_a) \;\&\&\; ((t_a + \Delta t_a) \leq t_x) \right] \tag{5.3}$$

$$\phi_d(n) = \sum_{i=1}^{n} V_i \left[(t_e \leq t_a) \; \right] + V_i \left[\; ((t_a + \Delta t_a) \leq t_x) \right] \tag{5.4}$$

If a RSU has the possibility of having n vehicles within its range and each vehicle has a probable dwell time of Δt_{d_i}, then we can calculate the average dwell time as:

$$\bar{D} = \frac{\sum_{i=1}^{n} \Delta t_{d_i}}{n} \tag{5.5}$$

We consider the RSUs in our model as individual Fogs which maintains their own resource pools. When the vehicles enter into the range of a RSU, it adds to the

cumulative resource pool of that Fog. When a vehicle looses its connection with the RSU, the RSU deducts the amount of resource from the pool. Subsequently a vehicle can do a service request for an on board application to the RSU. The RSU then allocates available resources for that application from the resource pool. The resource pool estimation is done in advance using the predicted number of vehicles and their respective dwell time. Therefore, latency can be decreased when an application request arises as there is no additional calculation involved to estimate the resource.

Chapter 6

RL based dynamic resource allocation

The overall goal of this work is to better the Quality of Service (Qos) for applications in a vehicular environment. To accomplish such a goal, our system works towards reducing the service delays and application denials by intelligently allocating resources within all the Fogs formed within the region. When a service request comes from a vehicle to the RSU, the RSU tries to allocate its own available resources to complete the process. If there are not enough resources to allocate, then it chooses some other Fogs based on metrics resource requirements and availability and the weights for each factor. In traditional approaches [25], these calculations and decisions takes place after when a service request has been made, with possible delays in decision making. Our work focuses to diminish this latency by completing the calculations and decision-making process before the service request comes, so that the fog can allocate the resources with minimal latency. Of course, as we are predicting the available resources with our MLR model much earlier, there is a margin of error. Therefore, by only reducing the latency, we cannot be definitive in assuming that our resource allocation model performs better than any established work. To this end, we incorporate a reinforcement learning model in making these decisions so that, over time, our model learns by itself and the performance becomes better with each iteration to gradually improve its allocation decisions. We use a simple AHP method to find possible solutions to the Fog selection problem but do not actually select the Fog with best priority score. Rather, we use an iterative Q-learning algorithm to evaluate those possible Fogs with higher AHP rankings to make decisions based on feedback from previous iterations. This enables us to make the resource allocation model even more accurate than the established works using only AHP.

The vehicles that are within the range of a Fog can generate random application requests periodically. Each request presents independent and particular resource requirements and deadlines. When an application request arrives within any of the Fogs, we put the application in a queue of apps waiting to be executed. Our model then determines the feasibility of the applications beforehand by predicting the amount of resources the fog may hold. We linearly select the most feasible applications by comparing the amount of resources the fog has and the required amount, and determining if the application execution can be completed with the available resources before the deadline.

6.1 Application Requests Overview

To accomplish seamless resource allocation and delivery, we couple the resource allocation model with multiple profilers and managers that are tasked with separate functions within the system. As the core goal of our resource allocation model is to improve the QoS, dividing the model functions into separate methods which can run independently can reduce latency to great extent. Therefore, these profilers and managers play a vital role in the overall performance of the resource allocation model.

6.1.1 Application Profiler

The proposed model is coupled with an application profiler built on-the-fly. The purpose of the application profiler is to keep track of all the application requests that are coming in to the RSU from the vehicles within its range. We keep track of the processing power and storage requirement that each application requires within the application profiler. For each of the RSUs, we deploy individual profilers that can run independently, resulting in prevention of overloading of the profilers. Application profilers constantly monitor requests. Therefore, whenever any vehicle generates an application request, the AP is the first to start processing that request.

Each application request has a header attached to it with the requirement information. The application profiler receives the computation and storage requirements from the app request and communicates to the fog selection method. The profiler also keeps building a database of the application requests and their status. In our model, we resorted to three types of initial statuses for each application: *initiated*, *denied*, and *assigned*. The assignment decision is made by the fog selection method discussed in Section 6.2.

After the deadline is over for the application, the profiler receives notification from the selected Fog, in which it reports the final execution status: success or failure. These two statuses are trivial in our model as we later use them to train our Q-learning algorithm.

6.1.2 Network Profiler

Each RSU is coupled with a network profiler (NP) that periodically monitors the network conditions among each RSUs. For our model, we resort to a star network topology among the fogs. The network profiler periodically updates the round-trip time (RTT) between each of the RSUs and feeds into the Fog selection method. To keep the data overhead to a minimum, the network profiler only records the most recent RTT value between two Fogs.

6.1.3 Resource Manager

Each RSU has a resource manager (RM) attached to it. The primary objective of the resource manager is to keep track of the available resource pool for each fog. As our model uses a look ahead resource estimation method, the resource manager needs to be able to store information in two stages. In part, the RM works closely with the CDT estimation model. The RSUs periodically receives WSMs from nearby vehicles regarding the possibility of being in range and the amount of resources the vehicle can contribute to the resource pool. Therefore, the RM can calculate an estimation of the resource available for a certain time in the future, which is then propagated to all the other RSUs nearby. When an application request comes to a RSU, it can then use the information from the resource manager to assign an application to a certain fog.

Apart from the probable resource estimation, the RM is also tasked with calculating a real-time resource pool. Whenever a vehicle enters into the range, it sends a WSM to the RSU with information regarding the available resources, which gets added to the resource pool for the fog, maintained by the RM. Accordingly, whenever the vehicle disconnects from the RSU, the RM deducts the available resource from the resource pool. Also, whenever an application is assigned to a RSU, the amount of resource needed to execute the application needs to be reserved and deducted from the resource pool. Therefore, the resource manager plays an important role in managing and maintaining information regarding all resources within a fog.

6.2 Fog Selection Method

6.2.1 Analytic Hierarchy Process (AHP)

The analytic hierarchy process (AHP) is a decision-making method that is used to analyze and make a wide array of complex decisions having multiple criteria. It was primarily developed by Thomas L. Saaty [27] considering many variables in the system and having prioritization of metrics over one another to make optimal decisions. While dealing with a problem, the AHP method generally solves it in three major segments. The first segment is to understand the problem that needs to be resolved. The second segment is to find the alternate solutions to the problem. The most important segment is the third one, which is the evaluation process of the alternate solutions that it undertakes based on various criteria. Therefore, for a problem that may have multiple potential solutions, the AHP process assigns weights on each of the criteria that it is assessing on. While it is easier to find a solution for problems with less criteria, AHP proves to be one of the best mathematical methods to solve problems with multiple solutions with various criteria.

6.2.2 Influence factors and calculations

In a vehicular resource management and allocation scenario, where there exists multiple probable sources of resources, the employed model has to choose the most optimal source with chances of providing the best possible result. In our proposed model we have application requests coming from vehicles to a RSU, and the RSU has to choose the best possible Fog from a pool of Fogs that it is connected to, to best serve the application. In a heterogeneous environment with intermittent network connection and variable resource requirement, the problem of choosing the best optimal Fog can be perceived as a multi-variable single goal problem. The goal is to choose the best fog where the application has the best chance of being served keeping the resource requirements, e.g. computing power, storage space, and bandwidth in consideration. Therefore, we incorporate an AHP based resource allocation policy [25] which allows for optimal decision making based on individual weights we assign for each criteria: computation power, storage space, service time and bandwidth. Table 6.1 summarizes the weights of the influence factors [25] used in this work. We then go ahead and transform the set of pairwise comparison values into a priority based ranking of the alternative choices. To achieve this, we first divide each element of the weight matrix by the sum of its own column to calculate the normalized relative weight and

Table 6.1: Weights of each influence factor.

Factor	Computation	Storage	Service time	Bandwidth
Computation	1	2	3	4
Storage	1/2	1	3	4
Service Time	1/3	1/3	1	4
Bandwidth	1/4	1/4	1/4	1

obtain the following matrix:

$$RW = \begin{bmatrix} 0.48 & 0.56 & 0.41 & 0.31 \\ 0.24 & 0.28 & 0.41 & 0.31 \\ 0.16 & 0.09 & 0.14 & 0.31 \\ 0.12 & 0.07 & 0.04 & 0.07 \end{bmatrix} \tag{6.1}$$

The sum of each column of RW is equals to 1. We can then obtain the normalized principal Eigen-vector also called as the priority vector by averaging each of the rows, resulting in the following matrix:

$$W = \begin{bmatrix} 0.44 \\ 0.31 \\ 0.17 \\ 0.08 \end{bmatrix} \tag{6.2}$$

Therefore, from the priority matrix we obtain the priority scores for each of the influence factors while selecting a Fog as follows:

- Computation power (W_{cp}): 0.44
- Storage space (W_{ss}): 0.31
- Service time (W_{st}): 0.17
- Bandwidth (W_{bw}): 0.08

When an application request arises through the application profiler, the Fog computing node then proceeds to calculate the weight based score for each of the available Fogs $f \in F$ in the system using the resource profiler. For a Fog f_i with available resource I_{cp}, I_{ss}, I_{st}, and I_{bw} we calculate the fog selection score as:

$$S_{f_i} = (W_{cp} * I_{cp}) + (W_{ss} * I_{ss}) + (W_{st} * I_{st}) + (W_{bw} * I_{bw}); \tag{6.3}$$

The conventional method of taking any decision using AHP is to choose the option that has the highest score achieved from Equation 6.3. While this method shows some good results in choosing optimal Fogs for better serving application requests over greedy approach, we have to set the influence factors on each of the variables beforehand. As all application requests are not identical and one application may prioritize one requirement over another, it is possible that the Fog selection method does not always provide optimal results. As the weights of the influence factors does not change over time, the Fog selection mechanism may be biased based on the weight distribution for each of the factors. Therefore, we do not simply use AHP as the main fog selection method in this work. We rather use AHP to obtain a set of probable Fogs that may be able to fulfill an application request.

To this end, we employ a Q-learning based reinforcement learning method on the Fog selection function to obtain more accurate results. Using an evolving Q-learning algorithm helps offset the bias that the Fog selection method may have depending on the weights of the influence factors. Even if a bias happens in the system and the model makes a few wrong decisions, the Q-learning agent evaluates the feedback on each iteration and tries to correct itself on the next decision making episodes. The Fog where the application request happens is treated as a state, and all the possible Fogs determined from the AHP process are treated as actions in the algorithm. The overall goal of this RL agent is to evaluate the decisions taken by the fog selection method and provide feedback to the model so that it can correct itself on the next decision making process, and therefore reach to a converging point where most of the decisions taken by the model can be considered as optimal. The overall optimization problem of optimal resource allocation is granulated into finite sub-problems. By making optimal choices on each of these sub-problems, the model ultimately contribute to the overall optimality of the solution to the problem.

6.3 Q-learning Method

Q-learning is perceived as an off-policy reinforcement learning algorithm which works to find the best possible action for a given current state. Q-learning was first introduced by Chris Watkins in 1989 [38] and a theorem was presented to prove the convergence of the algorithm by Watkins and Peter Dayan in 1992 [39]. It is termed

an off-policy algorithm mainly because of the fact that it does not have a policy to start with for choosing its actions. Rather, it learns the best policy to maximize the overall system reward. For a state $s \in S$ in the system, there can be a finite number of actions $a \in A$. On each iteration of the algorithm, for each state, the RL agent tests the outcome of choosing each of the actions available. For each decision made, there can be a reward feedback or a penalty feedback to the algorithm. Over the next iterations, the RL agent tends to choose the actions that have high reward values over the states that have low reward values. In an iterative manner, the actions that have high possibility of reaching the goal gain more reward points, and the actions that produces negative results have less reward points. Ultimately, the algorithm converges to a point where it can simply choose the actions that have the highest rewards values for each state and reach the system goal.

6.3.1 State and Action Space

The states of the algorithm in our model are defined as the Fogs that have formed within the region $S = (F_1, F_2, F_3, F_4, F_5, F_6)$. Each application request comes through the application profiler from each Fog. We denote the Fog where the application request happened as the current state (S_c). The action space of the Q-learning model is the set of Fogs available $A = (F_1, F_2, F_3, F_4, F_5, F_6)$. While all the Fogs in the system are considered as an action space, for each current state S_{c_i}, it is possible that all the actions are not available for the current state. In the proposed model, Fogs are chosen using the fog selection method, which can provide multiple possible Fogs. These probable Fogs act as the available actions for the model to choose from.

6.3.2 The Rewards

The most important part of the Q-learning algorithm is providing rewards for each action taken by the RL agent. For each state S_i there are six possible actions, and therefore six reward values. For six possible states S and six available actions A for each, we simply construct a 6x6 reward matrix R. To kick start the learning process, we provide an initial reward value of 100 for each state-action.

$$R = \begin{bmatrix} 100 & 100 & 100 & 100 & 100 & 100 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 100 & 100 & 100 & 100 & 100 & 100 \end{bmatrix} \tag{6.4}$$

In traditional Q-learning based applications, like a path finding robot, the R-matrix is pre-defined. In a highly dynamic mobile vehicular environment, with intermittent network condition and multitude of applications with variant demands, we simply cannot pre-define the reward values for choices the model make beforehand. Hence, we need to periodically update the R-matrix taking into account the previous action and the outcome of it. We provide the agent with a high positive reward (+1) for each of the successful application assignments. In addition, we penalize the agent with a negative reward (-1) for selecting a fog for application assignment where the application failed to execute. In this manner, the R-matrix of our proposed model also evolves with real-time feedback from the system. The periodical feedback based update of the R-values adds to the already adaptive Q-learning algorithm's performance to great extent and make the model more adaptive to variable changes in the vehicular scenario.

6.3.3 The Q-Values

The whole process of reinforcement learning with Q-learning algorithm revolves around finding the Q-values for each state-action pair. We use temporal differences (TD) for the Q-learning algorithm as the agent has no prior knowledge of the environment but learns and evolves on each episodes. For each state in the system and for each available actions of each state, the model stores and updates the q-value for the corresponding pair. Therefore, the information has a shape of [state, action]. While taking decisions on each episode, the agent of the RL model looks at the Q-values for current state and chooses the action based on the maximum value available. We build a 6x6 matrix called the Q-matrix to hold the Q-values. Initially we put zeros for all the state-action pair.

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.5}$$

After getting the feedback for each action taken by the agent, we then update the Q-matrix using values from R-matrix. We use Bellman's optimality equation for iteratively updating Q-values for each state-action pair until to the point the Q-function converges to an optimal policy. The process is termed as value-iteration and the Bellman equation is at the center of this process.

6.3.4 Bellman equation for State-value Function

Dynamic programming method is used to solve problems by solving sub-problems whose solutions are found and stored and then using these solutions to form solutions to the problems. In cases when the problems are optimization problems, the principle of optimality needs to hold in order for the method to find the optimal solution by solving and combining the solutions to the sub-problems. In 1954, Richard Bellman coined one of the key formulas that are used in reinforcement learning techniques today, stated as the Bellman Equation [6]. The purpose of the Bellman equation is to use the reward value for a state-action pair and update the corresponding Q-value in the Q-matrix. For a state s and action a, using the Bellman equation we can therefore calculate the q-value $Q(s, a)$ as:

$$Q(s, a) = (1 - \alpha) \times Q(s, a) + \alpha \times [r + \gamma \times \max_{a'} Q(s', a')] \tag{6.6}$$

α is the learning rate of the algorithm. It defines the importance factor that the old Q-value holds while calculating a new Q-value for a state-action pair and determines how fast we want the Q-learning agent to learn about the system and controls the rate of learning for the agent. r is the reward value obtained from the R-matrix. $\max_{a'} Q(s', a')$ denotes the maximum Q-value for a state s and all available actions derived from the Q-matrix. On each iterations of the algorithm, for each of the actions, the agent searches for the largest Q-value for all possible state-action pair. γ is used as a discount factor for reward calculation. For each episode of calculating the new q-value we choose the maximum q-value for that state and all available actions. However, during the learning process, the model cannot be conclusive that the action with maximum value has the best possible chance of providing optimal solution, and hence can be mistaken in choosing that action and calculating the q-value. Therefore, we can offset the error factor by using γ which helps in determining the importance the previous maximum value holds in calculating the new state-action value.

Algorithm 4 summarizes the whole process of the fog selection method using Q-learning. When an application request arrives at a Fog, we consider that Fog as a state of the Q-learning algorithm. We conduct the same process for all the states in the system. Initially the set of available actions (A_{av}) is empty (line 1). The resource manager continuously keeps track of the resource estimation for each of the Fog provided by the CDT estimation model. The core contribution of the CDT estimation model goes to the resource manager (RM). As the CDT estimation model predicts the number of vehicles and the time they are in range, the resource manager

can record the resource estimation from these factors. Whenever the Fog selection method calls for resource information, the RM therefore is ready to provide it without delay. Using the information from the RM, the algorithms progresses to find a set of available actions/Fogs (A_{av}) using AHP (line 3). The AHP-based Fog selection method leverages Equation 6.3 to find A_{av}, which represents a subset of all available actions/Fogs (A) with higher ranking scores. For the current state, the algorithm now can have multiple actions to choose from. We choose the Fog that has the maximum Q-value to allocate resources from. For the current state, the algorithm thus searches for the highest state-action value pair from the Q-matrix and select it as the chosen Fog (a_{sel}), from which resources can be allocated (lines 4 and 15). As the Q-learning model trains iteratively, we need to update the reward for the previous decision that has been made. Therefore, we evaluate the outcome/status of the decision taken by the algorithm for the previous application requests app_k from the information provided by the application profiler (line 6). We provide a positive reward of 1 if the application was a success (line 8) and -1 if failed (line 11) and update the R-matrix accordingly. We consider only the application that have completed their execution, successfully and unsuccessfully. After these applications are considered, they are removed from the set of application executions. Finally, for the current state-action pair we calculate the Q-value using Bellman equation, as described in Equation 6.6, and update the Q-matrix (line 13-14).

ALGORITHM 4: Q-learning algorithm for fog selection

Data: S; A; R; E_{app}; γ; α

$Av = \emptyset$;
for $a_i \in A$ **do**
    $A_{av} \cup AHP(S, a_j)$;
$a_{sel} = \max Q(S, A_{av})$;
for $s_i \in S$ **do**
    for app_k *in* $E_{app}(s_i, A)$ **do**
        if $app_k = success$ **then**
            $R[s_i][a_k] += 1$;
            $E_{app} - app_k$;
        else if $app_k = fail$ **then**
            $R[s_i][a_k] -= 1$;
            $E_{app} - app_k$;
    for a_j *in* A_{av} **do**
        $Q(s_i, a_j) = (1 - \alpha) \times Q(s_i, a_j) + \alpha \times [R[s_i][a_j] + \gamma \times \max Q(s_i, A_{av})]$;
return a_{sel};

We resort to an online and off-policy Q-learning algorithm because we want our model to be adaptive to unpredictable variable changes in the vehicular scenarios. Off-policy algorithms do not have a pre-defined policy to start with and develop an optimal policy over time, which is suitable to dynamic vehicular conditions. Potentially, a pre-trained offline model could perform better from the start than an online training approach. However, it leaves the model vulnerable to dynamic conditions where mobility and network conditions are unpredictable and cant change rapidly. Therefore, the proposed online training method caters best in highly changing environments adapting over time based on a feedback based mechanism.

Principle of Optimality

For a policy to be optimal, it is vital that it comprises of a property that no matter what the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision [5]. The proposed resource allocation model can be considered to be an optimization problem as the goal of the model is to optimize the resource allocation decisions and reach an optimal Fog selection policy. Therefore, the principle of optimality needs to hold

in order to provide the justification of using the Q-learning based algorithm. The principle of optimality holds if every optimal solution to a problem is composed of a sequence of sub-problem decisions where the final result is optimal. In case of our proposed model, the sub-problems are fog selection decisions on each iteration of the algorithm. The solution to each of this sub-problem relies on the feedback from previous state, and therefore are more efficient and returns higher reward than the previous decision. On each iteration of the algorithm, for each state, the agent chooses the action with highest Q-value among all possible actions. Each of the solution to the sub-problems can thus be perceived as a greedy approach by the Q-learning algorithm. However, according to the convergence theorem for Q-learning [39], the model converges to the optimum action-values with probability 1 as long as each state-action pair is be visited infinitely often and sampled discretely. For this theorem to hold, the probability of each action to be selected by the policy must have a non-zero value. As the model follows an off-policy algorithm and does not have a Fog selection policy to start with, the probability of being chosen to allocate resources from is never zero in any iteration of the algorithm. Also, on each of the iterations, each of the possible actions are sampled to determine a selection decision, and hence follows the Q-learning convergence theorem. Therefore, while the overall outcome of the model is a optimal solution, the solutions to the consisting sub-problems are also optimal. Therefore, we conclude that the principle of optimality holds in our proposed model.

Chapter 7

Performance Analysis

7.1 Simulation Environment

For the implementation and simulation of this project we have used VEINS along with SUMO and OMNet++. VEINS is a Open Source vehicular network simulation framework. It ships as a simulation suite for vehicular networking [32]. VEINS models are supported and executed by OMNet++. For this experiment we have used Veins-5.0. VEINS serves as the basis of simulation. Simulation of Urban Mobility or SUMO [20] is an open source and highly portable microscopic road traffic simulation package designed to handle large road networks. For this simulation we are using SUMO version: 1.2.0. OMNet++ models are based on modules which communicate by exchanging messages [36]. It is mainly used for network simulation. The active components of the models are programmed using C++. It also requires the Microsoft .Net framework. Version 5.0 has been used in this simulation. It communicates with SUMO using TCP sockets. The whole ITS scenario stands on WAVE short message (WSM) as the supporting communication protocol. Network nodes (e.g. Vehicles, RSUs) communicate through both V2V and V2I.

7.2 Traffic Network Topology

We use the map of Cologne metropolitan area as our urban center, projecting a recorded data set of real world traffic movement onto it, as depicted in Figure 7.1. The region comprises of dense urban area with standard grid layout as well as highways allowing variable mobility pattern across the whole region. We placed 6 RSUs in the scenario in such a way that the range of each does not overlap with each other. Each

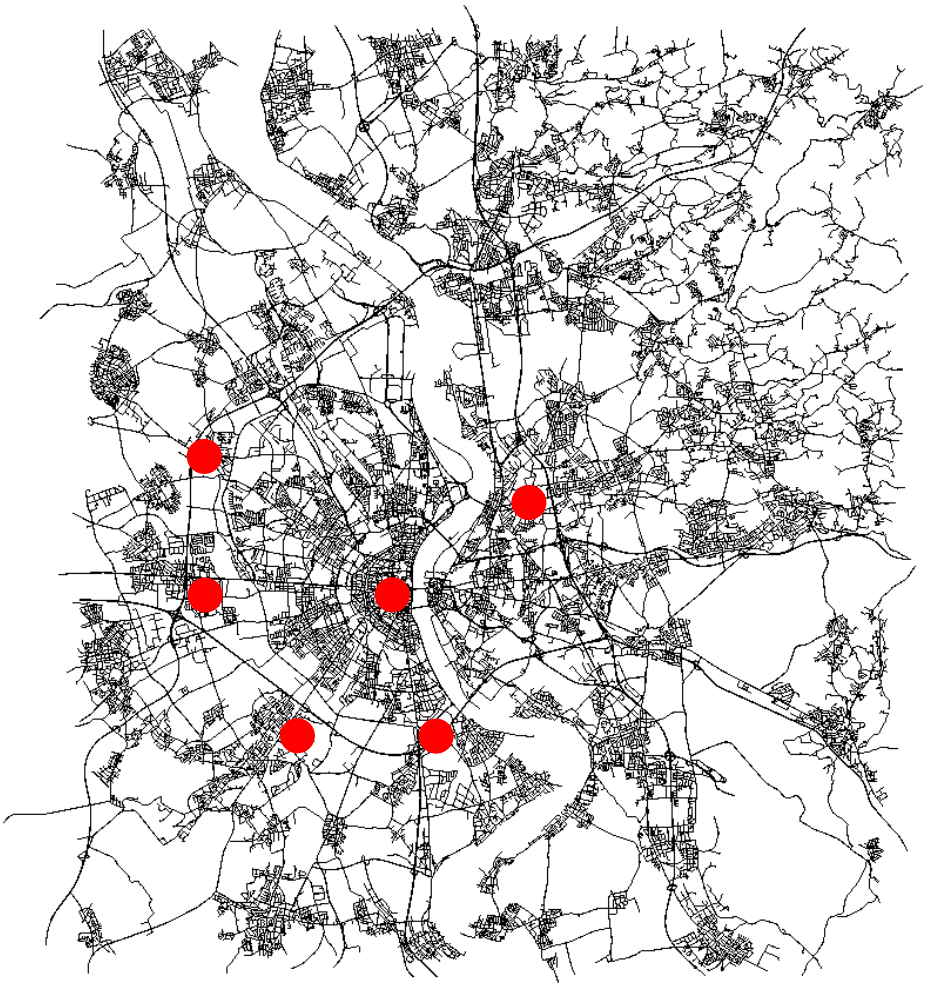

Figure 7.1: Cologne metropolitan area used in the simulation analysis.

vehicle starts their journey at a predefined time, and moves through the urban center until its journey ends. Vehicles communicate with the nearby RSUs from time to time and send information regarding their current and future positions. We compare the estimated values with actual positions after the simulation ends.

7.3 Performance analysis of CDT estimation

We have presented the theoretical mathematical modeling process of the proposed vehicular CDT estimation method. As a result, we have conducted performance analyses to evaluate the contribution of our proposed approach.

Table 7.1: Simulation parameter settings

Parameter	Value
Urban Area	$35000 \times 35000m^2$
Number of vehicles	1000 - 3000
RSU Density	6
PHY Model	IEEE 802.11p
Vehicle comm. range	400m
RSU comm. range	400m
Transmission power	120mW
Noise floor	-98dBm

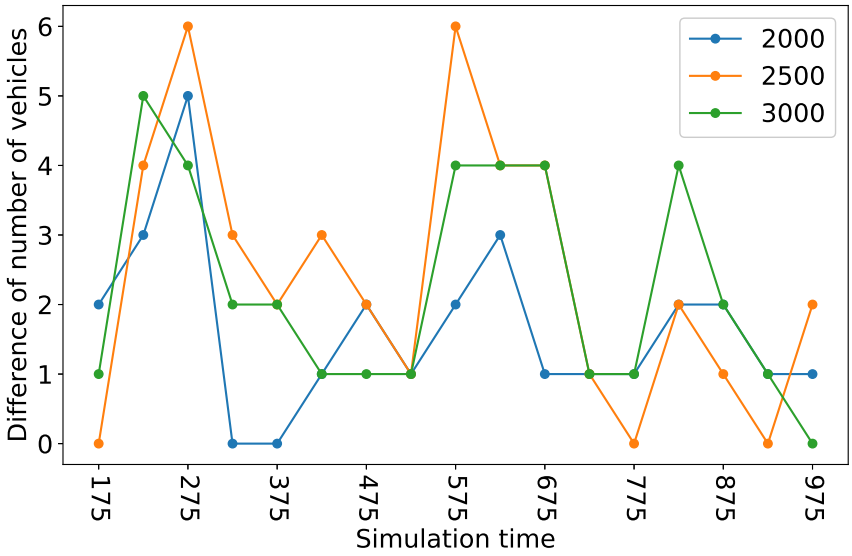

Figure 7.2: Difference of predicted and actual results with continuous connection for the number of vehicles (Snapshots of RSU 24).

7.3.1 Scenario and Methodology

The application request comes to a RSU a certain amount of time before the actual application launch. For simplicity of calculation, we always generate an application request 45s in the future with an execution time of 5 seconds. We then calculate and predict the number of vehicles that will be in range of the RSU throughout the entire execution time and record it into a database. We log the vehicles positions from when they enter into a RSU's range until they go out of the range. For this, the vehicles that are in the range, send WSM messages to the RSU. Upon receiving the WSM the RSU logs the vehicles' location.

Once the whole simulation scenario is complete we have the predictions and the

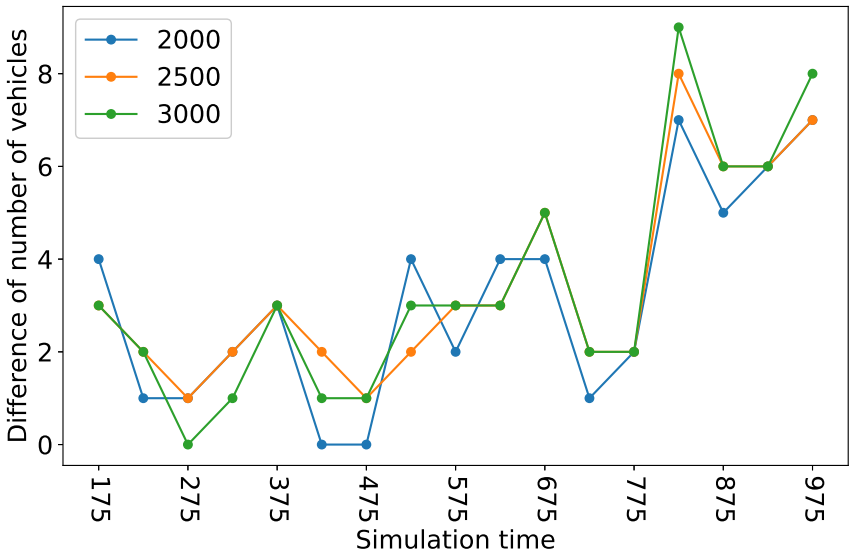

Figure 7.3: Difference of predicted and actual results with disrupted connections for the number of vehicles (Snapshots of RSU 24).

actual positions of vehicles with timestamps. We can then analyze the accuracy of the prediction model with the real data. We have adapted this performance analysis methodology because comparing with the actual results gives us the best possible accuracy in error rate measurement. Also, as the vehicles can directly communicate with the RSU without any necessity of data dissemination, the logging method can easily be extended to create a real time logging system which can in turn help correct the prediction model in real time.

7.3.2 Parameter Settings

For simplicity of calculation, we assume the communication range for the RSU and all vehicles is 400m. We have used 1000, 1500, 2000, 2500, and 3000 cars in simulations of total 1000 seconds. Table 7.1 summarizes the parameter settings used in the scenario.

7.3.3 Performance Metrics

We compared the predicted values with the actual results as one of the key performance metrics for our model. Moreover, we want to achieve efficient performance keeping the amount of messages in the system as low as possible. Therefore, the control overhead is another major performance metric for our analysis.

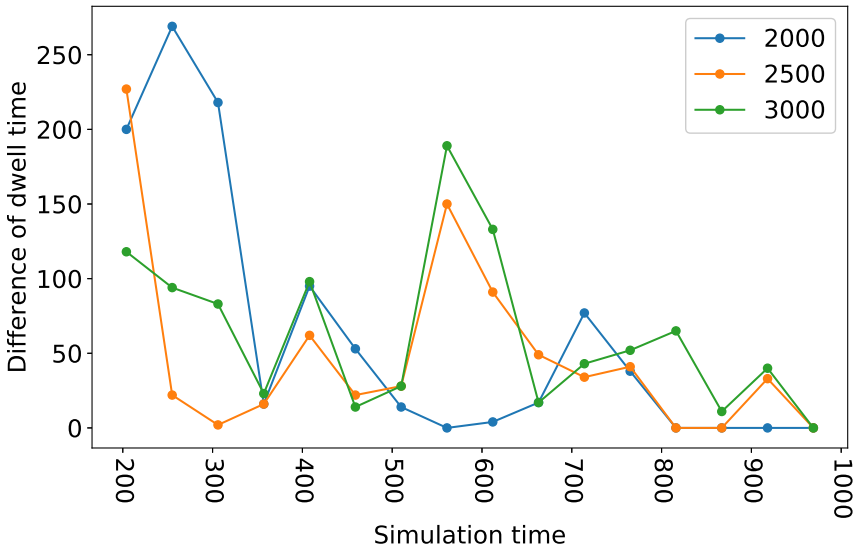

Figure 7.4: Difference of predicted and actual results for dwell time (Snapshots of RSU 24).

Estimation/Prediction Accuracy

- **Number of vehicles within a certain period of time**. Our model estimates the amount of vehicles that will be present within the Fog for certain period of time. From there, we can also calculate the amount of available resources. The metric is split into two views:
  - **Continuous connectivity**. It represents the number of same vehicles traversing the region.
  - **Disrupted connectivity**. It represents the total number of vehicles within the region.
- **Corresponding dwell time**. The model also predicts the continuous amount of time where there are at least N_{vc} vehicles within the Fog.

We measure the accuracy of our model by comparing the estimates against the actual simulation readings. The accuracy of these predictions are highly significant in this work as we use this model to determine the possible resource availability for each Fog and the feasibility of an application maintaining high quality of service.

Control Overhead

We measure the number of control messages sent through WSM from vehicles and RSUs necessary to disperse data throughout the scenario for the system to work

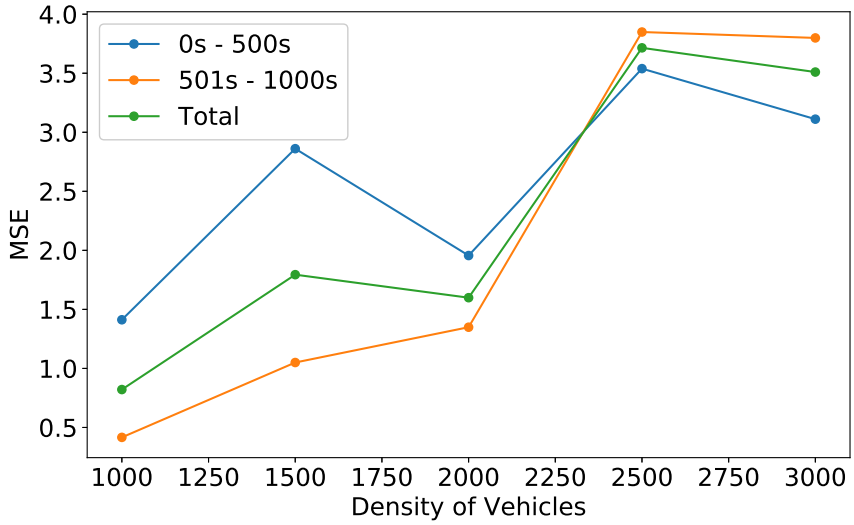

Figure 7.5: MSE analysis in estimations in two periods of simulation ($0s - 500s$ and $500 - 1000s$) for # of vehicles with continuous connectivity

efficiently. We measure how the number of WSMs vary depending on the number of vehicles and total communication overhead. We also measure the number of WSMs that fails to reach their destination and the effects of total number of messages on message delivery failures.

7.3.4 Results

Number of vehicles prediction

Figure 7.2 and Figure 7.3 shows the visual comparison between prediction and actual results. To be compatible with the later part of our work, we generate random application requests on each of the Fogs every 50 seconds. For simplicity of calculation,we assume the deadline for each service request is 5 seconds. We then calculate the possible amount of vehicles that might be in range of the RSU and therefore able to share their resources during the application execution time. The x-axis of Figure 7.2 and Figure 7.3 depicts the application time and the y-axis shows the difference between the estimate and the actual values in terms of number of cars. For various amounts of vehicles we saw that the RSU shows a similar pattern in the prediction deviation. For continuous connectivity, RSU 24 tends to converge to zero at the end of the experiment. Among the 6 RSUs placed within the scenario, we pick this particular RSU as it is close to the urban center and have larger vehicles cluster around it. As our model is driven by training the vehicle trajectory to estimate the results, we attained best results for the RSUs that has more vehicles within its range or nearby. We also

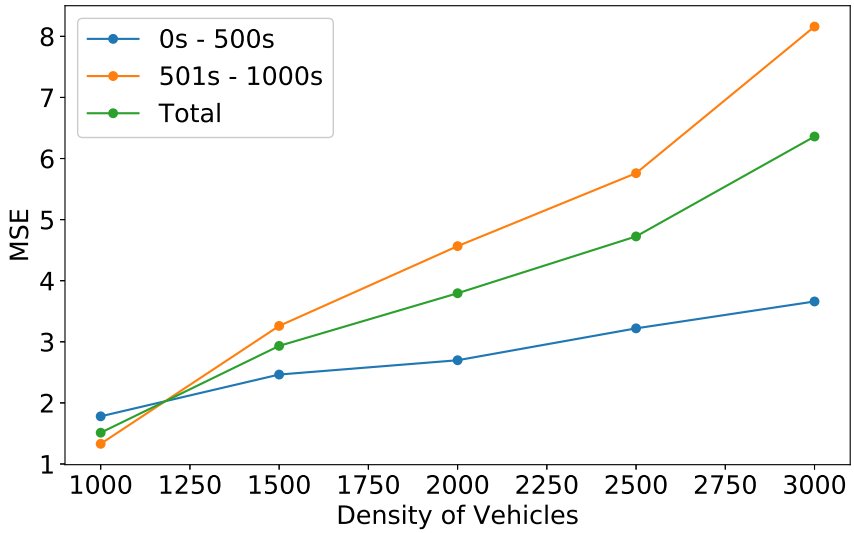

Figure 7.6: MSE analysis in estimations in two periods of simulation ($0s - 500s$ and $500 - 1000s$) for # of vehicles with disrupted connectivity

observed the estimates are less deviant for 2000 vehicles.

The result in Figure 7.2 and Figure 7.3 allows us to visualize the accuracy of the model in estimating the status of vehicles as they move through the region. Thus, it only serves to exemplify/illustrate through a snapshot. For us to show precision, we have used mean squared error (MSE) estimator to measure the average of the squares of the errors for the predictions. For n number of predictions and predicted amount of p and actual value of a we determine the MSE as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(|p_i - a_i|)^2 \tag{7.1}$$

Figure 7.5 and Figure 7.6 shows the error measurement for both variation for vehicle estimation. For better understanding of the behaviour of our model, we have split the analysis into three curves. We measured the error for the first half of the simulation, the second half of the simulation and the average of whole simulation time. For continuous connectivity, the overall performance of the system is better at the later half of the simulation. We have also analysed our results for over and underestimation as shown from Figure 7.8 to Figure 7.13. It can be concluded from the results that the model tends to overestimate in most cases and the deviation of over estimation is much higher than the underestimation. The overestimation is partially caused by the topology of the urban environment. The model does not consider the topological driving distance and congestion in the scenario. As a result,

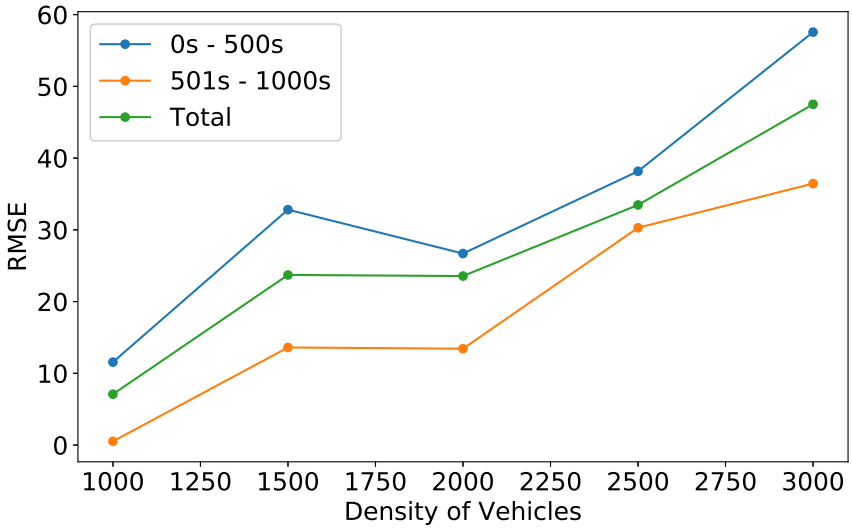

Figure 7.7: MSE analysis in estimations in two periods of simulation ($0s - 500s$ and $500 - 1000s$) for dwell time estimation

vehicles estimated to be in range are in fact not arriving the RSU range within the time window of the prediction. This finding gives us the opportunity to adapt our model accordingly in our future works.

Based on six RSUs placed in different locations and vehicle densities of 1000, 1500, 2000, 2500 and 3000, we achieved an MSE of 2.288 for continuous connectivity and 3.865 for disrupted connectivity.

Dwell Time estimation

Figure 7.4 shows the difference between the dwell time estimation from the model and the actual dwelltime for a particular RSU. The x-axis denotes the simulation time when the prediction was made, and y-axis shows the difference in seconds between the estimation and the actual value of the dwell time. We can observe that the difference drastically decreases as time passes for any density of vehicles. Figure 7.7 shows the error measurement for the dwell time estimation on the first half, second half, and the whole of the simulation time. We can see a general trend of getting more errors with the increase of number of vehicles. However, for any density of vehicles, we can notice that the system tends to be more precise in 500 - 1000s than 0 - 500s mostly because the second simulation period accumulates a longer history, facilitating a more precise estimation. We conducted the experiment for only 1000 seconds and from the achieved result, we concluded that the longer we keep training the model the more accurate it becomes and performs with more precision over time.

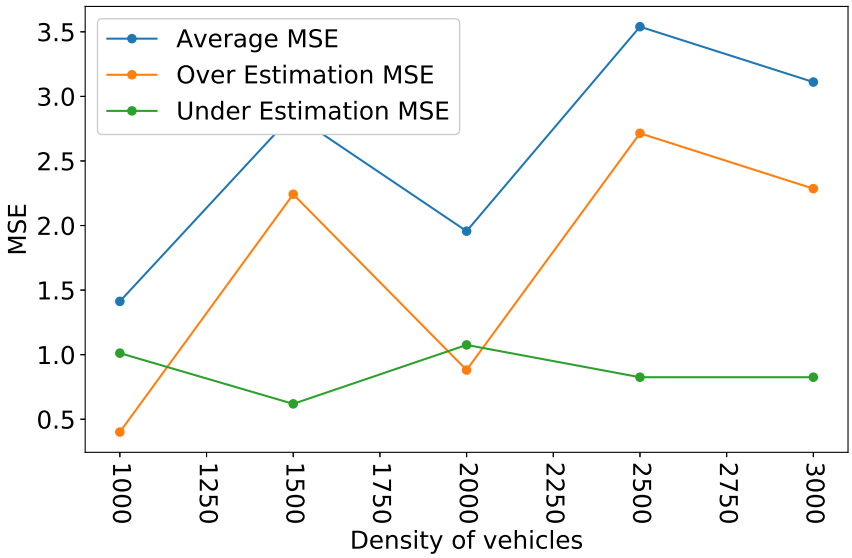

Figure 7.8: MSE analysis (general, over and under estimations) for the number of vehicles (Continuous connection 0s-500s)

Number of control messages (WSM)

Figure 7.14 shows the amount of sent and received WSMs as well as the total along with their confidence intervals in comparison with the amount of vehicles in the system. The number of WSMs sent increases linearly with the growth in the number of vehicles. The number of received WSMs is lightly affected by the number of vehicles.

7.4 Performance analysis for RL based resource allocation model

In this work we have theoretically presented our proposed model and discussed the possible contribution it may provide in resource allocation problems in vehicular Fog computing. Running extensive simulations and measuring the results for performance analysis is therefore the cornerstone in proving the contributions of our work. This section describes the parameters, metrics, and the performance analysis for the RL model.

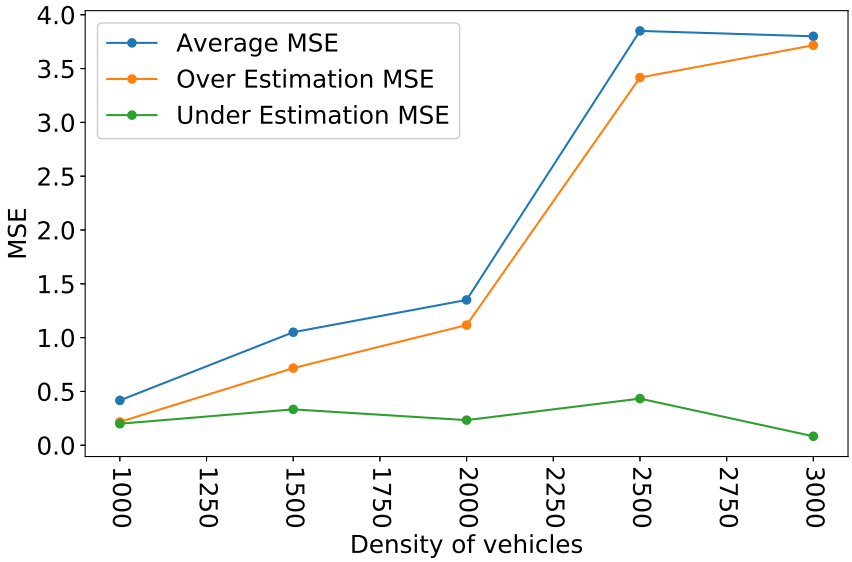

Figure 7.9: MSE analysis (general, over and under estimations) for the number of vehicles (Continuous connection 0s-500s)

7.4.1 Parameter Settings

We carry over the same features and parameters for the simulation environment from the resource estimation model. Additionally, for the RL model, we assume the Fogs in which an application request arises is a state and all the other available RSUs are actions. Table 7.2 summarizes the parameters for the Q-learning algorithm. We want the training period of our model as minimal as possible to start producing intelligent decisions but not to the point where is continuously makes mistake. We choose a learning rate of 0.75 and discount factor of 0.85 by the method of trial and error.

We use VEINS, SUMO and OMNET++ as described in 7.1 to simulate the proposed RL model for resource allocation. We use the map of the metropolitan area of Cologne, Germany for the simulation. We use a pre-recorded vehicle movement data to simulate real world traffic scenario. We place 6 RSUs on the simulation grid randomly. We placed the RSUs based on a visual inspection of the map determining where the vehicle clusters are more prominent. Figure 7.1 shows the map of Cologne and the positions of the RSUs while running the simulation. Table 7.1 shows the parameters used for this simulation environment.

7.4.2 Performance Metrics

In this section we describe the metrics we have used to analyse the performance of the RL based resource allocation model leveraging the CDT estimation. Creating

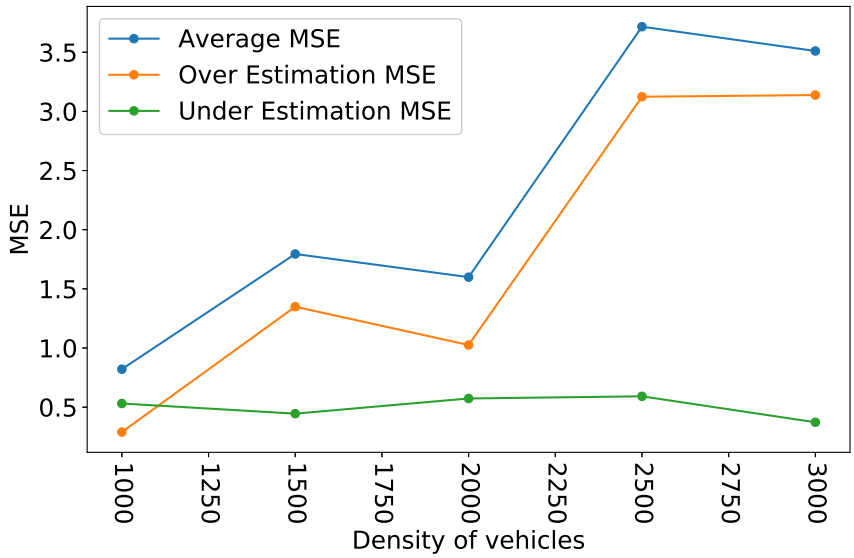

Figure 7.10: MSE analysis (general, over and under estimations) for the number of vehicles (Continuous connection 0s-1000s)

Table 7.2: Q-learning Parameters

Parameter	**Value**
Reward Matrix (R)	6×6
Q Matrix (Q)	6×6
Learning Rate (α)	0.75
Discount Factor (γ)	0.85

appropriate performance metrics enables us to prove the accuracy of our model.

Success of application requests

The key performance metric for the analysis of our proposed model is the success rate of application service by the Fogs. There can be three possible states of each application in the system: **denied** due to lack of resource, assigned to a particular Fog, and assigned but **failed** to execute. As our model makes decision based on future resource availability estimation, it is possible that the model over or underestimate the amount of resources for each Fog. Therefore, for an under estimation the RSU could deny an application request, whereas on an overestimation, the RSU can assign an application resources that might not exist in the short-term future. Therefore, it is of utmost importance to measure the ratio of success and failure of application service to understand the performance of the model. We also compare the results with established vehicular resource allocation model to further establish the accuracy

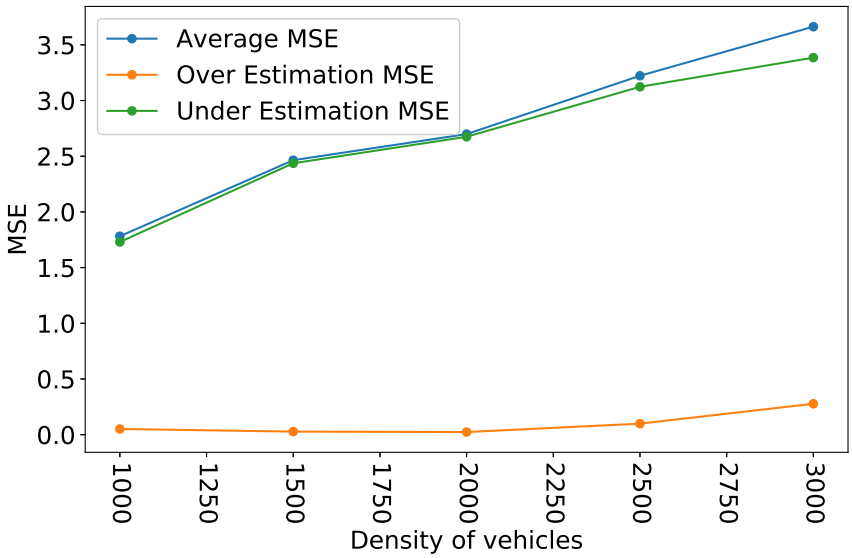

Figure 7.11: MSE analysis (general, over and under estimations) for the number of vehicles (Disrupted connection 0s-500s)

of our proposed model.

Convergence

We want the RL algorithm to come to a point where most of the decisions taken by the model are optimal and from the given parameters of the environment it is evident that more better result is not possible. The performance of an RL algorithm can be measured by the time it converges. The faster the algorithm converges the less computation it needs and thereby is proven to be a better algorithm.

Comparison with established methods

In order to show the advantage of using reinforcement learning for resource allocation we show the performance improvements for our model over simple Analytic Hierarchy Process (AHP) based resource allocation model. A part of our model uses an AHP process to determine priority scores for available fogs while allocating resources. Although in our model we don't just select the Fog with highest score, we are able to record the result of the AHP method. Therefore, after the simulation is complete we can then compare the results of the RL based model with the AHP method. This helps us clearly show the performance improvement in using a Q-learning based method for choosing Fogs over AHP.

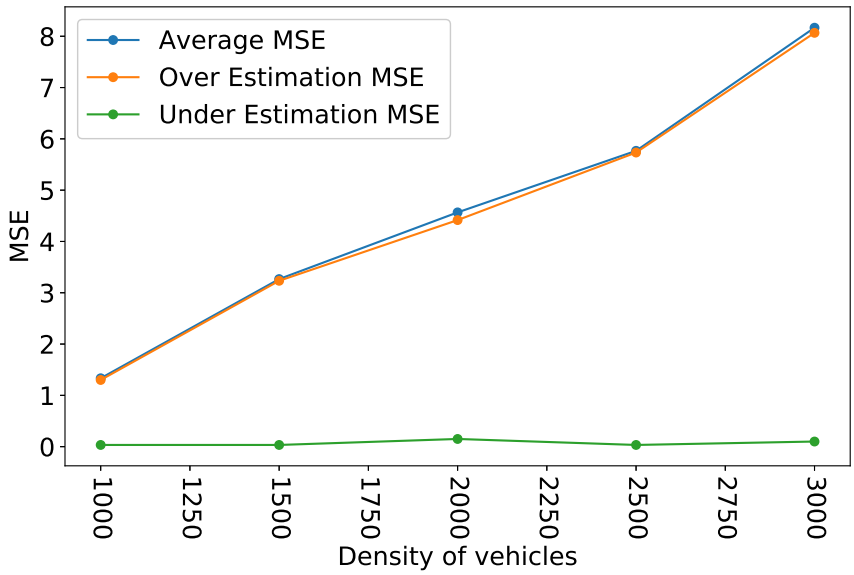

Figure 7.12: MSE analysis (general, over and under estimations) for the number of vehicles (Disrupted connection 500s-1000s)

Control Overhead

While running the simulation we keep a record of WSM messages exchanged between the vehicles and the the RSU. It is important to understand the control overhead requirement of the system to understand the scalability and sustainability of the model.

7.4.3 Scenario and Methodology

As our model is based on real time values and we need to train the RL agent periodically, we need to update the R-matrix repeatedly. Therefore, the RSUs holds the information regarding all the application requests and their service status creating a large application log over time. The RSUs are able to acquire these information through the application profiler, which records all information regarding any application generated within the system. The RSU can also learn the actual resource availability through the resource manager. Subsequently, the RSUs also hold the data for initial fog selection using the simple AHP method. After the simulation is completed we can then simply conduct the necessary analysis to measure the performance of our RL model and compare the performance with AHP model.

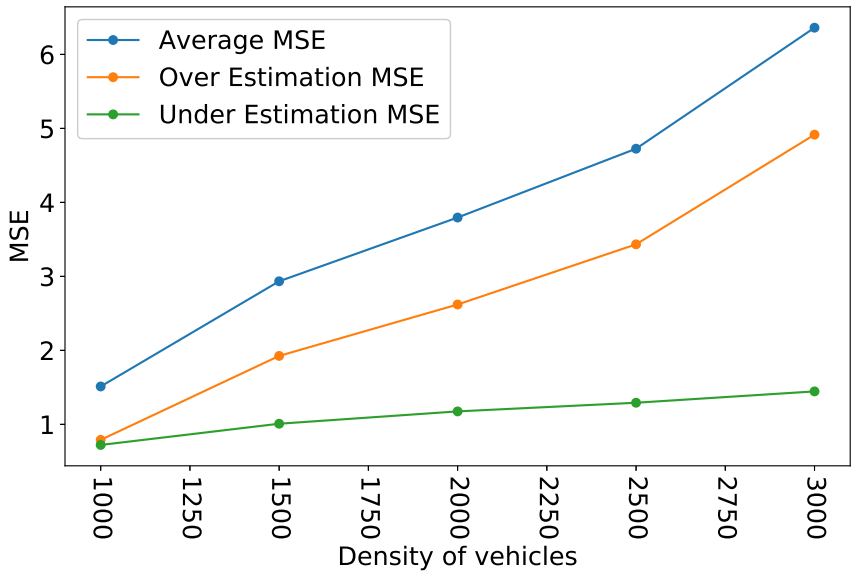

Figure 7.13: MSE analysis (general, over and under estimations) for the number of vehicles (Disrupted connection 0s-1000s)

7.4.4 Convergence

One of the key metrics of a Q-learning based model is the measurement of the performance converge of the algorithm. The convergence is defined as the learning point of a model when it is no longer producing substantially better results than the previous iterations. To update the reward matrix while running the experiment we have coupled the model with an update function. In addition to updating the R and Q-values, the method also keeps track of the running error rates of the decisions made by the model. On an episode $Epoch_i$, we compare the average error rate from $Epoch_{i-5}$ to $Epoch_{i-3}$ and average error from $Epoch_{i-2}$ to $Epoch_i$. We set a threshold of 3 percent error to decide the convergence of the algorithm. When the difference between the two measurements no longer exceeds the threshold, we conclude that the model has converged to an optimal result. If the algorithm produces erratic results and does not converge by the end of the simulation, we conclude it as a failed experiment.

7.4.5 Results

Status of application service

Figure 7.15 shows a visual comparison of the number of application served for various amounts of vehicles using simple AHP method and reinforcement learning technique. For any number of vehicles we can see that the amount of applications served by the

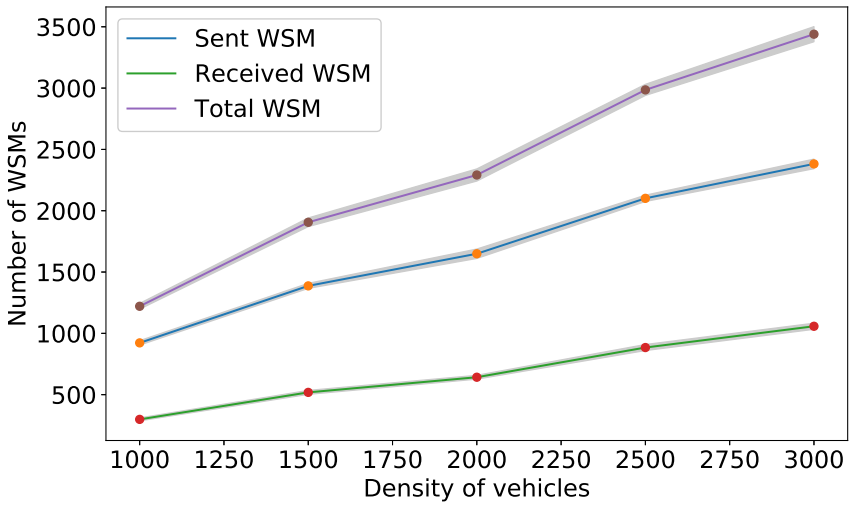

Figure 7.14: WSM sent and received in comparison with vehicle number

RL model is higher than that of AHP model and appears to provide better results for higher number of vehicles. Similar behaviour can be observed in terms of number of application denials depicted in Figure 7.16. It is evident from the figure that with the increase of vehicles, simple AHP model tends to fail a larger number of applications in our scenario than the employed RL model does. Figure 7.17 shows the number of applications denied for both AHP and RL model. We can see that the application denial is much higher for 1000 and 1500 vehicles. This occurrence is due to the fact that when there are fewer vehicles, the amount of resources available is also less, resulting in immediate denial of service whenever the application request comes in.

The overall goal of our proposed reinforcement learning based resource allocation model is to aid vehicular fog in choosing the right source for resource allocation to application service requests. The aforementioned results show a good comparison with an established resource allocation model, and it can be concluded that using Q-learning technique produces better results than using simple AHP based fog selection technique. However, the mentioned figures only illustrates the upside of using our proposed model. Therefore, to measure the performance and accuracy of our model, we have calculated the error rate for all the decisions taken by the RL agent. For a decision of Fog selection by the RL model, there can be two outcomes: application served and application failed to execute. When an application is assigned but it failed to execute we count that an error from the system. For 6 of the Fogs, we can mathematically calculate the average error rate considering 1000, 1500, 2000, 2500, and 3000 vehicles as follow:

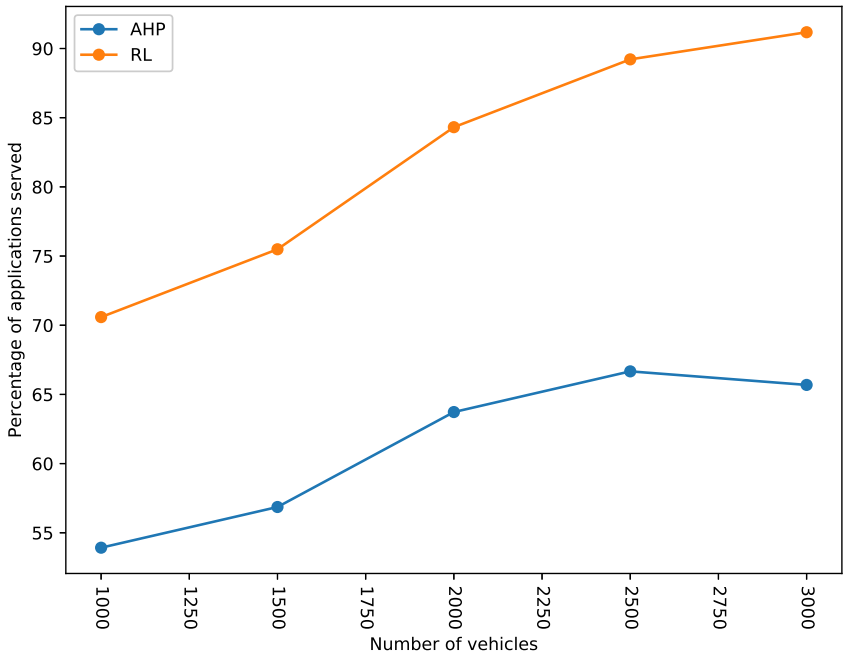

Figure 7.15: Comparison of percentage of applications served.

$$AER = \frac{1}{6} \sum_{i=1000}^{3000} (\frac{f}{T} \times 100) \tag{7.2}$$

Where f is total failed applications and T is the total number of decisions made. After running simulations for each variations of vehicle density extensively, we could achieve an error rate of 9.27 percent. Figure 7.18 shows the error rate for different amount of vehicles in different segments of time 0s-500s, 501s-1000s and the whole time 0s-1000s. It is clear from the illustration that the error rate in the second segment of the simulation is much lower than that of the first segment. As an RL agent learns over time to make optimal decisions, it is evident that the learning capability of our proposed RL model is evident and it makes more accurate decisions over time. The error rate is higher for fewer vehicles, which can be attributed to the over estimation of resources discussed in 7.3.4. However, the error rate appears to drop as the number of vehicles increases, which shows the adaptability and scalability of our proposed model. When there are more number of vehicles, there are more available resources within each of the Fogs. Therefore, the Fog selection method has more fogs to choose from and the possibility of the application of the successful is higher which results in the decrease in error rates.

Figure 7.19 shows the convergence of the Q-learning algorithm. The x-axis of the

figure represents the simulation time and the y-axis represents the error rates. We have plotted 5 curves for 5 variations of vehicle density. We can observe from the figure that the convergence happens at around 800 seconds into the simulation. For certain lower vehicle densities, we can see that the algorithm is still evolving and improving. As we have conducted the experiment for only 1000 seconds, we have set a rather high threshold of convergence to 3 percent. Also as the simulation time is short, we have set a high learning rate of 0.75 for the model. Setting a high learning rate can sometime produce worse result in a short period of time, which is not the case in this scenario. However, it is to be noted that it is always ideal to use a low learning rate and running the training process for longer period of time. It is evident from the figure that by giving more time into the experiment, the model has the potential to improve even further.

Number of control messages (WSM)

Figure 7.20 shows the amount of sent and received WSMs as well as the total along with their confidence intervals in comparison with the amount of vehicles in the system. The number of WSMs sent increases linearly with the growth in the number of vehicles and is evidently more than that of CDT estimation model. The extra amount of messages are the result of application requests and the status message sent back to the vehicles. The number of received WSMs is lightly affected by the number of vehicles as the packet collision in the wireless network increased with the amount of message exchange.

Type/size of service requests

We have classified the application requests into demanding applications and light applications based on their resource requirements. Fig 7.21 shows the amount of applications for each type and the total number of application requests generated per density of vehicles. The x-axis denotes the density of vehicles and the y-axis shows the corresponding numbers. We can observe from the figure that the number of application requests increases with the number of vehicles in the scenario.

7.4.6 Statistical Significance

We have also conducted statistical significance test for the performance of RL model with the simple AHP based method. We have used p-value estimation for hypothesis testing and used the simple AHP based model as a null hypothesis and the RL model

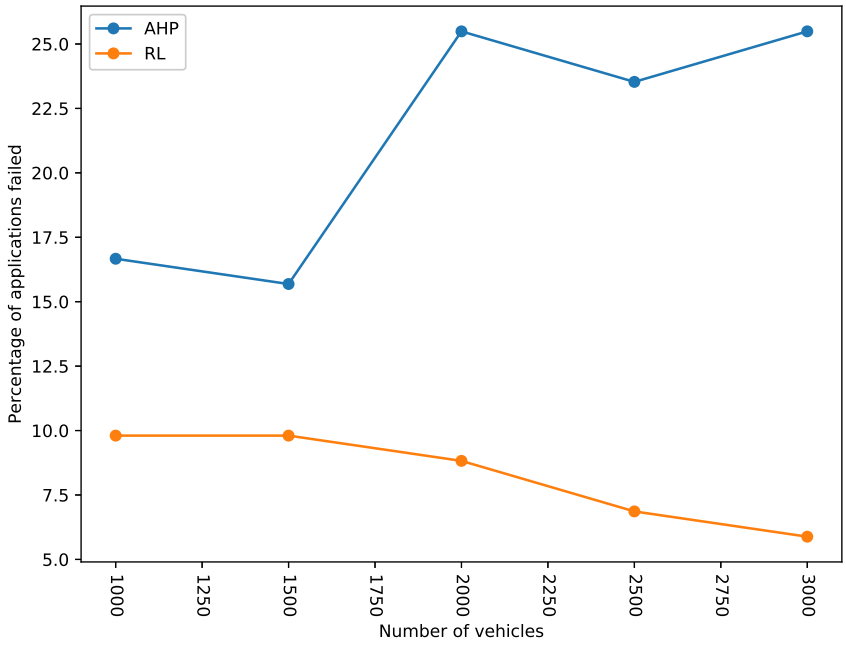

Figure 7.16: Comparison of percentage of applications failed

as the alternate hypothesis. For the estimation, the lower the p-value is, the lower the probability of the results being purely by chance. Before starting the testing we have set a p-value threshold of 0.05. Considering all variations of vehicle density from 1000 to 3000 we have obtained a p-value of 0.0023. Therefore, by rejecting the null hypothesis we conclude that the performance enhancement we have observed by the RL model for resource allocation is statistically significant and not by chance.

7.4.7 Remarks

Extensive analysis of the data regarding the resource availability and application service shows that our proposed model is able to provide optimal solution to the resource allocation problem in highly dynamic vehicular environment. We have conducted the experiment for only 1000 seconds which is not quite sufficient for a reinforcement learning model to converge. However, in this limited time we could show that the more time passes the model tends to improve in selecting the correct fogs to assign resources from for successful application service. Therefore, we conclude that our model is capable of handling large scale application requests and evolve over time to produce the best possible resource allocation policy.

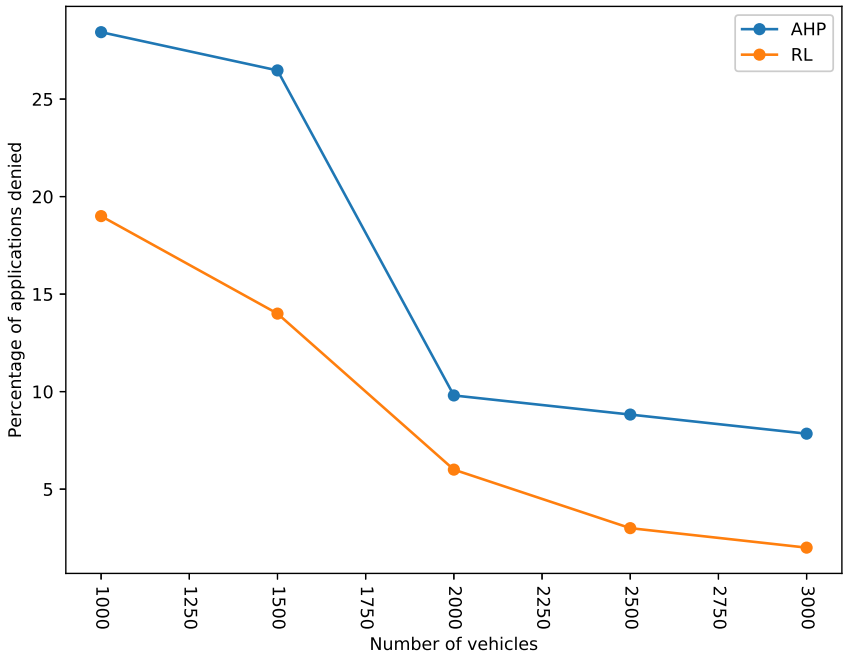

Figure 7.17: Comparison of percentage of applications denied

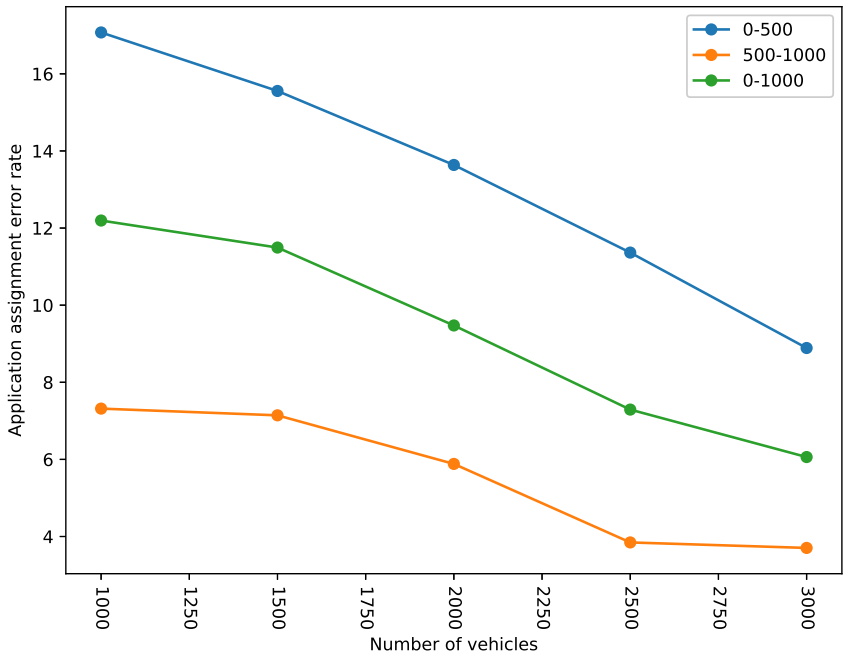

Figure 7.18: Application assignment error rate for different segment of time

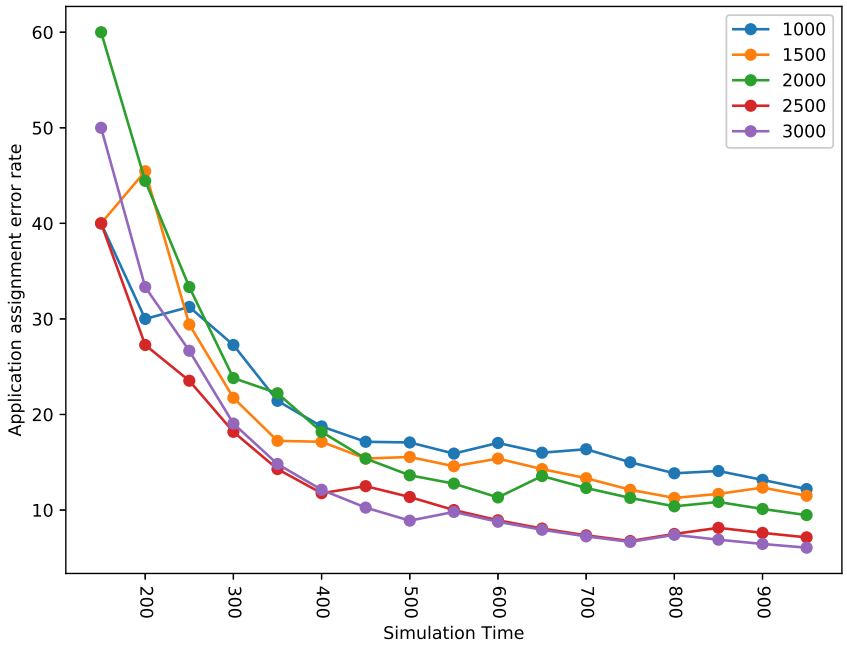

Figure 7.19: Convergence of the Q-learning algorithm over time

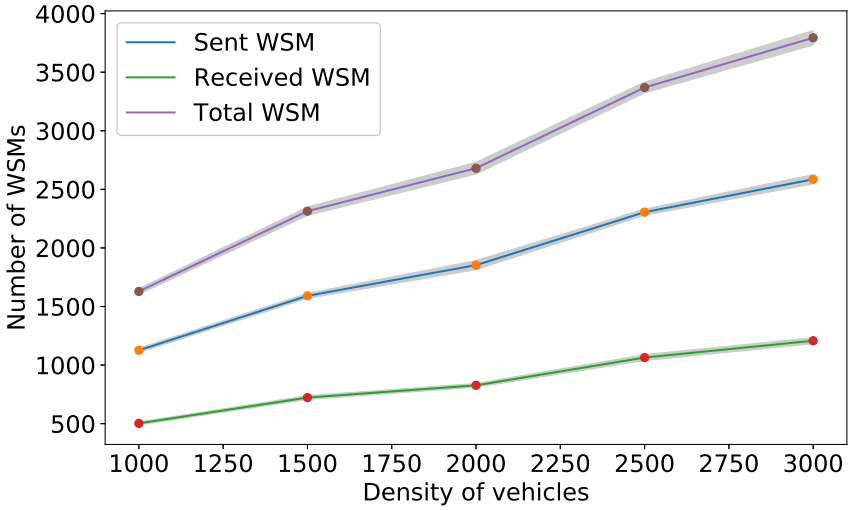

Figure 7.20: Control overhead

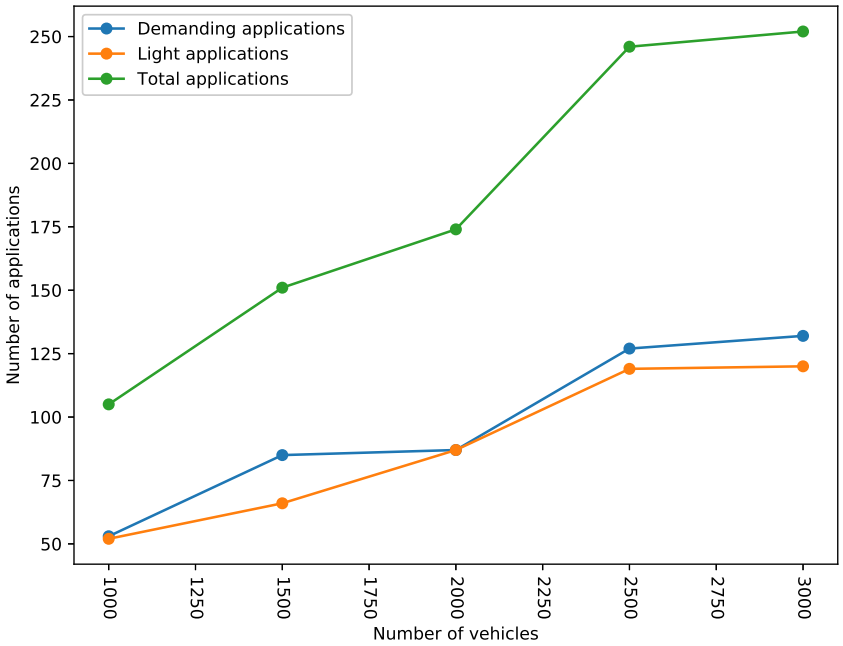

Figure 7.21: Types of application requests

Chapter 8

Conclusion

8.1 Summary

This work has handled the problem of precisely estimating the amount of available resources in range for building and serving applications in a vehicular Fog. With the increase in number of heterogeneous applications with limited computation power of on-board vehicular computation units, optimal resource allocation mechanism in vehicular scenario is a key factor to unlock the full potential of vehicular networks. The proposed mobility-oriented model described in Chapter 6 targets the accurate prediction of number of connected vehicles and their corresponding dwell time under the coverage of an RSU. Accurate estimation of the dwell time for each vehicle with available resources to share is important to estimate the highest and lowest possible vehicular resource pool that can be accumulated within a certain period of time. To accomplish this, we have implemented a real time vehicle trajectory predictor using LR and MLR to model traffic flow in the environment. In a highly dynamic scenario, we are able to predict the amount of connected vehicles to a Fog and the longest time they can provide resources to the resource pool. From these estimations, an estimation of possible vehicular resources within a Fog can be calculated. Results have demonstrated that the estimates could be leveraged to assess vehicular applications' feasibility in advance. A reactive-based system only asses vehicular applications' feasibility only when a request comes in. Therefore, completing all the necessary calculations and estimations in advance, our proposed model largely reduce latency and improves application service performance.

Solely relying on CDT estimation limits the appropriate management of vehicular applications since the only aspect that is considered in the allocation problem is mobility. Experimental analyses have demonstrated that the CDT model presents a few

performance issues with the increasing number of vehicles in the scenario and tend to over and under estimate available resources due to high mobility of the environment. Consequently, we incorporated characteristics of applications and resources/services in the process of assigning resources from available Fogs. Due to the high dynamicity of the vehicular environment, such a process requires adaptability where needs and availability constantly change. Therefore, in Chapter 7 we have introduced a reinforcement learning based dynamic resource allocation model based on CDT estimation and resource availability for vehicular Fog computing. We have taken the estimation model that have already shown good results and extended it further using Q-learning technique and employed an optimal policy selection method for application service. We have used AHP to determine a possible set of Fogs to allocate resources from, and choose the best fog according to the feedback given in previous iterations. This iterative process keeps improving the model and ultimately evolves to be an optimal policy selection mechanism. We have conducted experiments and analysed the results to show the accuracy of our model and compared it with established work on tackling resource allocation problems.

8.2 Future Research Directions

Even though the simulation of the proposed model has shown promising results in terms of accurately predicting resource amount and assignment and execution of applications by selectively choosing best suited fogs, there are several issues that can be improved to obtain more precise results.

The direction and mobility of vehicles in an urban scenario is highly dynamic and accurately predicting the vehicle trajectory based on limited mobility data proved to be challenging. We have used geometric analysis and multiple linear regression method to model the vehicle trajectory, whereas the topology on the urban centers are often complex and proved to be a challenging factor in trajectory estimation. Although we have used reinforcement learning method in the later part of our model to choose best possible fog for application assignment, the application of RL in cloudlet dwell time and resource estimation method appears to be a great solution to overcome the challenges and improve the overall results. We also want to experiment on dynamic weights of the influence factors which can keep changing over time. Improvement on some of the key steps of the proposed model can produce even better results and is a major research goal for us in the future.

Bibliography

[1] Scarborough-research. teen mall shopping attitudes and usage survey. 2005.

[2] A. Alrawais, A. Alhothaily, C. Hu, and X. Cheng. Fog computing for the internet of things: Security and privacy issues. *IEEE Internet Computing*, 21(2):34–42, 2017.

[3] S. Arif, S. Olariu, J. Wang, G. Yan, W. Yang, and I. Khalil. Datacenter at the airport: Reasoning about time-dependent parking lot occupancy. *IEEE-T on Parallel and Distributed Systems*, 23(11):2067–2080, 2012.

[4] A. Ashok, P. Steenkiste, and F. Bai. Vehicular cloud computing through dynamic computation offloading. *Comp. Comms.*, pages 125–137, 2018.

[5] R. Bellman. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences*, 38(8):716–719, 1952.

[6] R. E. Bellman. *Dynamic Programming*. Dover Publications, Inc., USA, 2003.

[7] S.K. Bhoi and P.M. Khilar. Vehicular communication: A survey. *Networks IET, vol. 3*, pages 204–207, 2014.

[8] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16, 2012.

[9] P. J. Burke. The output of a queuing system. *Oper. Res.*, 4(6):699–704, December 1956.

[10] H. El-Sayed, S. Sankar, M. Prasad, D. Puthal, A. Gupta, M. Mohanty, and C. Lin. Edge of things: The big picture on the integration of edge, iot and the cloud in a distributed computing environment. *IEEE Access*, 6:1706–1717, 2018.

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

[12] Md T. Hossain and R. de Grande. Cloudlet dwell time model and resource availability for vehicular fog computing [accepted]. In *proceedings of the International Symposium on Distributed Simulation and Real Time Applications*, 2021.

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

[14] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *CoRR*, cs.AI/9605103, 1996.

[15] W. Kim and M. Gerla. Navopt: Navigator assisted vehicular route optimizer. In *2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 450–455, 2011.

[16] Emmanouil Koukoumidis, Dimitrios Lymberopoulos, Karin Strauss, Jie Liu, and Doug Burger. Pocket cloudlets. *SIGARCH Comput. Archit. News*, 39(1):171–184, 2011.

[17] H. Liang, X. Zhang, J. Zhang, Q. Li, S. Zhou, and L. Zhao. A novel adaptive resource allocation model based on smdp and reinforcement learning algorithm in vehicular cloud system. *IEEE TVT*, 68(10):10018–10029, 2019.

[18] C. Lin, D. Deng, and C. Yao. Resource allocation in vehicular cloud computing systems with heterogeneous vehicles and roadside units. *IEEE Internet of Things Journal*, 5(5):3692–3700, 2018.

[19] Y. Liu, H. Yu, S. Xie, and Y. Zhang. Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks. *IEEE TVT*, 68(11):11158–11168, 2019.

[20] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wiessner. Microscopic traffic simulation using sumo. In *21st International Conference on Intelligent Transportation Systems*, pages 2575–2582, 2018.

[21] R. I. Meneguette and A. Boukerche. A cooperative and adaptive resource scheduling for vehicular cloud. In *2017 IEEE Symposium on Computers and Communications (ISCC)*, pages 398–403, 2017.

[22] R. I. Meneguette, A. Boukerche, and R. de Grande. Smart: An efficient resource search and management scheme for vehicular cloud-connected system. In *IEEE Global Communications Conference*, pages 1–6, 2016.

[23] H. Mousannif, I. Khalil, and S. Olariu. Cooperation as a service in vanet: Implementation and simulation results. *Mobile Information Systems*, 8:153–172, 2012.

[24] S. Olariu, M. Eltoweissy, and M. Younis. Towards autonomous vehicular clouds. *ICST Trans. Mobile Comms. Apps.*, 11:e2, 09 2011.

[25] R. S. Pereira, D. D. Lieira, M. A. C. da Silva, A. H. M. Pimenta, J. B. D. da Costa, D. Rosário, and R. I. Meneguette. A novel fog-based resource allocation policy for vehicular clouds in the highway environment. In *IEEE Latin-American Conference on Comms.*, pages 1–6, 2019.

[26] Q. Qi, J. Wang, Z. Ma, H. Sun, Y. Cao, L. Zhang, and J. Liao. Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach. *IEEE TVT*, 68(5):4192–4203, 2019.

[27] T.L. Saaty. *Fundamentals of Decision Making and Priority Theory With the Analytic Hierarchy Process.* AHP series. RWS Publications, 2000.

[28] M. A. Salahuddin, A. Al-Fuqaha, and M. Guizani. Reinforcement learning for resource provisioning in the vehicular cloud. *IEEE Wireless Communications*, 23(4):128–135, 2016.

[29] M. K. Saroa and R. Aron. Fog computing and its role in development of smart applications. In *proceedings of the IEEE International Conference on Big Data and Cloud Computing*, pages 1120–1127, 2018.

[30] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23, 2009.

[31] R Shrestha, R. Bajracharya, and S. Y. Nam. Challenges of future vanet and cloud-based approaches. *Wireless Communications and Mobile Computing*, 2018.

[32] C. Sommer, R. German, and F. Dressler. Bidirectionally coupled network and road traffic simulation for improved ivc analysis. *IEEE-T on Mobile Computing*, 10(1):3–15, 2011.

[33] P. Sun, A. Boukerche, and R. W. L. Coutinho. A novel cloudlet-dwell-time estimation method for assisting vehicular edge computing applications. In *IEEE Global Comms. Conference*, pages 1–6, 2019.

[34] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[35] I. J. Taylor and A. B. Harrison. *Gnutella*, pages 181–196. Springer London, London, 2009.

[36] A. Varga and R. Hornig. An overview of the omnet++ simulation environment. ICST, 2010.

[37] C. Wang, Y. Li, D. Jin, and S. Chen. On the serviceability of mobile vehicular cloudlets in a large-scale urban environment. *IEEE-T on Intelligent Transportation Systems*, 17(10):2960–2970, 2016.

[38] C. Watkins. Learning from delayed rewards. 01 1989.

[39] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992.

[40] H. Ye, G. Y. Li, and B. F. Juang. Deep reinforcement learning based resource allocation for v2v communications. *IEEE TVT*, 68(4):3163–3173, 2019.

[41] R. Yu, X. Huang, J. Kang, J. Ding, S. Maharjan, S. Gjessing, and Y. Zhang. Cooperative resource management in cloud-enabled vehicular networks. *IEEE-T on Industrial Electronics*, 62(12):7938–7951, 2015.